

Nagios Information Provider

Nagios Information Provider

Abstract

This information provider gathers data from a Nagios system and publishes it into the WS MDS in the standard GLUE schema. The Nagios WS MDS Information Provider gets its information directly from the Nagios v. 2.x `status.dat` file. (Support remains for the v 1.3 `status.log` file in the form of the alternate script `globus-mds-cluster-nagios1.3`) It includes some custom plugins that return the data in a format that the information provider can use. The official Nagios site states the following:

Nagios is a host and service monitor designed to inform you of network problems before your clients, end-users or managers do. It has been designed to run under the Linux operating system, but works fine under most *NIX variants as well. The monitoring daemon runs intermittent checks on hosts and services you specify using external plugins which return status information to Nagios. When problems are encountered, the daemon can send notifications out to administrative contacts in a variety of different ways (email, instant message, SMS, etc.). Current status information, historical logs, and reports can all be accessed via a web browser.

You can download a PDF version of Nagios Information Provider information [here](#)¹.

¹ nagios.pdf

Table of Contents

1. Release Notes	1
1. Component Overview	1
2. Feature Summary	1
3. Changes Summary	1
4. Bug Fixes	1
5. Known Problems	1
6. Technology Dependencies	1
7. Tested Platforms	2
8. Backward Compatibility Summary	2
9. Associated Standards	2
10. For More Information	2
2. Reference Guide	3
1. Overview	3
2. Prerequisites	3
3. Configuring	3
4. Resource Properties	8
5. Schema	8
6. Security Considerations	8
7. Testing	8
8. Troubleshooting	8
Glossary	10

Chapter 1. GT 4.2.1 Release Notes: Nagios Info Information Provider

1. Component Overview

This information provider gathers data from a Nagios system and publishes it into the WS MDS in the standard GLUE schema. The Nagios WS MDS Information Provider gets its information directly from the Nagios v. 2.x `status.dat` file. (Support remains for the v 1.3 `status.log` file in the form of the alternate script `globus-mds-cluster-nagios1.3`) It includes some custom plugins that return the data in a format that the information provider can use. The official Nagios site states the following:

Nagios is a host and service monitor designed to inform you of network problems before your clients, end-users or managers do. It has been designed to run under the Linux operating system, but works fine under most *NIX variants as well. The monitoring daemon runs intermittent checks on hosts and services you specify using external plugins which return status information to Nagios. When problems are encountered, the daemon can send notifications out to administrative contacts in a variety of different ways (email, instant message, SMS, etc.). Current status information, historical logs, and reports can all be accessed via a web browser.

2. Feature Summary

Features new in release 4.2.1:

- The Nagios WS MDS Information Provider gets its information directly from the Nagios v. 2.x `status.dat` file. (Support remains for the v 1.3 `status.log` file in the form of the alternate script `globus-mds-cluster-nagios1.3`) It includes some custom plugins that return the data in a format that the information provider can use.

3. Changes Summary

This is an existing information provider available from GT4.1.x to GT 4.2.1.

4. Bug Fixes

There are ongoing fixed bugs for this information provider (see Bugzilla).

5. Known Problems

- There may currently be bugs for this information provider (see Bugzilla).

6. Technology Dependencies

This information provider depends on the following GT components:

- [Java WS Core](#)
- [GRAM4](#)

This information provider depends on the following 3rd party software:

- A working Perl installation
- A working Condor installation

7. Tested Platforms

Tested Platforms for this information provider:

- N/A

Tested containers for this information provider

- Java WS Core container

8. Backward Compatibility Summary

This information provider works with all GT4 and WS MDS releases.

9. Associated Standards

Associated standards for this Information Provider:

- N/A

10. For More Information

See [Chapter 2, GT 4.2.1: Nagios Information Provider Reference](#) for more information about this information provider.

