GDMA Advanced

This section is designed for those that have used GDMA previously, have it already set up and working, and would like to use the advanced features of the new version.

Detailed Installation and Setup

Controlling Poller and Spooler Status Messages

The GDMA profiles contain services called gdma_poller and gdma_spooler. By default, these services will receive status messages from the poller and spooler processes. These messages tell you about how much time the poller is using to perform the checks, and whether that time is excessive. The spooler reports transmission failures, and statistics about how many messages it sent, and if any messages were purged, etc. The state of the services will change, as well, based on whether there is an error to report.

These messages are useful, but can be excessive, especially in large configurations. They should be left on in initial deployments, but after calibration is done, you may wish to curtail their use. or even remove the services. *GDMA* adds some parameters to allow you better control over these messages:

- **Poller_Service="gdma_poller"** - This parameter controls the name of the service to send poller messages under, defaulting to gdma_poller. You can change this to another service, possibly combining it with the service for *Spooler* messages (*Spooler_Service*) to direct all *GDMA* messages to one service instead of two.
- **Poller_Status=On/Off** - This controls whether the poller generates the status messages at all. If you turn it off, that host will not generate these messages. You should remove the gdma_poller service from the host in this case.
- **Spooler_Service=On/Off/Updates** - This controls whether the spooler sends status messages, and, if so, when. If it's off, no spooler messages will be sent. If it is on, you will get a message each time the spooler runs, which is every 30 seconds by default. If you set it to Updates, it will send you a message only if there has been an error, or something has changed (such as a recovery from an error). *GDMA* sets this to *Updates* by default.
- **Warning_Threshold, Critical_Threshold=(integer percent)** - Applied to the poller, these control when you will get notified that there is a problem with the poller chewing up more than the indicated percentage of its cycle time to perform a single check. That situation is something you probably want to know about, since it means you may be trying to check too frequently, or to check too many services. Remember, *GDMA* is designed to use very little CPU resources, so it has some built-in waits, and (except in multi-host mode), it actually does not perform checks in parallel. If you are getting close to the limit and you can manage it, you can change these thresholds from their defaults (60 and 80 percent) to cut down on false positives for these services.

Using Fully Qualified Host Names

If you manage multiple data centers, or if you have domains of systems that you want to use GDMA on, and want those systems to report all to the same GroundWork server, you may run into a situation where the same short host name is used for more than one host. GroundWork Monitor uses Nagios, which requires that host names be unique. For GDMA, this presents a problem, as the host name is automatically determined on the GDMA system, so it is possible to have two systems reporting status, and the GroundWork server will only represent them as one.

As long as the domain names differ, however, this can be accommodated by setting up the GDMA hosts in GroundWork Monitor using fully qualified names. Thus a host like server1.foo.com can be distinguished from `server1.bar.com`, and GroundWork Monitor will be able to tell them apart.

To enable this feature, use the parameter Use_Long_Hostname. Set this parameter to On in the host external, and change the host name in GroundWork Monitor Configuration to be the fully qualified name. This will change the way GDMA reports its results for services, and though the first run with the new settings will generate a single spooler messages under the old (short) hostname, subsequent cycles will have the long fully-qualified name used for poller and spooler messages as well.

Checking Multiple Hosts from a Single Linux GDMA Host

This is an advanced feature that is useful when checking clustered hosts, or simply setting up some lightweight monitoring of nearby hosts from a single Linux GDMA.

- The Linux GDMA must be installed using the command-line option `--multihost 1`
- You need to set up a configuration group for all hosts that will be checked in this way, with a specific build directory for the config files
- You need to create that build directory, and set permissions

Downloading New Plugins Automatically

GDMA (2.2.1 and later) with GroundWork Monitor (6.4 and later) allows you to download new plugins (or new versions of old plugins) automatically to specific versions of the GDMA. The following instructions illustrate how to enable this feature.

⚠ This feature is disabled by default.

Enable the Plugin Download Feature

1. Edit the file:

   ```
   /usr/local/groundwork/config/foundation.properties
   ```

   change:

   ```
   gdma.plugin.upload.enable=false
   ```

   to:

   ```
   gdma.plugin.upload.enable=true
   ```

   and save the file.

