

GDMA Advanced

Contents

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.

1.0 Detailed Installation and Setup

1.1 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.

1.2 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 `gdma_auto.conf` file on the GDMA client, 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.

1.3 Forcing determination of GDMA client hostname

The `Forced_Hostname` directive is used to force determination of the GDMA client hostname to a fixed value. This option is fully supported in the latest GDMA release, on all platforms. `Forced_Hostname` is an optional directive in the GDMA client config files. The value is the exact hostname (unqualified or fully-qualified, as specified) to be used in place of any dynamic determination of the hostname. GDMA will lowercase the supplied value before use, to correspond to the uniform use of lowercase hostnames in the rest of GDMA.

Additionally, this option is used in support of GDMA auto-registration so that when a GDMA client successfully registers itself with the GroundWork server, the hostname that the server determined was correct for this client gets returned to the client. The client then stashes that hostname as the value of the `Forced_Hostname` option in the new `gdma/config/gdma_override.conf` file, so both the poller (which received the hostname from the server) and the spooler (which passively accepts this instruction) know the proper hostname to use even if those components are bounced. This way, the base `gdma_auto.conf` configuration file is never modified by the auto-registration processing. When the `Forced_Hostname` is used in this manner for auto-registration, any value, which may have been manually set in the `gdma/config/gdma_auto.conf` file, will be ignored in favor of the value in the `gdma/config/gdma_override.conf` file.

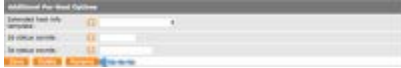
Also note that the GDMA auto-registration feature provides special support on the server for forcing the assignment of hostnames to particular values for certain client hosts, via the `<hardcoded_hostnames>` section of the `config/register_agent.properties` file.



KNOWN ISSUE - Renaming a GDMA Host

- **Known Issue:**

Using the Rename button, located at the bottom of the host detail screen (Configuration > Hosts > Hosts > {hostgroup} > {host} > Detail), to rename a GDMA host will cause the host to stop receiving checks. This action fails to overwrite the agent configuration on the remote machine and restart the remote agent to allow it to pull the new configuration. Once the workaround below has been completed, a new configuration file will present itself with the corrected name, and the forced config file is also updated with the correct name.



- **Workaround:**

This must happen on the GDMA client only after the next Commit after the server-side Rename occurs. That way, the new setup will be in place on the server when the updated client reaches for it. Also note that, after the Commit, the administrator should re-build externals, so make sure they are all up-to-date, before making changes on the GDMA client.

- Stop GDMA service
- Delete the forced config and the host named config files
- Start GDMA service

- **GDMA client files to delete:**

- `gdma/config/gdma_override.conf` (which file is supplied to the GDMA client by the GroundWork server during auto-registration, and holds a `Forced_Hostname` directive specifying the form and content of the hostname that the client is to use when interacting with the server)
- `gdma/config/gwmon_hostname.cfg` (which file, named using the precise form of the client hostname which the client should be using, is supplied to the GDMA client by the GroundWork server, and contains host and service externals from the server which are relevant to this client, note this is the actual hostname of the GDMA client, not the literal characters "hostname")

1.4 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

2.0 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.

2.1 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.

- Restart Foundation, run as root:

```
service groundwork restart gwservices
```

- Restart Apache, run as root:

```
service groundwork restart apache
```



This will log off all users.

2.2 Upload new plugin files

- Go to *GroundWork Administration > Foundation*, and select the link *Manage Plugins*.



- Click the add icon from the Plugin List screen.



- Choose the platform and architecture the plugin being uploaded will be used 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*. Click Upload.



- To view your current plugins, 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.



- In GroundWork Monitor (6.5 and later), 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.

2.3 Tell GDMA to download the plugins

1. Go to *Configuration > Hosts, Host Externals > Modify* and select 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.