Chapter 2. GT 4.2.1: Nagios Information Provider Reference

1. Overview

This information provider gathers data from a Nagios system and publishes it into the WS MDS in the standard GLUE schema. The Nagios WS MDS Information Provider gets its information directly from the Nagios v. 2.x `status.dat` file. (Support remains for the v 1.3 `status.log` file in the form of the alternate script `globus-mds-cluster-nagios1.3`) It includes some custom plugins that return the data in a format that the information provider can use. The official Nagios site states the following:

Nagios is a host and service monitor designed to inform you of network problems before your clients, end-users or managers do. It has been designed to run under the Linux operating system, but works fine under most *NIX variants as well. The monitoring daemon runs intermittent checks on hosts and services you specify using external plugins which return status information to Nagios. When problems are encountered, the daemon can send notifications out to administrative contacts in a variety of different ways (email, instant message, SMS, etc.). Current status information, historical logs, and reports can all be accessed via a web browser.

2. Prerequisites

1. A working Perl installation
2. A working Nagios installation

3. Configuring

The following configuration is required for this information provider

1. Initial setup for Nagios WS MDS Information Provider:

The Nagios WS MDS Information Provider requires access to the status file of a Nagios installation that has been set up to provide the necessary information. Thus, if the WS MDS deployment is not on the same machine as the Nagios deployment, the Nagios' `status.dat` file must be exported in some manner to the machine with the WS MDS deployment. This could be accomplished via NFS or `rsync`, or a method of your choice.

The location of the `status.dat` file on the WS MDS deployment machine must be edited at the top of `$GLOBUS_LOCATION/libexec/globus-mds-cluster-nagios`.

2. Nagios Side Setup:

The Nagios' `services.cfg` file will need service stanzas for each host for each of the MDS plugins. The `service_description` fields by default are `MDS_OS`, `MDS_FS`, `MDS_PROC` and `MDS_MEM`. If there already exist tests that output the information needed, in the format needed, these values can be edited at the top of the `globus-mds-cluster-nagios` file. An example service stanza:

```
define service{
    use                               generic-service
```

```

host_name                tg-c001.uc.teragrid.org
service_description      check MDS_OS
is_volatile              0
check_period             24x7
max_check_attempts       3
normal_check_interval    3
retry_check_interval     5
contact_groups           teragrid-admins, nagios-mail
notification_interval    0
notification_period      24x7
notification_options     w,u,c,r
check_command            check_nrpe
}

```

There are several methods for Nagios to get information from remote hosts. NRPE, NSCA and SNMP are three of them. The Nagios WS MDS Information provider does not care how the information is obtained, as long as there is a correct service in services.cfg for each of the four types for each host.

There are sample plugins for the data discovery found in `$GLOBUS_LOCATION/share/nagios-plugins/check_os.mds`, `check_fs.mds`, `check_proc.mds` and `check_mem.mds`. These are extremely simple shell script plugins--their important feature over other similar nagios plugins is that they output real information on stdout that is then used by the information provider.

3. The Sample Nagios MDS Information Provider Plugins:

The plugins included in the Nagios MDS Information provider package (`check_os.mds`, `check_fs.mds`, `check_proc.mds`, and `check_mem.mds` found in `$GLOBUS_LOCATION/share/nagios-plugins/`). These, or functionally equivalent plugins will need to be installed on each host. As included in the Nagios WS MDS Information Provider package, these plugins expect a symlink to whatami to be present in their installed directory. They can be edited to point to the actual install of whatami.

The major difference between these plugins and other similar plugins that may already exist in your Nagios installation is that they provide specific information used by the Nagios WS MDS Information provider. This information is written to standard output when the plugin returns success. Thus, any plugin that returns the same information in the same format could be used instead.

Plugin Information:

MDS_OS

Takes no arguments, and prints the name of the operating system as determined by whatami -r

MDS_FS

Takes an argument of the mount point of the filesystem to be checked. Prints, in whitespace delimited columns, the name of the device associated with the mount point, the number of 1024-blocks on disk, the number of blocks used, the number of blocks available, the percent capacity, and the mount point. In other words, the output of `df -Pk` for that mount point.