2. Edit the file:

   ```
   /usr/local/groundwork/apache2/conf/httpd.conf
   ```

   change:

   ```
   #Alias /plugin_download "/usr/local/groundwork/apache2/htdocs/agents/plugin_download"
   ```

   to:

   ```
   Alias /plugin_download "/usr/local/groundwork/apache2/htdocs/agents/plugin_download"
   ```

   and save the file.

3. Restart gwservices, run as `root`:

   ```
   /etc/init.d/groundwork restart gwservices
   ```

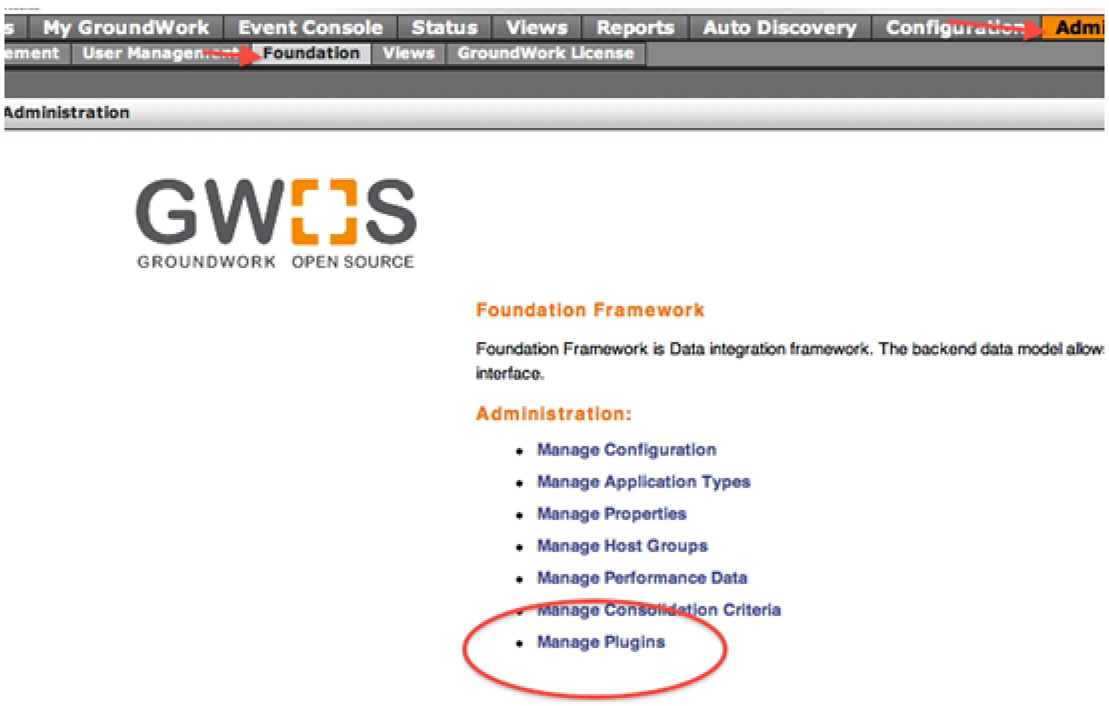
4. Restart apache, run as `root`:

   ```
   /etc/init.d/groundwork restart apache
   ```

   ⚠ This will log off all users.