- 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"
```



Important Note

When using *HTTPS*, the <Server Name> must exactly match what is in the server's SSL certificate (typically, a fully-qualified name). For more information regarding SSL support see the Bookshelf document [How to enable SSL support](#).

- Save the host external and choose *Replace existing externals* or *Merge with existing externals* as appropriate.



2. And build externals, *Configuration > Control > Build Externals*. Plugins will 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.

2.4 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.

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.



3.0 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.

3.1 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 PowerShell 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:

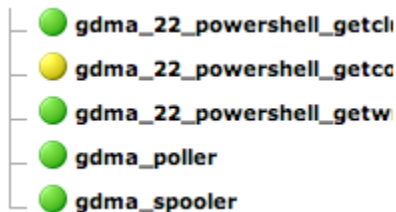
```
Set-ExecutionPolicy RemoteSigned
```


3.2 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 > Profile Importer*.
2. Select *gdma-22-windows-host.xml* from the list.
3. Click *Import*. This will result in a host profile, which you can then apply to a host. Make this the host on which you have installed and enabled PowerShell previously. Once you commit, and build externals these services will be active. Of course, it's normal not to have OK status on an un-configured 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.

 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.

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

3.3 '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 $Plugin_Directory$w2\check_cpu_load_percentage.vbs -h $Monitor_Host$ -inst _Total -t 80,90"  
Check_gdma_wmi_cpu[1]_Check_interval="1"
```



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 "/".

4.0 Process for Macro Substitutions

GDMA macro substitution is done in two places:

1. On the server, when externals are built
2. On the client, when checks are executed

Here are all the variables that are substituted into the `host-config` file content, both when externals are built on the server, and when the contents of that file are interpreted on the client.

1. When externals are built on the server:
 - a. Nagios resource macros are substituted first. These are `$USERn$` references, as defined under *Configuration > Control > Nagios resource macros*.
 - b. Next, Monarch group macros are substituted, as defined under *Configuration > Groups > Macros* (to first establish the existence of a given macro name) and then under *Configuration > Groups > Groups > <group name> > Detail > Macros*.

A given host can be a member of one or more Monarch groups. This membership can be either via assignment of the host to a specific Monarch group, or indirect via assignment of the host to a hostgroup, and assignment of the hostgroup to a Monarch group.

Typically, when a host is a member of more than one Monarch group, these Monarch groups are arranged in some kind of parent-group/sub-group tree (which can be quite complex). If the host is a member of multiple Monarch groups like that, then when the externals file for the top-parent Monarch group is created, group-macro substitutions are made in order from sub-groups to parent groups, up the chain of ancestry. This allows the sub-group macro definitions to override parent-group macro definitions, which is an important use case for having sub-groups.

When externals are built for a given host, they are really only built for each top-level Monarch group associated with that host; you don't get an extra externals file for each sub-group to which the host belongs.

- c. Finally, a few fixed-name macros are substituted, in order: `$HOSTNAME$`, `$HOSTADDRESS$`, and `$HOSTALIAS$`. These values are drawn from the Monarch configuration for this host.
2. When externals are interpreted on the client:
 - a. `$Plugin_Directory$` (drawn from the client's configured `Poller_Plugin_Directory`) and `$Monitor_Host$` (the name of the host for which the check is being run) are substituted first into each plugin execution string.
 - b. Option parameters are substituted into the plugin execution string, in essentially arbitrary order (you can't depend on a specific ordering of these substitutions).

This last category, of option parameters, seems not to be documented anywhere, rather to our chagrin. We'll get this corrected; this capability

ought to be listed in the GDMA Configuration Reference page, under Per Service Configuration Parameters, along with some realistic examples of its use. In the meantime, here are the details, drawn from reverse-engineering the code.

