

# **GT 4.2.1 WS MDS Trigger Service: User's Guide**

---

## **GT 4.2.1 WS MDS Trigger Service: User's Guide**

### **Introduction**

The WS MDS Trigger Service collects information about Grid resources and can be configured to execute a program if the collected data meets certain conditions.

End-users will typically interact with the Trigger Service indirectly, using some mechanism specific to the triggered executable program (for example, an executable program may send mail to an end-user or write a structured log file that will later be read by some other program).

---

---

# Table of Contents

Trigger Service How-tos .....	5
1. Trigger Service - Easy HowTo .....	1
1. Purpose .....	1
2. Prerequisites .....	1
3. Introduction .....	1
4. Trigger Tutorial .....	2
2. MDS Trigger Commands .....	9
1. Create a new trigger - mds-trigger-create .....	9
2. View information about existing trggers - mds-trigger-view .....	9
3. Modify a trgger - mds-trigger-edit .....	10
3. Graphical User Interface .....	12
4. Troubleshooting .....	13
1. No triggers displayed in <b>mds-trigger-view</b> .....	13
2. Trigger never fires .....	13
3. Error Messages .....	13
4. General troubleshooting information .....	13
Index .....	15

---

## List of Tables

2.1. mds-trigger-create options .....	9
2.2. mds-trigger-view options .....	10
2.3. mds-trigger-edit options .....	10
4.1. WS MDS Trigger Service Error Messages .....	13

---

# Trigger Service How-tos

## C

command line clients  
admin,

## E

errors,

## G

GUI  
WebMDS,

## T

troubleshooting,  
tutorial  
basic,

---

# Chapter 1. Trigger Service - Easy HowTo

## 1. Purpose

The purpose of this Easy HowTo is to introduce the GT4/WS MDS component known as the Trigger, as well as to provide an example of setting one up successfully. The current GT 4.2.1 documentation provides a basic reference and will be updated as features are added, but for those of you who would like to get a simple trigger working without going through all of the documentation, this document is for you.

We will be creating a simple trigger from scratch, and setting it up completely. To get the basic idea of how this is done, we will only use elements available in the default GT4 installation to show you how to use triggers.

## 2. Prerequisites

To get the most out of this tutorial, you will need:

- A Globus Toolkit installation
- Some basic familiarity with [XML Path Language \(XPath\)](#).<sup>1</sup>
- A valid X.509 user certificate

## 3. Introduction

The Trigger Service collects information and then performs actions based on that information. The Trigger Service works like this:

1. Administrators use a configuration file to specify the names and locations of *trigger actions*, programs that can be executed by the Trigger Service as a result of trigger conditions being met.
2. Administrators use a service interface to specify what information will be collected by the Trigger Service. This interface is called the Aggregator Framework and is the same configuration interface used by the Index Service
3. Users use a service interface to define *triggers*. A trigger includes (among other things) an XPath query string and the name of one of the trigger actions defined in step 1.
4. The Trigger Service periodically collects data (based on the configuration specified in step 2) and, for each trigger specified in step 3, evaluates the XPath query associated with the trigger and then executes the trigger's action if the query returns true.

In this example, we will configure the Trigger Service to monitor the Default Index Service running in the same Globus container, and then set up a trigger that will add an entry to a log file any time the number of Index Service is less than 1. This is not necessarily a practical example of how you would use a trigger, but it's simple enough to give you a basic idea of how to set one up. So let's get started!

---

<sup>1</sup> <http://www.w3.org/TR/xpath>

## 4. Trigger Tutorial

### 4.1. Preliminaries: Set Up Your Environment

First things first -- in order to run most Globus commands, you must have your environment set up correctly and have a valid proxy certificate. To set up your environment, first set the `GLOBUS_LOCATION` environment variable to the directory in which Globus is installed. To finish setting up your environment, run:

```
. $GLOBUS_LOCATION/etc/globus-user-env.sh
```

if you're a Bourne shell user, or

```
source $GLOBUS_LOCATION/etc/globus-user-env.csh
```

if you're a C shell user. Finally, generate a proxy certificate with:

```
$GLOBUS_LOCATION/bin/grid-proxy-init -verify -debug
```