Upload the New Plugin Files

1. Go to Administration>Foundation, then select the link Manage Plugins.



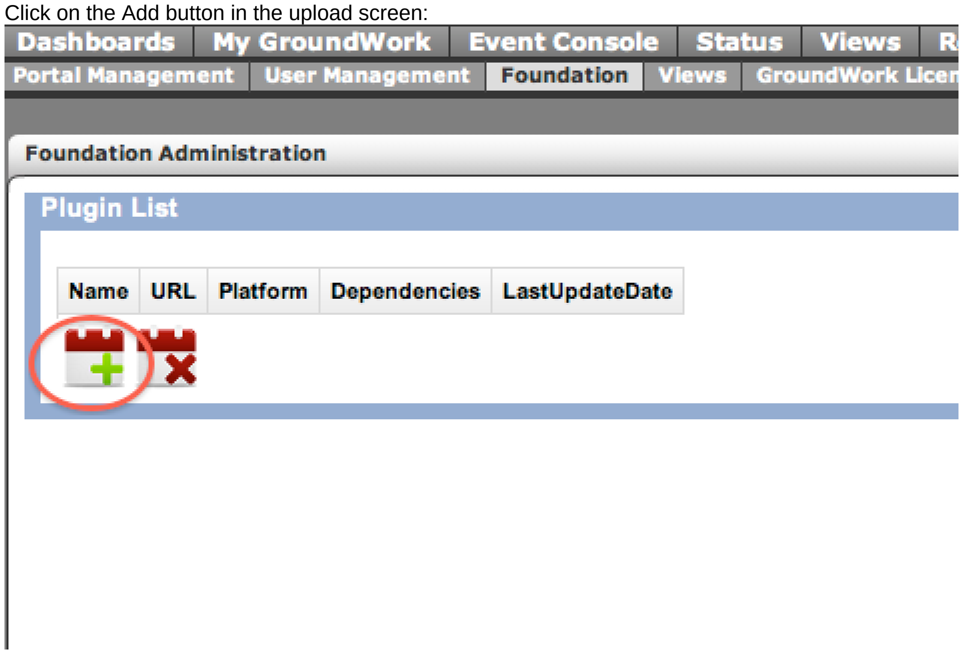
Foundation Framework

Foundation Framework is Data integration framework. The backend data model allow interface.

Administration:

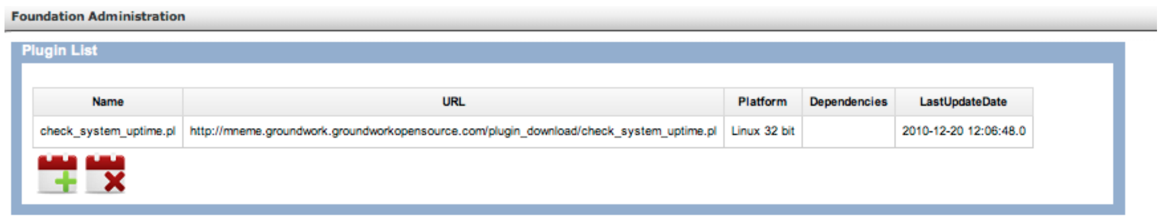
- Manage Configuration
- Manage Application Types
- Manage Properties
- Manage Host Groups
- Manage Performance Data
- Manage Consolidation Criteria
- Manage Plugins

2. Click on the Add button in the upload screen:



Name	URL	Platform	Dependencies	LastUpdateDate

3. Choose the platform and architecture that the plugin you are uploading will be used on from the drop-down. If your plugin will work on all the platforms on which you run *GDMA*, choose one of the Multiplatform options. If you want to run the plugin on both 32-bit and 64-bit platforms, you will need to upload it twice, once for Multiplatform 32 bit and once for Multiplatform 64 bit. Generally, however, you will have to upload a different version of a plugin for each platform you want to use it on.
   The names of the plugins for different operating systems or bit widths can be the same, though you will need to keep them in separate, well-defined places on your source machine to keep track of which copy is for which platform. The uploaded plugins will be stored in separate directories on the server, so there will be no confusion there. For instance, a plugin written as a shell script might work on just Linux but for both 32 bit and 64 bit versions of that OS, so you would upload it once for *Linux Intel 32 bit* and once for *Linux Intel 64 bit*.
4. Click Upload.
5. To check if your plugins are present, click the Home icon in the lower-left corner of the Add Plugin box. Or click Foundation in the menu, and then click the Manage Plugins link again. You should see a list of uploaded plugins:


Foundation Administration

Plugin List

Name	URL	Platform	Dependencies	LastUpdateDate
check_system_uptime.pl	http://mneme.groundwork.groundworkopensource.com/plugin_download/check_system_uptime.pl	Linux 32 bit		2010-12-20 12:06:48.0

6. In GroundWork Monitor 6.5, if you need to update a plugin (keeping the same name, but changing its content), you will need to delete and re-add it on each platform. The extra step involved in a delete/add as opposed to just re-uploading will be addressed in a future release.

Tell GDMA to Download the Plugins

1. Go to Configuration>Hosts, and expand the Host Externals section to modify a host external for the hosts that will be downloading plugins. Alternatively you can enable this for individual hosts by opening the host external for the specific host you want, but changing the host external for all like hosts will make it the downloads apply to many hosts at once. You may wish to try this on one host, and make sure you have it right before rolling out to all hosts of a given type.