In addition to the documented parameters:

```
Check_{service}_Enable
Check_{service}_Service
Check_{service}_Command
Check_{service}_Timeout
Check_{service}_Check_Interval
```

which are used in service externals like these lines:

```
Check_gdma_linux_swap[1]_Enable="ON"
Check_gdma_linux_swap[1]_Service="linux_swap"
Check_gdma_linux_swap[1]_Command="check_swap -w 10% -c 5%"
Check_gdma_linux_swap[1]_Check_Interval="1"
```

you can also define any number of "Parm" parameters for a given service, in any of several forms:

```
Check_Disk[1]_Parm_--warning = "10%"
```

The value of a double-quoted parameter value is set to the string that is enclosed in quotes.

```
Check_Response_Servlet[1]_Parm_-n = Monitor_Server[1]
```

```
Check_This_Servlet[1]_Parm_-n = Check_Other_Servlet[1]_Parm_-n
```

If there are no quotes after the "=" character, the value of this parameter can be set to the value of another parameter that has already been defined. (Cross-references like this can have considerable complexity, which we are not describing in detail here.)

```
Check_Response_Servlet[1]_Parm_-n = Monitor_Server[1],Monitor_Server[2]
```

In the case just described, comma-separated multiples like this are supported.

```
Check_Response_Servlet[1]_Parm_-n = gw.company.com
```

If there are no quotes around the value, and that value does not represent some other parameter which is already defined, the value will be taken as-is.

```
Check_Response_Servlet[1]_Parm_-n = gw1.company.com,gw2.company.com
```

In the case just described, comma-separated multiples like this are supported.

```
Check_Disk[1]_Parm_--errors-only
```

If there is no "=" character in the parameter line, the parameter value is set to an empty string.

These parameters will be substituted into the plugin execution string as follows:

1. A parameter with a name like "Parm_--ABC" will be appended to the command line as "--ABC=\$value", where \$value is the value of that parameter as listed above. Note that in the present GDMA code, \$value will not itself be surrounded by additional quotes when it is used in the command line this way, which can be problematic (any spaces or shell metacharacters in the \$value will not be protected from interpretation by the shell). If you need such protection, the \$value itself should contain single-quote characters to provide such protection:

```
Check_Load[1]_Parm_--message = "A protected string."
```

2. A parameter with a name like "Parm_-DEF" will be appended to the command line as "-DEF \$value", where \$value is the value of that parameter as listed above. The same issue with quoting applies here.

It looks like there is some inconsistency in the way the present code works. You would probably want this specification:

```
Check_Disk[1]_Parm_--errors-only
```

to be substituted into the plugin execution string by having the string `--errors-only` be appended to the command line. But it looks like the present code will instead append `--errors-only=`; this can be considered a bug. There are probably some other boundary cases that are also not handled as well as they should be. We will need to clean up these things in a future GDMA release.

5.0 Configuration Reference

5.1 GDMA Configuration


This section describes the configuration details and installation methods for GroundWork Distributed Monitoring Agent (GDMA).




In normal operation, a GDMA host configuration is generated via the configuration tool Monarch using the Externals capability. Global (per-host) configuration parameters are set via a host external property and per-service parameters are set via a service external property, see [Configuring Externals](#).



GDMA uses two configuration files to control its operation. These files share the same structure and command parameters.

- **Autoconfiguration Config** - The file `$basedir/config/gdma_auto.conf` is a component of the installation image and governs the behavior of the newly installed agent in an unconfigured state. The purpose of this file is to enable the agent to contact the master server and obtain a host configuration file without operator intervention. In general, this file should not be modified unless special circumstances require non-default autoconfiguration behavior. This file is also used to enable the agent to revert to autoconfiguration in the event that the host configuration file becomes corrupted or if the master server becomes unavailable.
- **Host Config** - The file `$basedir/config/gwmon_$host.cfg` contains the normal-mode configuration for the agent. This file overrides all parameters from the autoconfiguration file when it exists. This file will contain the global (per-host) configuration and per service configuration details for the agent and the services to be monitored.