### 4.2. Configure Trigger Action Programs

Next, we will specify what commands can be used in Trigger Service triggers. The Trigger Service comes with some simple action scripts in the `$GLOBUS_LOCATION/libexec/trigger` directory; we will edit the `$GLOBUS_LOCATION/etc/globus_wsrf_mds_trigger/jndi-config.xml` file to enable them:

```
<?xml version="1.0" encoding="UTF-8"?>
<jndiConfig xmlns="http://wsrf.globus.org/jndi/config">
  <global>
    <resource name="configuration"
              type="org.globus.mds.aggregator.impl.AggregatorConfiguration">
      ...
    </resource>

    <resource name="triggerConfiguration"
              type="org.globus.mds.trigger.impl.TriggerConfiguration">
      <resourceParams>
        <parameter>
          ...
        </parameter>
        <parameter>
          <name>executableMappings</name>
          <value>trigger-action-default=trigger-action-default.sh, trigger-action-input-def
        </parameter>
      </resourceParams>
    </resource>

  </global>

  <service name="TriggerRegistrationService">
    ...
```

```
</service>
...
</jndiConfig>
```

This `jndi-config.xml` file defines an `executableMappings` parameter. The format of the `executableMappings` parameter is a sequence of `name=value` strings, separated by commas. The left hand side of each `name/value` pair is the name that users will specify in trigger definitions; the right hand side is the path name (relative to the `$GLOBUS_LOCATION/libexec/trigger` directory) of the program to execute. In this example, we define two trigger actions: `trigger-action-default` maps to `$GLOBUS_LOCATION/libexec/trigger/trigger-action-default.sh`, and `trigger-action-input-default` maps to `$GLOBUS_LOCATION/libexec/trigger-action-input-default.sh`. These action scripts are distributed as part of the Globus distribution. The version of `$GLOBUS_LOCATION/etc/globus_wsrp/mds_trigger/jndi-config.xml` distributed with Globus has these mappings defined in a commented-out section; in order to continue with this example, you must uncomment them.

Before you continue, you'll need to restart your Globus container to make the changes to `jndi-config.xml` take effect. If you normally use `/etc/init.d/gt4container`, then you can type `/etc/init.d/gt4container restart`, or you can kill the running Globus container (if there is one) and run `$GLOBUS_LOCATION/etc/globus-start-container-detached` by hand. If you have a production container running and want to test the trigger service with a different instance on another port, you can run `globus-start-container-detached -p NNNN` to cause the new container to listen on port `NNNN`.

## 4.3. Configure the Trigger Service to Collect Information

The next thing we will do is configure the trigger service to collect some information (we will later configure the trigger service to periodically run a query on that information and, based on the results of the query, take some action). In this example, we will configure the trigger service to collect information by querying the Default Index Service running in the same Globus container for the entire contents of its index.

The Trigger Service uses the [Aggregator Framework](#) to configure its sources of information. Aggregator sources are configured through a service interface; we will create a file specifying configuration parameters and then run the `mds-servicegroup-add` command to read that configuration file and register the configuration information with the Trigger Service. We will start with the example trigger registration file included with Globus distributions in `$GLOBUS_LOCATION/etc/globus_wsrp/mds_trigger/trigger-registration-example.xml`

```
<DefaultServiceGroupEPR>
  <wsa:Address>https://localhost:8443/wsrp/services/TriggerRegistrationService</wsa:Address>
</DefaultServiceGroupEPR>
<ServiceGroupRegistrationParameters
  xmlns="http://mds.globus.org/servicegroup/client"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:agg="http://mds.globus.org/aggregator/types">
  <RegistrantEPR
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:Address>https://localhost:8443/wsrp/services/DefaultIndexService</wsa:Address>
  </RegistrantEPR>
  <RefreshIntervalSecs>600</RefreshIntervalSecs>
  <Content xsi:type="agg:AggregatorContent"
    xmlns:agg="http://mds.globus.org/aggregator/types">
    <agg:AggregatorConfig xsi:type="agg:AggregatorConfig">
      <agg:GetResourcePropertyPollType
```

```
        xmlns:wssg="http://docs.oasis-open.org/wsrif/sg-2">
        <agg:PollIntervalMillis>30000</agg:PollIntervalMillis>
        <agg:ResourcePropertyName>wssg:Entry</agg:ResourcePropertyName>
        </agg:GetResourcePropertyPollType>
    </agg:AggregatorConfig>
    <agg:AggregatorData />
</Content>
</ServiceGroupRegistrationParameters>
</ServiceGroupRegistrations>
```