   Only modify host externals for hosts running *GDMA* (2.2.1 or later). Older versions will error out if these parameters are placed in the host externals.

2. Add the following lines to the selected host external where <Server Name> is the resolvable name of the GroundWork server from the point of view of the GDMA:

```
Enable_Poller_Plugins_Upgrade = "On"
Poller_Plugins_Upgrade_URL="http://<Server
Name>/foundation-webapp/restwebservices/pluginUpdates/findUpdates"
```

   You may also use an https URL if you have configured *GroundWork Monitor* to use *HTTPS*, e.g:

```
Poller_Plugins_Upgrade_URL="https://<Server
Name>/foundation-webapp/restwebservices/pluginUpdates/findUpdates"
```

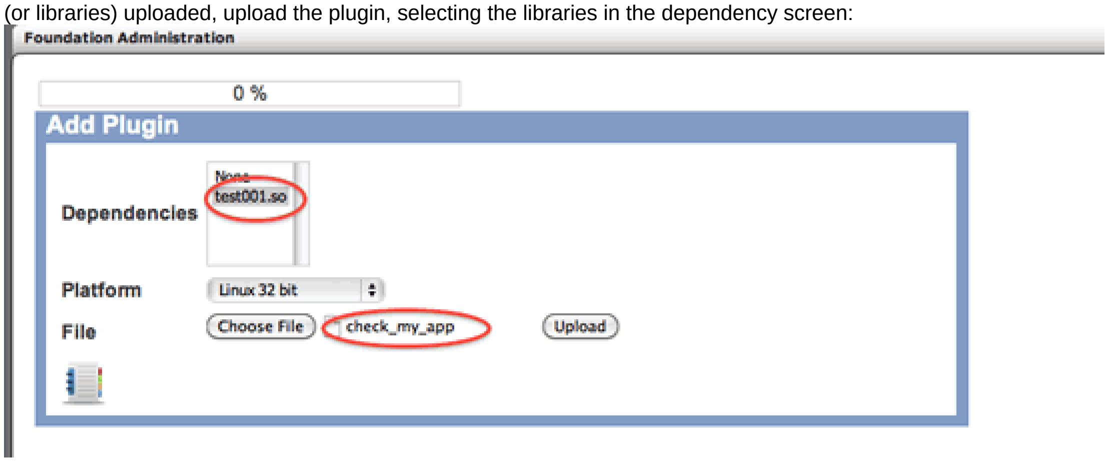
3. Save the host external and choose Replace existing externals or Merge with existing externals as appropriate.
4. Go to Configuration>Control, and select Build Externals.

Plugins will now be downloaded to the GDMA libexec directory, and will be flagged executable by the user running the GDMA process ("gdma" by default). Existing plugins with the same file name will be replaced.

Plugin Dependencies (UNIX versions)

You may want to use a plugin that depends on a library that GroundWork Monitor does not supply with the GDMA. For Windows systems, this is not usually the case, since though one could characterize a .dll file as a dependency, most Windows plugins are either vbscript (executed by `cscript.exe`), powershell (executed by `powershell.exe`) or Windows `.bat` batch files, executed by cmd.exe. If you need to download a .dll file to Windows, you should simply add it as you would a plugin, and it will be copied to the `libexec` subdirectory of the GDMA, where it will be available for the plugin.

If you are using the UNIX GDMA (Linux, Solaris or AIX), you may have access to the library you need. If you have this library (typically, one or more `.so` files) available, you can transfer it to the GDMA host into the `groundwork/common/lib` directory. Simply perform the above procedure, uploading the dependency files first, choosing platform and architecture as you would for the plugin. Then, when you have the library (or libraries) uploaded, upload the plugin, selecting the libraries in the dependency screen:



Dependencies: None, test001.so
Platform: Linux 32 bit
File: check_my_app

In this example, the `test001.so` library is needed by the new `check_my_app` plugin. Selecting the dependency when the plugin is loaded identifies test001.so as a dependency, which goes to the `groundwork/common/lib` directory, and `check_my_app` as a plugin, which gets placed in the `groundwork/nagios/libexec` directory.

If you update an existing plugin, the new version will simply overwrite the old. No special action will be taken to preserve the original plugin.

Plugins are downloaded only as needed. The system will check to see if the plugins are new, and have a different ND5 sum prior to downloading, so there is no downside to keeping a large number of plugins and dependencies on the GroundWork server.