Global Configuration Parameters


Parameter Name	Default Value (Valid Values)	Used by Poller/Spooler	Description
Auto_Register_Attempts	<code>fibonacci</code> (never, once, arithmetic, exponential, fibonacci, or periodic)	Poller	<div style="background-color: #ffffcc; padding: 10px; border: 1px solid #ccc;">  This option should not be set in host externals; it has no business being managed on the server, it controls behavior of the GDM client before the host external file is downloaded from the server. So this option should appear only in the client's own <code>gdma_auto.conf</code> file. </div> <p>Specifies what sort of algorithm should be used to decide whether repeated auto-registration should be made, once an initial attempt has failed. This parameter is available starting with G 2.3.1.</p> <p>The <code>arithmetic</code>, <code>exponential</code>, and <code>fibonacci</code> settings specify increasing delays between successive attempts to auto-register. In each of these cases, the increasing delays are capped by the <code>Auto_Register_Max_Increase</code> setting. The standard default value for <code>Auto_Register_Attempts</code> is <code>fibonacci</code>, which invokes several quick retries and then backs off until the full <code>Auto_Register_Max_Interval</code> period is reached. This allows rapid response when the server responds quickly, and eventual automatic recovery if the server doesn't respond for a long time, and low latency requests that will never be successfully answered.</p> <ul style="list-style-type: none"> • <code>never</code>: Make no attempts at all, not even an initial attempt. • <code>once</code>: Make just one attempt, with no follow-ups. • <code>arithmetic</code>: Make further attempts after the following full-cycle delays between cycles where auto-registration is tried again: 1, 2, 3, 4, 5, 6, etc. • <code>exponential</code>: Make further attempts after the following full-cycle delays between cycles where auto-registration is tried again: 1, 2, 4, 8, 16, 32, etc. • <code>fibonacci</code>: Make further attempts after the following full-cycle delays between cycles where auto-registration is tried again: 1, 1, 2, 3, 5, 8, etc., according to the standard Fibonacci sequence. • <code>periodic</code>: Make further attempts to auto-register if auto-registration is tried again, with subsequent attempts spaced by the <code>Auto_Register_Max_Interval</code> setting.

Auto_Register_Host_Profile	empty string (The name of a host profile on the GroundWork server, or an empty string. The usual value is a per-platform host profile, namely one of <code>gdma-aix-host</code> , <code>gdma-linux-host</code> , <code>gdma-solaris-host</code> , or <code>gdma-windows-host</code> . But you may specify another host profile name, if you wish.)	Poller	<div data-bbox="1138 159 1495 436" style="background-color: #ffffcc; padding: 5px;">  This option should not be set in host externals; it has no business being managed on the server, it controls behavior of the GDMA client before the host external: file is downloaded from the server. So this option should appear only in the client's own <code>gdma_auto.conf</code> file. </div> <p>The host profile to be applied to this machine monitoring setup during auto-registration. This value is optional; if missing or left blank, the profile will be defaulted on the server. This parameter is available starting with GDMA</p>
Auto_Register_Max_Interval	86400 (1-1000000000 seconds)	Poller	<div data-bbox="1138 646 1495 924" style="background-color: #ffffcc; padding: 5px;">  This option should not be set in host externals; it has no business being managed on the server, it controls behavior of the GDMA client before the host external: file is downloaded from the server. So this option should appear only in the client's own <code>gdma_auto.conf</code> file. </div> <p>Defines the maximum period (specified in seconds) to wait between successive auto-registration attempts. This parameter is available starting with GDMA 2.3.1.</p>
Auto_Register_Pass	no default (the password corresponding to <code>Auto_Register_User</code> , or an empty string)	Poller	<div data-bbox="1138 1108 1495 1386" style="background-color: #ffffcc; padding: 5px;">  This option should not be set in host externals; it has no business being managed on the server, it controls behavior of the GDMA client before the host external: file is downloaded from the server. So this option should appear only in the client's own <code>gdma_auto.conf</code> file. </div> <p>Defines the password used for auto-registration attempts. If this value is missing or empty, auto-registration will be disabled, and the client will fall back to using the older auto-configuration protocol. This parameter is available starting with GDMA 2.3.0.</p>