MDS_PROC

Expects the number of CPUs for the host to be passed in as its first argument. If this number does not match the number found in

/proc/cpuinfo, the script will return with an error. If it does match, the script will print to standard out, the number of CPUs, the architecture (as determined by whatami -m) and the clock speed.

MDS_MEM

Takes no arguments, and outputs the total physical memory for your machine (as determined by free -m).

4. MDS Side Setup:

Unfortunately, the **[mds-gluerp-configure]** command does not produce a configuration file for a Nagios information provider. However, you can run **mds-gluerp-configure** as you would for Ganglia and simply edit the *resourcePropertyElementProducerConfig* section to include the correct ClassName and your actual OS-native file path to \$GLOBUS_LOCATION/libexec/globus-mds-cluster-nagios.

```
<ns1:resourcePropertyElementProducers xsi:type="ns1:resourcePropertyElementProducerConf
<ns1:className xsi:type="xsd:string" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
org.globus.mds.usefulrp.rpprovider.producers.ExternalProcessElementProducer</ns1:className
<ns1:arguments xsi:type="xsd:string" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
/PATH/TO/GLOBUS/LOCATION/HERE/libexec/globus-mds-cluster-nagios
</s1:arguments>
```

Ganglia can be configured as follows though it's only used here as a reference. Be sure to edit it as mentioned above for a working Nagios installation.

On resources running Ganglia:

1. Change working dir to \$GL/etc/gram-service-PBS (or -LSF, or -Condor, depending what you installed)
2. Run "mds-gluerp-configure pbs ganglia"

If you're not using PBS, look for \$GLOBUS_LOCATION/globus-scheduler-provider-*. Use the value that appears there. You should see the following output:

```
Successfully wrote configuration output file to:
gluerp-config.xml
```

3. If Ganglia is running on the same (local) host as the container, and on the default port, then you can stop here. Otherwise, you will need to edit the gluerp-config.xml file to change the host and/or port. Open the file and look for the following lines:

```
<ns1:className xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:type="xsd:string">org.globus.mds.usefulrp.glue.GangliaElementProducer
</ns1:className>
<ns1:arguments xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:type="xsd:string">localhost</ns1:arguments>
<ns1:arguments xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:type="xsd:string">8649</ns1:arguments>
```

The last two "ns1:arguments" lines are the host and port parameters, respectively. Change them to match your Ganglia installation host and port.

5. Configure cluster/sub-cluster provider:

Many of the Teragrid clusters are divided into sub-clusters based on some kind of characteristic, like processor speed, available memory, etc. This information will be aggregated and made visible to users on the basis of your site's cluster configuration file.

Copy `$GLOBUS_LOCATION/etc/globus_wsrp/mds_usefulrp/cluster-sample-config.xml` to `$GLOBUS_LOCATION/etc/globus_wsrp/mds_usefulrp/cluster_config.xml` and edit it.

This file is what organizes all of the site specific hosts returned into SubClusters. These SubClusters are groupings of hosts that share enough to be logically grouped in some manner. The groupings are arbitrary and you can have as many SubClusters as desired. Please consult the sample file to see an example of how each site needs to configure their cluster configurations.

6. Configure resource manager provider:

Please make the following edits to `$GL/etc/globus_wsrp/mds_usefulrp/scheduler-info.xml`:

1. Set `LRMSType` to "PBS-Torque", "PBS-OpenPBS", "PBS-PBSPro", "SGE", "Condor", "LoadLeveler", or "LSF"
2. Set `LRMSVersion` to the correct version
3. Set `HostName` to the URL of your GRAM service (like "https://tg-grid1.uc.teragrid.org:8443/wsrp/services/ManagedExecutableJobSer")
4. Set `GatekeeperPort` to the port running a GT4.0.1 gatekeeper [will be the alternate number until the service goes live]
5. Set `TotalCPUs` to the number of CPUs in your cluster