You may remove a plugin or dependency (or several at once) at any time from the *Manage Plugins* screen, by selecting the files to be removed and clicking the *X* button.

Working with Powershell

The GDMA for Windows can be used to run Windows Powershell Commandlets, or small powershell programs that you write or modify to return results that can be interpreted as status and performance data by GroundWork Monitor. Especially in 64-bit environments, this is quite a powerful way to monitor your Windows systems.

GroundWork Monitor (6.4 and later) and GDMA (2.2.1 and later) come with some sample Powershell scripts that leverage commandlets to check some otherwise difficult-to-access data about 64-bit Windows systems. This section will take you through getting GDMA set up to run these examples.

Installing Powershell

Before you can run the Powershell plugins, you must have Powershell installed and working on your Windows system. Please see http://technet.microsoft.com/en-us/library/ee692944.aspx for information on getting Powershell running.

In particular you MUST enable Powesershell to run scripts. To do this, launch each version of the Powershell interpreter on your system (both the 64 bit Windows Powershell and the 32-bit Windows Powershell (x86)), and type:

```
Set-ExecutionPolicy RemoteSigned
```

Loading Example Profile

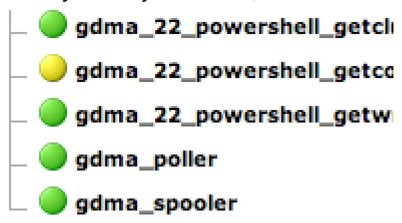
You will notice a new GDMA profile; the host profile gdma-22-windows-host.xml contains the service profile gdma-22-powershell, which in turn contains three services:

- **gdma_22_powershell_getcluster**: A plugin to get the status of a Microsoft cluster server, if it is installed
- **gdma_22_powershell_getcounter**: A generic plugin that gets any perf counter, but must be modified before it will be useful
- **gdma_22_powershell_getwmi**: An example of calling a WMI counter (CPU in this case) from a Powershell script

The modification of these example scripts in to more useful or specific uses is ongoing at GroundWork, and will doubtless be the source for many more profiles. For the moment, however, these are a starting point. To get these examples loaded:

1. Go to Configuration>Profiles.
2. Click on Profile Importer.
3. Select gdma-22-windows-host.xml from the list.
4. Select Import.

This will result in a host profile, which you can them apply to a host. Make this the host on which you have installed and enabled Powershell previously. Once you commit, and build it's externals these services will be active.



Of course, it's normal not to have OK status on an unconfigured service, so `gdma_22_powershell_getcounter` will be in warning until you modify the plugin and give it something to monitor. Also, unless you have Microsoft Cluster Server loaded on the host, you will not get logical output from `gdma_22_powershell_getcluster`.

> ⚠ The service externals are not generic, and may need to be modified to work with your system.

The service externals assume the default location of Powershell and GDMA. The example here explicitly calls `powershell.exe` in the default location, and passes it the full, no-spaces version of the path to the Powershell script to run. The quoting used is standard, and can be used in most cases. See 'Command='Line Tricks below.

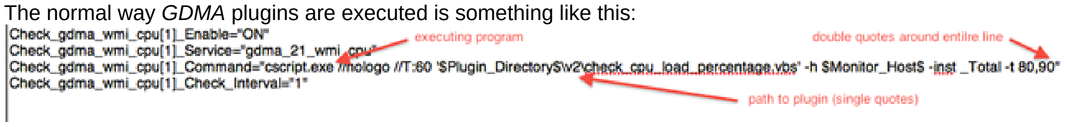
```
Check_gdma_powershell_getcounter[1]_Command="c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe
'C:\Progra~2\groundwork\gdma\libexec\v3\getCounter.ps1'"
```

If you instal GDMA in a different directory, you will need to modify this path. The short form Progra~2 expands out to Program Files (x86), which is where GDMA, a 32-bit program, is installed in 64-bit Windows by default.

'Command='line Tricks

In some cases you may want to execute a check command that does not fall under the usual formatting rules. In particular if you want to make your own plugins, or integrate a plugin that you download off of the Internet, you may need to adjust the format of the command line in the *Service External* you use to control the plugin execution.