Auto_Register_Service_Profile	empty string (The name of a service profile on the GroundWork server, or an empty string.)	Poller	 This option should not be set i host externals; it has no busin being managed on the server, it controls behavior of the GDM client before the host external: file is downloaded from the server. So this option should appear only in the client's own gdma_auto.conf file. <p>The service profile to be applied to this ma monitoring setup during auto-registration. value is optional; if missing or left blank, n service profile (beyond any that are includ host profile) will be applied. This paramete available starting with GDMA 2.3.0.</p>
Auto_Register_User	no default (a GroundWork Monitor username reserved for this purpose, or an empty string)	Poller	 This option should not be set i host externals; it has no busin being managed on the server, it controls behavior of the GDM client before the host external: file is downloaded from the server. So this option should appear only in the client's own gdma_auto.conf file. <p>Defines the username used for auto-regist attempts. If this value is missing or empty, auto-registration will be disabled, and the c fall back to using the older auto-configurat protocol. This parameter is available startii GDMA 2.3.0.</p>
ConfigFile_Pull_Cycle	1 (1-10)	Poller	Controls how often new configuration infor should be pulled from the Master server. T is a multiple of Poller_Proc_Interval.
ConfigPull_Timeout	10 (1-100)	Poller	Controls how long in seconds to wait before out a configuration file pull attempt.
Enable_Auto	on (on, off)	Poller, Spooler	Enables and disables auto configuration r This should normally be toggled on in gdma_auto.conf and off in gwmon_\$host.c
Enable_Local_Logging	off (on:off)	Poller, Spooler	Enables local logging of events and errors is off. If this is enabled, some provision to rotate logs should be employed.
GDMA_Auto_Host	gdma-autohost (string conforming to GroundWork/Nagios host object rules)	Spooler	Host, defined in GroundWork Monitor Configuration, for which GDMA will submit configuration messages to the Target_Ser the status of the gdma_auto service.
GDMA_Auto_Service	gdma_auto (string conforming to GroundWork/Nagios service object rules)	Spooler	GroundWork service under which GDMA v submit autoconfiguration messages.

GDMAConfigDir	gdma (valid path string)	Poller	Defines the path where the agent will attempt to pull a valid gwmon_\$host.cfg file from. This is concatenated with Target_Server and the hostname of the target system to identify the configuration file to pull. Example: gdma_auto.conf: Target_Server="http://gdma-autohost" gdma_auto.conf: GDMAConfigDir="gdma" Target hostname="gdma-test" Would result in the agent pulling its configuration file from "http://gdma-autohost/gdma/gwmon_gdma
GDMA_Multihost	off (on, off)	Poller	Enable Multihost (windows child) behavior
Logdir	/usr/local/groundwork/gdma/log/ or c:\program files\groundwork\gdma\log (absolute directory path)	Poller, Spooler	Path to write local log files. Must be valid and writable by agent user.
Max_Server_Redirects	5 (unsigned integer)	Poller	Specifies the maximum number of HTTP(S) redirects the client will tolerate when fetching from the server and sending automated agent registration requests. A non-zero value here allows redirection when transitioning an infrastructure from use of HTTP to use of HTTPS for instance. The Max_Server_Redirects can be set to 0 to completely disallow redirection, the highest level of security. Because many of the uses of this parameter involve actions that precede having an external file from the server in hand, it really only makes sense to set this parameter in the client's config file, not in host external files. This option is available starting with the GroundWork 2.3.2 release.
Poller_Service	gdma_poller (string conforming to GroundWork/Nagios service object rules)	Poller	GroundWork service under which GDMA will submit poller messages.
Spooler_Service	gdma_spooler (string conforming to GroundWork/Nagios service object rules)	Spooler	GroundWork service under which GDMA will submit spooler messages.
Poller_Status	off (on, off)	Poller	On: Send poller heartbeat status message containing poller statistical data. Off: Do not send any poller heartbeat status messages.
Spooler_Status	on (on, off, Updates)	Spooler	On: Send all spooler status messages. Off: Do not send any spool processing updates. Updates: Only send when there are other updates being submitted (e.g. Do not send status messages when the only update is a spooler status message.)
Poller_Proc_Interval	600 (10-3600 seconds)	Poller	Controls how often the poller engine will run. Used as a base interval by ConfigFile_Pull and Check_(service)_Check_Interval. Note that this value is not recommended without understanding the full impact on performance, collection and target system overhead for your environment.