For this example, the only items in the registration file that you might need to edit are the `DefaultServiceGroupEPR` (the address of your trigger registration service) and the `RegistrantEPR` (the address of the resource you want to monitor; in this case, your Default Index Service). If these look correct (i.e., the host names and port numbers correspond to the Globus container you just started), then you do not need to edit this file at all.

Finally, run `mds-servicegroup-add` to register these configuration parameters to the Trigger Service:

```
mds-servicegroup-add $GLOBUS_LOCATION/etc/globus_wsrif_mds_trigger/trigger-registration-exa
```

Of course, if you copied the `trigger-registration-example.xml` file before editing it, you would use the name of the edited file instead. The output from `mds-servicegroup-add` should look something like this:

```
Processing configuration file...
INFO: Processed 1 registration entries
INFO: Successfully registered https://localhost:8443/wsrif/services/DefaultIndexService to
```

## Warning

In general, it's a bad idea to use loopback addresses like "localhost" or "127.0.0.1" in MDS configuration files (because if you configure a remote host to poll "localhost" for information, the remote host will poll itself, not the host that the `mds-servicegroup-add` command was run from). We can get away with it here because all the addresses we're using are local, but in real life, it's better to use non-local IP addresses or fully-qualified domain names

## 4.4. Define Triggers

Now, we're going to define a trigger that checks how many Entries are being registered by the Index Service and then takes actions based on the results. But first, let's check how many Entries are being registered by the Index Service. Type the following command in one line (substituting your hostname and port if appropriate):

```
$GLOBUS_LOCATION/bin/wsrif-query -s https://127.0.0.1:8443/wsrif/services/DefaultIndexService
```

On our setup we get: 3.

## 4.5. Create A Trigger

At this point, the Trigger Service is collecting information, but we haven't told it to do anything with that information. We will now follow a few simple steps to set up a trigger.

1. First, we'll create the trigger:

```
$GLOBUS_LOCATION/bin/mds-trigger-create -b https://127.0.0.1:8443/wsrif/services https://
```

The first argument (`-b https://127.0.0.1:8443/wsrfl/services`) specifies the Trigger Service that we want to use. The second argument specifies which monitored resource we would like to act upon. (Note: in this tutorial, we're only monitoring one resource. But we could be monitoring several, by specifying several sets of `ServiceGroupRegistrationParameters` in the configuration file we used with `mds-servicegroup-add`, or by running `mds-servicegroup-add` multiple times with different configuration files).

The client should produce output similar to the following:

```
MDS4 Trigger Creation Client
-----

**      Service URL: https://127.0.0.1:8443/wsrfl/services/DefaultIndexService

Checking current monitored services (Trigger Registrations)...
OK
Address: https://128.9.64.191:8443/wsrfl/services/TriggerService
Reference property[0]:
<ns1:TriggerResourceKey xmlns:ns1="http://mds.globus.org/2007/03/TriggerResourceKey">54
...
--> Trigger has been created.
```

The `TriggerResourceKey` is an identifier created by the Trigger Service to identify this newly-created trigger. It is sometimes also referred to as a Trigger ID.

2. Now we have a trigger, but not a very interesting one. We can see some information about it by typing:

```
$GLOBUS_LOCATION/bin/mds-trigger-view -b https://127.0.0.1:8443/wsrfl/services
```

As above, the `-b https://127.0.0.1:8443/wsrfl/services` specifies the Trigger Service to use. If you specify a Trigger ID, then `mds-trigger-view` will display detailed information about that trigger. In this case, we didn't, so `mds-trigger-view` will display summary information about all triggers. The output should look something like the following:

```
MDS4 Trigger View Client
-----

Monitored Services (Trigger Registrations)

1) /wsrfl/services/DefaultIndexService

TriggerID:      546aae00-418b-11dd-a5ea-ebfac2dfbbee
TRIGGER NAME:   Default Trigger Name
MATCHING RULE:  count(//*[local-name()='Entry'])<1
ACTION SCRIPT:  trigger-action-default
TRIGGER STATUS: disabled
```