The normal way *GDMA* plugins are executed is something like this:



```
Check_gdma_wmi_cpu[1]_Enable="ON"
Check_gdma_wmi_cpu[1]_Service="gdma_21_wmi_cpu"
Check_gdma_wmi_cpu[1]_Command="cscript.exe //nologo //T:60 '$Plugin_Directory$\v2\check_cpu_load_percentage.vbs' -h $Monitor_Host$ -inst _Total -t 80,90"
Check_gdma_wmi_cpu[1]_Check_Interval="1"
```

executing program
double quotes around entilre line
path to plugin (single quotes)

The entire command line is enclosed in double quotes. Within this enclosure, there are two main sections. The first is the executing program (`cscript.exe`, in this case, but in the Powershell example in the previous section, it was `powershell.exe`). Any arguments to the executing program come next. If you are using UNIX, the executing program is implicit: the command shell. In that case you can stop there, for example:

```
Check_gdma_linux_mem[1]_Command="check_mem.pl -F -w 20 -c 10"
```

In Windows, as well, you can run compiled programs as plugins in this way without any further specifications. Just be sure to include the enclosing double quotes.

The second main part is the interpreted plugin, with its full path, encased in single quotes. You can usually use the $Plugin_Directory$ macro, and this will be replaced with the location of the plugins in your GDMA installation. Note that you can have subdirectories off of this mail location, for example many of the vbscript plugins we supply are stored in the v2 subdirectory. Powershell plugins are stored in v3 by convention.

After the single-quoted full path to the interpreted plugin, and before the terminating double quote mark, you can supply any arguments to the interpreted plugin. The GDMA will understand bare arguments, arguments specified with a dash "-", or with a slash "/".

Troubleshooting

Common Issues

Time sync

Many installations do not properly synchronize time with NTP. This is particularly an issue with Virtual environments, where time slicing may introduce latency and drift to otherwise stable time signatures. For this reason, GDMA has been made tolerant of time drift. In the setup instructions, the file `/usr/local/groundwork/config/bronx.cfg` was modified to allow up to 900 seconds latency and 90 seconds imminence, or future timestamps, on incoming messages.

If the time synchronization is sufficiently off to be outside this 16.5 minute envelope, the GroundWork server will reject the incoming message. To correct this situation, you must either:

1. Synchronize the time on all monitored systems and the GroundWork server, or
2. Widen the envelope by changing the parameters in `bronx.cfg`.
   It is highly recommended that the time be synchronized.

> ⚠ The GroundWork Monitor version of `send_nsca` supplied with GDMA allows for the sending of old time stamps, which are the time stamps associated with when the check was run, not when the spooler has sent the data, which is potentially much later. This allows the system to accurately process performance data for checks that have been spooled, potentially for long periods.

Encryption Mismatch

A common security requirement that GroundWork Monitor supports is the encryption of data sent over the NSCA channel. By default, however, GDMA is installed with encryption turned off, which is also the default on the GroundWork server. Thus it is possible to have an encryption mismatch, for example, if you enable encryption on the GroundWork server (in `bronx.cfg`), and fail to enable it on the GDMA in `send_nsca.cfg`. Always make sure these files are consistent between GDMA (`send_nsca.cfg`) and GroundWork (`bronx.cfg`).

One way to manage this situation is to set up GDMA to use an alternate send_nsca.cfg file. This file can be downloaded to the GDMA as if it were a plugin, and specified for use via the host external using the GDMA parameter called Spooler_NSCA_Config. For example:
If you create a file called:

```
send_nsca_gdma_encrypted.cfg
```

and you set it up for download as a plugin for Linux, it will be placed on the Linux GDMA in:

```
/usr/local/groundwork/gdma/tmp/send_nsca_gdma_encrypted.cfg
```

That means you could set up the GDMA to start using this file at the same time you switch the GroundWork server to use encryption, just by changing:

```
Spooler_NSCA_Config="/usr/local/groundwork/gdma/tmp/send_nsca_gdma_encrypted.cfg"
```

Generating Logs

In general, GDMA will not output much in the way of log information unless you specify that it should do so. If you want logs, you can use the Enable_Local_Logging parameter to output log data to the file specified in the Logdir parameter, which is the log subdirectory by default. You should not leave this enabled permanently, as there is no facility to remove old logs.

Running by Hand

You can also generate useful output (the same as what gets placed in the log) by running the GDMA Poller and/or Spooler by hand.