Poller_Pull_Failure_Interval	86400 (1 Day) (0- 2592000 seconds (0-30Days))	Poller	Controls how long the poller will continue to operate in normal mode with the current host configuration after it can no longer pull a configuration file from the master server. When the timer expires, the poller engine will remove the current host configuration file and revert to autoconfiguration mode. Setting this value to 0 disables this feature and the agent will continue to operate in normal mode with the stale configuration file indefinitely.
Poller_Plugin_Timeout	5 (0- 900 seconds)	Poller	Controls how long the poller will wait for a plugin to execute, if a per-service Check_(service)_plugin_timeout value is not specified.
Poller_Plugin_Directory	/usr/local/groundwork/gdma/libexec/ or c:\program files\groundwork\gdma\libexec (absolute directory path)	Poller	Path to local plugins
Spooler_Batch_Size	20 (integer >= 5)	Spooler	Defines the max NSCA batch size to use when transmitting results to each master server.
Spooler_Max_Retries	10 (1 <= integer <= 100)	Spooler	Attempt transmission of results this many times before being purged. Multiple of Spooler_Proc_Interval time.
Spooler_NSCA_Config	/usr/local/groundwork/gdma/config/send_nsca.cfg or c:\Program Files (x86)\groundwork\gdma\config\send_nsca.cfg	Spooler	This is the absolute path to the configuration file send_nsca.
Spooler_NSCA_Port	5667 (valid tcp port number)	Spooler	Port to use when transmitting spooled results. This port number must match the listener_port value on the master server (on each individual Target_Server machine).
Spooler_NSCA_Program	/usr/local/groundwork/common/bin/send_nsca or c:\Program Files\groundwork\gdma\bin\send_nsca.exe	Spooler	Location of send nsca program
Spooler_NSCA_Timeout	10 (1-30 seconds)	Spooler	NSCA send timeout
Spooler_Proc_Interval	180 (60-900 seconds)	Spooler	Controls how often the spooler engine will be used as a base interval by Spooler_Max_Retries.
Spooler_Retention_Time	900 (1 <= integer <= 900)	Spooler	Unsent spooled results older than this will be purged.
Target_Server	[http://gdma-autohost] (a valid http or https url)	Poller, Spooler	Defines the address(es) to pull configuration files from and send all check results to. This may be the form of a comma separated list to send to multiple targets. If a list is specified, the first element in the list is the one from which configuration files will be pulled.
Target_Server_Secondary	no default (a valid http or https url)	Spooler	Defines the system where the agent will attempt to send check results when the Target_Server is unavailable.
Forced_Hostname	The exact hostname (unqualified or fully-qualified, as specified) to be used in place of any dynamic determination of the hostname.	Poller, Spooler	This is a directive used to force determine the GDMA client hostname to a fixed value. It is fully supported in the GDMA 2.3.0 release on all platforms. See GDMA Advanced for detail.