This gives us some important information:

- a shorthand reference to the service being monitored (the same one we specified in the `mds-trigger-create` command)
- the newly-created Trigger ID (which we will use in future requests to the Trigger Service)

- the matching rule (an XPath query that returns true if the number of Index Service entries is less than one)
  - the trigger action (if the condition specified in the matching rule is met, then the trigger will run the command mapped to the name `trigger-action-default` in the trigger service's `jndi-config.xml` file). This is a script that will append a line to the file `$GLOBUS_LOCATION/trigger_service_base_action_log`.
  - Most importantly, the trigger status is disabled, which means that the matching rule will not be evaluated, nor will the trigger action be run.
3. The `mds-trigger-edit` command is used to change the trigger's properties (its matching rule, action, enabled/disabled status, etc.). The syntax is:

```
mds-trigger-edit -b baseURL [TriggerID] [Parameter="Value"]
```

Let's enable this trigger. By enabling/activating the trigger, we turn it "on", meaning that it will take the Matching Rule and evaluate this against incoming aggregator data from our monitored service (the Default Index Service).

```
mds-trigger-edit -b https://127.0.0.1:8443/wsrf/services \
    546aae00-418b-11dd-a5ea-ebfac2dfbbee TriggerStatus=enabled
```

Now that this trigger is "enabled", we have an active trigger that is evaluating data. You may notice this in the service container logs if it is running in "debug" mode (You can allow "debug" information by uncommenting: `log4j.category.org.globus.mds.trigger=DEBUG` in your `$GLOBUS_LOCATION/container-log4j.properties` file). However, in a default Globus setup, the Matching Rule for this trigger always evaluates to false, so the trigger will not fire. (The action associated with this trigger appends a line to the log file `$GLOBUS_LOCATION/trigger_service_base_action_log`. You can verify that the trigger is not firing by checking that the file doesn't exist (or that if it does exist, that it hasn't been appended to recently).

Let's change our Matching Rule so that the trigger will evaluate to "true" and cause the trigger to fire.

```
mds-trigger-edit -b https://127.0.0.1:8443/wsrf/services \
    546aae00-418b-11dd-a5ea-ebfac2dfbbee \
    MatchingRule="count(//*[local-name()='Entry'])>0"
```

Typing `mds-trigger-view -b https://127.0.0.1:8443/wsrf/services` will summarize what we've done:

```
MDS4 Trigger View Client
```

```
-----  
Monitored Services (Trigger Registrations)
```

```
1) /wsrf/services/DefaultIndexService
```

```
TriggerID:      546aae00-418b-11dd-a5ea-ebfac2dfbbee  
TRIGGER NAME:   Default Trigger Name  
MATCHING RULE:  count(//*[local-name()='Entry'])>0  
ACTION SCRIPT:  trigger-action-default  
TRIGGER STATUS: enabled
```

To view more details about this particular trigger, type:

```
mds-trigger-view -b https://127.0.0.1:8443/wsrf/services \
    546aae00-418b-11dd-a5ea-ebfac2dfbbee
```

MDS4 Trigger View Client

-----Detailed Trigger Information-----

MONITORED SERVICE: https://127.0.0.1:8443/wsrp/services/DefaultIndexService  
TriggerID: 546aae00-418b-11dd-a5ea-ebfac2dfbbee  
TRIGGER NAME: Default Trigger Name

MATCHING RULE: count(//\*[local-name()='Entry'])>0  
ACTION SCRIPT: trigger-action-default  
TRIGGER STATUS: enabled

ENABLE BOOLEAN: true  
ACTION SCRIPT INPUT FULL ORIGINAL: true  
ACTION SCRIPT INPUT XPATH QUERY RESULT: true

MINIMUM FIRING INTERVAL: 20  
MINIMUM MATCH TIME: 30

START TIME: N/A  
END TIME: N/A

INVALIDITY START TIME: N/A  
INVALIDITY END TIME: N/A

-----Non-editable stats-----

RULE LAST CHECKED AT: 2008-06-23 19:09:18 PDT-0700  
CONDITION TRUE SINCE: 2008-06-23 19:03:48 PDT-0700  
ACTION FIRED AT: 2008-06-23 19:09:18 PDT-0700