In Windows, open a command shell, and type:

```
cd \Program Files\groundwork\gdma\bin
gdma_poll.exe -i -d2 -x
```

As the Windows GDMA can be installed in alternate locations, you may need to enter a different path. For example in 64-bit Windows, GDMA is installed in \Program Files (x86)\groundwork\gdma by default.

This will give you output similar to the following:

```
C:\Users\Administrator>cd "\Program Files (x86)\groundwork\gdma"\bin

C:\Program Files (x86)\groundwork\gdma\bin>gdma_poll.exe -i -d2 -x
Getting [http://gdma-autohost/gdma/gwmon_wingdma64-qa-2.cfg] to C:\Program Files
(x86)\groundwork\gdma\config\gwmon_wingdma64-qa-2.cfg(timeout 10)
Attempting to get [http://gdma-autohost/gdma/gwmon_wingdma64-qa-2.cfg] to C:\Program Files
(x86)\groundwork\gdma\config\gwmon_wingdma64-qa-2.cfg (timeout is set to  10)
get_gdma_cfg_file_using_http: config file modified

Retrieved content for [http://gdma-autohost/gdma/gwmon_wingdma64-qa-2.cfg]

Checking autoconfig file for a change.
Reloading config file
Sucessfully read host config file
Host config file syntax is ok.
Spooling poller startup message
spool_startup_info: Poller 2.2.2 started at Wednesday January 12 11:34:25 2011
Pushed result into the buffer: 0   0   1294860865   wingdma64-qa-2   gdma_poller    0
&nbsp;&nbsp;Poller 2.2.2   started at Wednesday January 12 11:34:25 2011
Executing poller normal mode.
ConfigFile_Pull_Cycle = 1
ConfigPull_Timeout = 10
Enable_Auto = On
Enable_Local_Logging = on
GDMAConfigDir = gdma
GDMA_Auto_Host = gdma-autohost
GDMA_Auto_Service = gdma_auto
GDMA_Multihost = off
HostSequenceNumber = 10
Logdir = C:\Program Files (x86)\groundwork\gdma\log\
Max_Concurrent_Hosts = 1
NumHostsInInstallation = 11
Poller_Plugin_Directory = C:\Program Files (x86)\groundwork\gdma\libexec\
Poller_Plugin_Timeout = 45
Poller_Proc_Interval = 600
Poller_Pull_Failure_Interval = 86400
Poller_Service = gdma_poller
Poller_Status = Off
Spooler_Batch_Size = 20
Spooler_Max_Retries = 10
Spooler_NSCA_Config = C:\Program Files (x86)\groundwork\gdma\config\send_nsca.cfg
Spooler_NSCA_Port = 5667
Spooler_NSCA_Program = C:\Program Files (x86)\groundwork\gdma\bin\send_nsca.exe
Spooler_NSCA_Timeout = 5
Spooler_Proc_Interval = 30
Spooler_Retention_Time = 900
Spooler_Service = gdma_spooler
Spooler_Status = Updates
Target_Server = [http://gdma-autohost]
wingdma64-qa-2_Check_gdma_powershell_getcluster[1]_Command  =
c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe
C:\Progra~2\groundwork\gdma\libexec\v3\getCluster.ps1
...
```

There are several diagnostic steps that GDMA goes through each time it runs. Errors in the retrieval of the config file, or with a syntax error in that file, will be presented near the top of this output. Any problems running the checks will appear in the last sections.