<p>Use_Long_Hostname</p>	<p>undefined (on, off, or undefined (not specified)) (Note: GDMA 2.3.0 inappropriately set this option to "on" in the freshly installed <code>gdma_auto.conf</code> file; that setting is not appropriate for many customers. Users may wish to install GDMA 2.3.1 instead, which does not define this option by default, or comment out the value in the config file.)</p>	<p>Poller, Spooler</p>	<p>Controls whether the poller and spooler use fully-qualified hostname for the <code>gwmon_\$host.cfg</code> filename, and the host under which GDMA reports its results.</p> <p>"on" means that the GDMA client will attempt to use fully-qualified hostname when retrieving its externals file from the GroundWork Monitor and when reporting check results.</p> <p>"off" means that the GDMA client will attempt to use unqualified hostname in those circumstances.</p> <p>If this option is not set at all, the GDMA client will attempt to use a fully-qualified hostname when retrieving its externals file back to using an unqualified hostname if that succeeds. Whichever form succeeds will be used for check results.</p> <p>A site which wants to force the use of fully-qualified hostnames should set this option "on" in the <code>gdma_auto.conf</code> file for those hosts. Setting this parameter in host externals is too late and should be avoided, because this parameter is used to determine the name of the downloaded configuration file. See GDMA Advanced for details. If this parameter is left undefined, GDMA will first try to use a long hostname (FQDN), and if that fails, it will try to use an (unqualified) hostname. This provides a graceful automatic failover mechanism.</p> <div data-bbox="1138 1016 1502 1780" style="background-color: #ffffcc; padding: 10px; border: 1px solid #ccc;"> <p> The setting of this option will be effectively ignored if the <code>Forced_Hostname</code> option is in play. (<code>Forced_Hostname</code> is to be used sparingly in manual configurations, only in exceptional conditions when the local configuration simply cannot be set up to correctly determine the hostname by ordinary automatic means.) In particular, successful auto-registration creates an extra <code>gdma_override.conf</code> file that contains a <code>Forced_Hostname</code> directive specifying the GDMA client hostname as determined by the GroundWork Monitor server. The auto-registration code on the server has its own controls over the format of the hostnames it returns to clients and those controls take precedence over the <code>Use_Long_Hostname</code> option on the client when the server determines the desired client hostname. See [DOCDEV:GD FAQs] for a detailed description.</p> </div>
--------------------------	---	------------------------	---

Critical_Threshold	80 (1 <= integer <= 100)	Poller	This is a threshold, specified as an integer percentage, that will generate a critical status the poller on the service specified in the Poller_Service parameter if the polling cycle longer than the specified percentage of the specified in the Poller_Proc_Interval. Default so a critical status will be generated if more 80% of the Poller_Proc_Interval is used for
Warning_Threshold	60 (1 <= integer <= 100)	Poller	This is a threshold, specified as an integer percentage, that will generate a warning from poller on the service specified in the Poller parameter if the polling cycle takes longer specified percentage of the time specified Poller_Proc_Interval. Default is 60, so a warning status will be generated if more than 60% Poller_Proc_Interval is used for polling.

Per Service Configuration Parameters

The parameters in this section are generally provided via host externals and service externals which are managed via the Monarch configuration tool.

The (*service*) component of the following parameter names must be substituted with a name for the service, followed by an instance number in square brackets, numbered starting with 1. The (*service*) name specified here need not exactly match the name of the service as it is known elsewhere in *Monarch* or *Nagios*. For instance, a single instance of the `gdma_wmi_disk_C` service would be listed as `gdma_wmi_disk_C[1]`, and it might have some or all of the following parameters defined where the `Check_gdma_wmi_disk_C[1]_Service` value might be `gdma_21_wmi_disk_C` (the name by which this service would be known to GroundWork Monarch and Nagios):

- `Check_gdma_wmi_disk_C[1]_Enable`
- `Check_gdma_wmi_disk_C[1]_Service`
- `Check_gdma_wmi_disk_C[1]_Command`
- `Check_gdma_wmi_disk_C[1]_Timeout`
- `Check_gdma_wmi_disk_C[1]_Check_Interval`

Parameter Name	Default Value (Valid Values)	Used by Spooler/Poller	Description
Check_(service)_Enable	on (on, off)	Poller	Enable or disable this service check
Check_(service)_Service	n/a (string conforming to GroundWork/Nagios service object rules)	Poller	GroundWork/Nagios service name
Check_(service)_Command	n/a (command definition to be executed by the poller)	Poller	Command definition
Check_(service)_Timeout	5 (0- 900 seconds)	Poller	Controls how long the poller will wait for this service command to execute.
Check_(service)_Check_Interval	n/a (positive integer)	Poller	Controls how often the service check will run, as a positive integral multiple of <code>Poller_Proc_Interval</code> . Defaults to 1 (run during every poller cycle) if not specified.