If you do this, you will need to add a transformation in the `SchedulerInfoElementProducer` section of the `gluerp-config.xml` file. That section is line 15-26, and the addition should be added after the `GLUESchedulerElementTransform` element. The transformation has the syntax:

```
<ns1:transformArguments xsi:type="xsd:string"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">/PATH/TO/GL/etc/globus_wsrp/mds_usefulrp/s
```

Here is an example of scheduler-info.xml files, though they contain fake version data - it's just to show what they look like syntactically:

```
<!-- NCSA example: useful for overriding the Info fields only -->
<ce:ComputingElementType xmlns:ce="http://mds.globus.org/glue/ce/1.1">
  <ce:Info ce:LRMSType="Fake-PBS" ce:LRMSVersion="FakeVersion"
    ce:GRAMVersion="4.0.1"
    ce:HostName="http://tg-login1.ncsa.teragrid.org:8888/wsrp/services/ManagedJobFac
    ce:GatekeeperPort="2119" ce:TotalCPUs="12800" />
</ce:ComputingElementType>
```

And here's an example gluerp-config.xml files that have the scheduler-info.xml transformation in the GLUES-chedulerElementTransform: * NCSA example

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- NCSA example: Configuration file for the MDS GLUECE resource property provider -->
<!-- -->

<ns1:ResourcePropertyProviderConfigArray
xmlns:ns1="http://mds.globus.org/rpprovider/2005/08"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns1:ResourcePropertyProviderConfigArray">

  <ns1:resourcePropertyProviderConfiguration xsi:type="ns1:resourcePropertyProviderConf

    <ns1:resourcePropertyName
xmlns:mds="http://mds.globus.org/glue/ce/1.1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsi:type="xsd:QName">mds:GLUECE</ns1:r

    <ns1:resourcePropertyImpl
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:string">org.globus.mds.usefulrp.rpprovider.GLUEResourceProperty</ns1:

    <ns1:resourcePropertyElementProducers
xsi:type="ns1:resourcePropertyElementProducerConfig">

      <ns1:className xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:string">org.globus.mds.usefulrp.rpprovider.producers.SchedulerInfoE

      <ns1:arguments xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:string">libexec/globus-scheduler-provider-pbs</ns1:arguments>

      <ns1:transformClass xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:string">org.globus.mds.usefulrp.rpprovider.transforms.GLUEScheduler

    <ns1:transformArguments xsi:type="xsd:string"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">/scratch/cbacon/INSTALL/etc/globus_wsrf_m

      <ns1:period xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsi:type="xsd:int">300</

  </ns1:resourcePropertyElementProducers>

  <ns1:resourcePropertyElementProducers xsi:type="ns1:resourcePropertyElementProducer

    <ns1:className xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:string">org.globus.mds.usefulrp.rpprovider.producers.URLElementProd

    <ns1:arguments xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:type="xsd:string">http://tg-clumon.ncsa.teragrid.org/glue.php</ns1:arguments>

    <ns1:transformClass xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xsi:type="xsd:string">org.globus.mds.usefulrp.rpprovider.transforms.GLUECom
```

```
<ns1:transformArguments xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:type="xsd:string">./etc/globus_wsrp_mds_usefulrp/cluster_config.xml</ns1:tran
```