If all of that is working ok, then you might try running the spooler:

```

C:\Program Files (x86)\groundwork\gdma\bin>gdma_spool_processor.exe -i -d2 -x
Checking config [file:C:\Program] Files (x86)\groundwork\gdma\config\gwmon_wingdma64-qa-2.cfg for a
change.
Checking config [file:C:\Program] Files (x86)\groundwork\gdma\config\gdma_auto.conf for a change.
Reloading config files.
Sucessfully read host config file
Host config file syntax is ok.
The configuration file contains:
```

```
ConfigFile_Pull_Cycle = 1
ConfigPull_Timeout = 10
Enable_Auto = off
Enable_Local_Logging = on
GDMAConfigDir = gdma
GDMA_Auto_Host = gdma-autohost
GDMA_Auto_Service = gdma_auto
GDMA_Multihost = off
HostSequenceNumber = 10
Logdir = C:\Program Files (x86)\groundwork\gdma\log\
Max_Concurrent_Hosts = 1
NumHostsInInstallation = 11
Poller_Plugin_Directory = C:\Program Files (x86)\groundwork\gdma\libexec\
Poller_Plugin_Timeout = 45
Poller_Proc_Interval = 600
Poller_Pull_Failure_Interval = 86400
Poller_Service = gdma_poller
Poller_Status = Off
Spooler_Batch_Size = 20
Spooler_Max_Retries = 10
Spooler_NSCA_Config = C:\Program Files (x86)\groundwork\gdma\config\send_nsca.cfg
Spooler_NSCA_Port = 5667
Spooler_NSCA_Program = C:\Program Files (x86)\groundwork\gdma\bin\send_nsca.exe
Spooler_NSCA_Timeout = 5
Spooler_Proc_Interval = 30
Spooler_Retention_Time = 900
Spooler_Service = gdma_spooler
Spooler_Status = Updates
Target_Server = [http://gdma-autohost]
wingdma64-qa-2_Check_gdma_powershell_getcluster[1]_Command  =
c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe
C:\Progra~2\groundwork\gdma\libexec\v3\getCluster.ps1
wingdma64-qa-2_Check_gdma_powershell_getcluster[1]_Enable = ON
wingdma64-qa-2_Check_gdma_powershell_getcluster[1]_Check_Interval = 1
wingdma64-qa-2_Check_gdma_powershell_getcluster[1]_Service = gdma_22_powershell_getcluster
wingdma64-qa-2_Check_gdma_powershell_getcounter[1]_Command  =
c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe
C:\Progra~2\groundwork\gdma\libexec\v3\getCounter.ps1
wingdma64-qa-2_Check_gdma_powershell_getcounter[1]_Enable = ON
wingdma64-qa-2_Check_gdma_powershell_getcounter[1]_Check_Interval = 1
wingdma64-qa-2_Check_gdma_powershell_getcounter[1]_Service = gdma_22_powershell_getcounter
wingdma64-qa-2_Check_gdma_powershell_getwmi[1]_Command  =
c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe
C:\Progra~2\groundwork\gdma\libexec\v3\getWMI.ps1
wingdma64-qa-2_Check_gdma_powershell_getwmi[1]_Enable = ON
wingdma64-qa-2_Check_gdma_powershell_getwmi[1]_Check_Interval = 1
wingdma64-qa-2_Check_gdma_powershell_getwmi[1]_Service = gdma_22_powershell_getwmi
Spooling spool processor startup message
spool_startup_info: Spool processor 2.2.2 started at Wednesday January 12 11:38:52 2011
Pushed result into the buffer: 0   0   1294861132   wingdma64-qa-2   gdma_spooler   0   Spool
processor 2.2.2 started at Wednesday January 12 11:38:
52 2011
Spool processor running in normal mode.
Processing live targets.
Processing for target gdma-autohost
Spooling heart beat message
Spooler heart beat message: Spooler transmitted 2 results in 0.308 secs |
RESULTS=2;;;;TIME=0.308sec;;;;
Pushed result into the buffer: 0   0   1294861132   wingdma64-qa-2   gdma_spooler   0   Spooler
```

```
transmitted 2 results in 0.308 secs | RESULTS=2;;;;T
IME=0.308sec;;;;
Loop count=0. Last loop exec time = 0.313 seconds.
```

The spooler merely takes output from the checks that have been run and sends it to the Target server or servers. Of interest here is if it succeeded in contacting the Target, which will be indicated as Processing live targets. Dead targets are Target servers that can't be reached for some reason, and indicate a down primary or standby system, or network issues in contacting the Target. The spooler will keep trying until it reaches a dead target, or times out according to the Spooler_Retention_Time and Spooler_Max_Retries parameters.

For UNIX, the command is similar, but you must run as the gdma user, with the full execution environment of that user, and specify the GroundWork-supplied perl interpreter.

On Linux and AIX

```
su - gdma
cd /usr/local/groundwork/gdma/bin
/usr/local/groundwork/perl/bin/perl gdma_poll.pl -i -d2 -x
```

On Solaris

```
su - gdma
cd /opt/groundwork/gdma/bin
/opt/groundwork/perl/bin/perl gdma_poll.pl -i -d2 -x
```

The output is essentially the same for UNIX as for Windows.