5.2 Error and Status Messaging

Unless configured with `Enable_Local_Logging = on` GDMA will not produce any log output on the target system. Normal messaging is returned to the Master system via the standard spooling and NSCA transmission methods.

- `gdma_auto`, missing configuration - Indicates unconfigured *GDMA* host or corrupt configuration file.
- `gdma-autohost:gdma_auto;3;No configuration file: my_host [192.168.5.13] running MSWin32 5.00`
- `gdma_spooler` startup message - Indicates normal startup
- `my_host:gdma_spooler;0;Spool processor 1.0 started at Wednesday January 6 16:55:27 2010`
- `gdma_spooler` status message - Indicates normal operation
- `my_host:gdma_spooler;0;Spooler transmitted 11 results in 0 secs`
- `gdma_spooler` transmission failure - Indicates that spooler failed in transmitting results
- `my_host:gdma_spooler;2;Failed to transmit 1 results to 192.168.5.74`

- gdma_spooler purge message - Indicates that results are being expired from the spool
- my_host;gdma_spooler;1;Retention timer 900 reached for 38 messages, messages purged
- gdma_poller status message - Indicates normal operation
- my_host;gdma_poller;0;OK Poller processed 9 checks in 8.547 secs Using 1.42% of 600 sec Polling Interval
- gdma_poller startup message Indicates normal startup
- my_host;gdma_poller;0;Poller 1.0 started at Wednesday January 6 17:01:08 2010
- gdma_poller config change message - Indicates that a configuration change has been detected.
- my_host;gdma_poller;1;Configuration change detected :C:\groundwork\gdma\config\gwmon_my_host.cfg at Wednesday January 6 16:13:06 2010

5.3 GDMA Switches

gdma_poll and gdma_spool_processor provide the following switches for troubleshooting. These should not be used in normal operation.

- **Poller Switches** - The GDMA poller agent monitors system statistics on this server, and dumps the results to a spool file. Available options can be shown via a command such as:

```
/usr/local/groundwork/perl/bin/perl gdma_poll.pl -h
```

- Options:

-c <CONFIG FILE>	Config file containing monitoring parameters.
-a <AUTOCONF FILE>	File with default settings. Must contain a target server address.
-l <LOG DIR>	Full path for log directory for this script
-d <DEBUGLEVEL>	Debug mode. Will log additional messages to the log file; <DEBUGLEVEL> should be 1 for moderate logging, or 2 for additional detail.
-h	Displays help message.
-x	Run once. If this option is not selected, run continually with sleep.
-i	Run interactively - shows output to the Command Line Interface (CLI; used in non-service mode) as well as to the log file.
-v	Show version.

- **Spooler Switches** - The GDMA spool processor picks up results from the spool file and sends them back to the GroundWork server(s) using NSCA. Available options can be shown via a command such as:

```
/usr/local/groundwork/perl/bin/perl gdma_spool_processor.pl -h
```

- Options:

-c <CONFIG FILE>	Config file containing monitoring parameters.
-a <AUTOCONF FILE>	File with default settings. Must contain a target server address.
-l <LOG DIR>	Log directory for this script, relative to GDMA HOME.
-d <DEBUGLEVEL>	Debug mode. Will log additional messages to the log file; <DEBUGLEVEL> should be 1 for moderate logging, or 2 for additional detail.
-h	Displays help message.
-x	Run once. If this option is not selected, run continually with sleep.
-i	Run interactively - shows output to the Command Line Interface (CLI; used in non-service mode) as well as to the log file.
-v	Show version.