Now after a minute or so, you will notice that the trigger has fired successfully. You can verify this by checking the contents of the log file we created in our action script from above:

```
more $GLOBUS_LOCATION/trigger_service_base_action_log
```

This should look similar to the following

```
Trigger Service Entry: Sun Jun 17 14:45:26 CDT 2007  
Trigger Service Entry: Sun Jun 17 14:45:56 CDT 2007  
Trigger Service Entry: Sun Jun 17 14:46:26 CDT 2007  
Trigger Service Entry: Sun Jun 17 14:46:56 CDT 2007
```

There is a 30 second interval that we specified in our aggregator configuration file above. This should probably be lengthened eventually so that you don't have the triggers going off so often.

## 4.6. Congratulations!

You have now successfully setup, configured, registered, created, edited and tested a trigger from scratch!

*Next Steps:* Check out the documentation and create more triggers to perform actions more relevant to your own objectives. Experiment with the XPath queries to expand the possibilities of what you can use them for. If you have questions, feel free to [\[fixme contact us\]](#)!

## 4.7. Troubleshooting

For MDS Trigger troubleshooting information, see [Troubleshooting MDS Trigger](#).

---

# Chapter 2. MDS Trigger Commands

The `mds-servicegroup-add(1)` command in the Aggregator Framework is used to configure the Trigger Registration Service to collect information from various sources. In addition, the Trigger Service has three command-line clients

## 1. Create a new trigger - `mds-trigger-create`

### Synopsis

```
mds-trigger-create [options] -b baseURL monitoredURL
```

### Description

This command creates a new trigger.

**Table 2.1. `mds-trigger-create` options**

<i>-b baseURL</i>	Specify the trigger service's base URL (everything in the Trigger Service URL up to the service name). This option is used instead of the customary <code>-s</code> or <code>-e</code> options because this client communicates with more than one trigger-related service.
<i>monitoredURL</i>	Specify the URL of the service to be monitored; this should be the same as the address of a service registered to the Trigger Registry Service's service group.

### Example

The first command creates a new trigger on the server *triggerhost.org* to monitor the information in the default Index Server running in the same Globus container. The second command creates a new trigger on the server *triggerhost.org* to monitor the information in an Index Server running on the server *otherhost.org*

```
mds-trigger-create -b https://triggerhost.org:8443/wsrf/services \
https://triggerhost.org:8443/wsrf/services/DefaultIndexService
```

```
mds-trigger-create -b https://triggerhost.org:8443/wsrf/services \
https://otherhost.org:8443/wsrf/services/DefaultIndexService
```

## 2. View information about existing triggers - `mds-trigger-view`

### Synopsis

```
mds-trigger-view [options] -b baseURL [TriggerID]
```

### Description

This displays information about triggers.

**Table 2.2. mds-trigger-view options**

<i>-b baseURL</i>	Specify the trigger service's base URL (everything in the Trigger Service URL up to the service name). This option is used instead of the customary <code>-s</code> or <code>-e</code> options because this client communicates with more than one trigger-related service.
<i>TriggerID</i>	If a Trigger ID is specified, detailed information about the specified trigger will be displayed; if not, summary information about all triggers will be displayed.

**Example**

The first command displays summary information about all triggers known to the Trigger Service; the second displays detailed information about one trigger

```
mds-trigger-view -b https://triggerhost.org:8443/wsrf/services
```

```
mds-trigger-view -b https://triggerhost.org:8443/wsrf/services \
546aae00-418b-11dd-a5ea-ebfac2dfbbee
```

## 3. Modify a trigger - mds-trigger-edit

**Synopsis**

```
mds-trigger-edit [options] -b baseURL TriggerID Parameter=Value
```

**Description**

This command is used to modify trigger parameters, in order to change the trigger conditions, actions, status (enabled or disabled), etc.