```
<ns1:transformArguments xsi:type="xsd:string" xmlns:xsd="http://www.w3.org/2001/XMLSche
  <ns1:period xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsi:type="xsd:int">300</
  </ns1:resourcePropertyElementProducers>
  </ns1:resourcePropertyProviderConfiguration>
</ns1:ResourcePropertyProviderConfigArray>
```

4. Resource Properties

- The data gathered is published as part of the GLUECE RP

4.1. Namespace URI

- The GLUE namespace is: `http://mds.globus.org/glue/ce/1.1`
- The CE namespace is: `http://mds.globus.org/glue/ce/1.1`
- The XML namespace is: `http://www.w3.org/2001/XMLSchema`

5. Schema

- `$(GLOBUS_LOCATION)/share/schema/mds/usefulrp/ce.xsd`

6. Security Considerations

General security considerations associated with the container and all MDS services apply. See: [Aggregator Framework](#).

6.1. WS MDS Aggregator Services (Index Service and Trigger Service) Security Considerations

By default, the *aggregator sources* do not use authentication credentials -- they retrieve information using anonymous SSL authentication or no authentication at all, and thus retrieve only publicly-available information. If a user or administrator changes that configuration so that a service's aggregator source uses credentials to acquire non-privileged data, then that user or administrator must configure the service's aggregator sink to limit access to authorized users.

7. Testing

N/A

8. Troubleshooting

It should be noted that the `$(GLOBUS_LOCATION)/etc/globus_wsrp_mds_usefulrp/cluster_config.xml` file is **required** with the above configuration and, at a minimum, ALL host names **MUST** appear once (and only once) in the cluster-configuration file. If a host is retrieved from the Cluster information provider (whether it's CluMon, Ganglia, Nagios, etc) and it is not located in the cluster-configuration, an error will occur and the cluster information will not appear in the WS MDS Index. The error looks like:

```
2006-04-27 15:30:30,289 ERROR transforms.GLUEXMLOrganizer
[Thread-13,reorganizeGLUEDocument:826] Fatal error!
Unmapped hostname "lear.rcac.purdue.edu" in GLUE input data.
```

The `cfg:Name` in this file should correspond to the hostname as it will be reported by the monitoring system. For instance, if `clumon` refers to "tg-c001", the `cfg:UniqueID` will be "tg-c001". However, this Name should be unique across the Teragrid. For this purpose we can specify a string to append to the hostnames to get the unique names in the `gluERP-config.xml` file.

Here are two example `cluster-config.xml` files. The NCSA example is organized into subclusters based on various host machine characteristics. The Purdue example is not. I generated the `purdue` file by running a "for f in `seq -w 0 512`" loop to output the hostnames. They could be sorted into subclusters by an admin who knew how the machines were distributed.

http://software.teragrid.org/docs/ctss3/globus/4.0.1-r3/examples.mds/cluster_config.xml.ncsa

http://software.teragrid.org/docs/ctss3/globus/4.0.1-r3/examples.mds/cluster_config.xml.purdue

For your "Cluster" Name and UniqueID, please use the output of "tgwhereami|cut -f. -f1-2". For your "SubCluster"s please use the value for your Cluster with appended subcluster information.

For example, for Cluster Name, UniqueID "dtf.ncsa" SubCluster could be "dtf.ncsa-fastio", or "dtf.ncsa-<whatever>"

If you need to add a suffix to the hostnames, add the following entry below the `GLUEComputeElementTransform` element in the `gluERP-config.xml` file (around line 45 in the file):

```
<ns1:transformArguments xsi:type="xsd:string"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">.ncsa.teragrid.org</ns1:transformArgument
```

Note also that if the `transformArgument` is used to modify host names (i.e. transforming "tg-c001" into "tg-c001.ncsa.teragrid.org"), it is *required* that the cluster configuration file contain the post-transformed hostnames rather than the original hostname. (i.e. "tg-c001.ncsa.teragrid.org")

Here are two example `gluERP-config.xml` files. The NCSA example has the suffix transformation because their `Clumon` install does not report the ".ncsa.teragrid.org" suffix on the hostnames. The Purdue example does not have a suffix transformation, because their `Ganglia` server reports on the FQDN of the host.

<http://software.teragrid.org/docs/ctss3/globus/4.0.1-r3/examples.mds/gluERP-config.xml.ncsa>

<http://software.teragrid.org/docs/ctss3/globus/4.0.1-r3/examples.mds/gluERP-config.xml.purdue>

Glossary

A

aggregator source

A Java class that implements an interface (defined as part of the Aggregator Framework) to collect XML-formatted data. WS MDS contains three aggregator sources: the query aggregator source, the subscription aggregator source, and the execution aggregator source.