**Table 2.3. mds-trigger-edit options**

<i>-b baseURL</i>	Specify the trigger service's base URL (everything in the Trigger Service URL up to the service name). This option is used instead of the customary <code>-s</code> or <code>-e</code> options because this client communicates with more than one trigger-related service.
<i>TriggerID</i>	The identifier of the trigger to be modified
<i>Param=value</i>	Set the named parameter to the specified value. The parameter can be any writable Trigger Service <a href="#">resource property</a>

**Examples**

The first command enables a trigger; the second command disables it.

```
mds-trigger-edit -b https://triggerhost.org:8443/wsrf/services \
546aae00-418b-11dd-a5ea-ebfac2dfbbee \
TriggerStatus=enabled
```

```
mds-trigger-edit -b https://triggerhost.org:8443/wsrf/services \
546aae00-418b-11dd-a5ea-ebfac2dfbbee \
TriggerStatus=disabled
```

Change the trigger condition (matching rule) so that the trigger fires if there are no Index Service entries.

```
mds-trigger-edit -b https://triggerhost.org:8443/wsrp/services \  
546aae00-418b-11dd-a5ea-ebfac2dfbbee \  
MatchingRule="count(//*[local-name()='Entry']=0"
```

Change the trigger action.

```
mds-trigger-edit -b https://triggerhost.org:8443/wsrp/services \  
546aae00-418b-11dd-a5ea-ebfac2dfbbee \  
ActionScript=trigger-action-input-default
```

---

# Chapter 3. Graphical User Interface

The release contains WebMDS which can be used to display the status of resources registered to a Trigger Service in a normal web browser.

---

# Chapter 4. Troubleshooting

For a list of common errors in GT, see [Error Codes](#).

## 1. No triggers displayed in mds-trigger-view

I created a trigger using the `mds-trigger-create`, but I don't see any triggers when I type `mds-trigger-view`! Where are the triggers? Why is nothing happening?

Did you set up a trigger registration? (See the Trigger Service [Admin Guide](#)) The trigger has been created (unless there was an error), but you will not see it and you cannot access it if the trigger registration has not been set up.

## 2. Trigger never fires

I'm sure the registration was made properly, but the trigger script never fires. OR I followed all of the above steps, but where are the triggers? Why is nothing happening?

Once you've completed the trigger registration, you can now create individual triggers. Trigger creation is performed using a client. See the [User's Guide](#) for more information on clients.

## 3. Error Messages

**Table 4.1. WS MDS Trigger Service Error Messages**

Error Code	Definition	Possible Solutions
Error ; nested exception is: org.apache.commons.httpclient.NoHttpResponseException: The server xxx.x.x.x failed to respond	Happens when trying to create a trigger for the Trigger Service. The above error is accompanied by the following error in container: [JWSCORE-192] Error processing request java.io.IOException: Token length 1347375956 > 33554432. FIXME - what causes this?	Be sure that you have properly edited the <code>client-config-settings</code> file under <code>globus_wsrf_mds_trigger</code> . The <code>DefaultServiceAddress</code> parameter should properly reflect the service prefix from your container, e.g.: <code>https://127.0.0.1:8444/wsrf/services/</code> . The services you wish to monitor should also be consistent.

WS MDS is built on Java WS Core, please see [Java WS Core Error Codes](#) for more error code documentation.

## 4. General troubleshooting information

- In general, if you want to investigate a problem on your own please see [Chapter 10. Debugging](#) for details on how to turn on debugging.
- Most of the command line clients have a `-debug` option that will display more detailed error messages, including the error stack traces.

- Search the mailing lists<sup>1</sup> such as [gt-user@globus.org](mailto:gt-user@globus.org)<sup>2</sup> or [jwscore-user@globus.org](mailto:jwscore-user@globus.org)<sup>3</sup> (before posting a message).
- If you think you have found a bug please report it in our Bugzilla<sup>4</sup> system. Please include as much as detail about the problem as possible.

---

<sup>1</sup> <http://www.globus.org/email-archive-search.php>

<sup>2</sup> <mailto:gt-user@globus.org>

<sup>3</sup> <mailto:jwscore-user@globus.org>

<sup>4</sup> <http://bugzilla.globus.org/bugzilla/>

---

# Index

## C

command line clients  
  admin, 9

## E

errors, 13

## G

GUI  
  WebMDS, 12

## T

troubleshooting, 13  
tutorial  
  basic, 1