GMI-Cmd.exe Reference Manual

GMI Command Utility
General Management Interface Foundation
http://www.gmi-foundation.org

Program Description

The "GMI-Cmd.exe" program is a standard part of the GMI program suite, and provides general utility in communicating with agents. The program allows an operator to connect with an agent, query values, set control values, upload packages, and perform general-purpose inspection of agent programs.

The GMI-Cmd.exe program operates without any need for other software. It is executed at a command prompt (or in a batch file), and typically residing in an operator's path. The program is not intended for application development (including web screen development) nor advanced scripting, however provides many useful administrative functions as described in this reference manual.

Of special interest to software developers: The GMI-CMD.exe program is built with open-source calls, available for scripting and program development. A complete description of the Command API associated with GMI-Cmd.exe is available from the "GMI-Cmd.exe API" documentation, provided in a different document.

Command Directives And Syntax

After launching the GMI-Cmd.exe program, commands are typed by the operator at the "GMI>" prompt. The program accepts a small set of commands, documented in detail within the paragraphs below.

The user may obtain brief help on these commands by typing "?" at the "GMI>" prompt. The user may obtain additional help on commands using "(command) ?" at the "GMI>" prompt.

`connect (ipaddr) [ (passkey) ]`

The "connect" command is necessary before querying any agent. Typically, it is the first command issued by the operator on GMI-Cmd.exe program startup.

The "connect" command takes either one or two arguments: the first argument is the IP address of the remote agent in standard dot notation. The second argument is an optional passkey, required ONLY if the agent is configured with a passkey. If no passkey is provided, and the passkey is required by the agent, the connect command will immediately query the operator for the passkey before proceeding.

`disconnect`

The "disconnect" command is optional, and disconnects from the specified agent program. This command is provided mainly for completeness, since successfully connecting to a new agent program via the "connect" command will automatically disconnect the program from any agent when the new connection is executed.

`status`

The "status" command shows the status of the program's connection, if any. When the program is connected, the IP address of the connected device is displayed; otherwise the program displays a "Not connected." message.

`pwd`

The "pwd" (print working directory) command displays the current working agent-directory in standard path notation. This command is useful for printing the current working directory in a fashion similar to UNIX "pwd" commands.

`cd (agent-directory)`

The "cd" (change directory) command is used to traverse the agent-path hierarchy in a fashion similar to the "cd" command of UNIX platforms, or Windows "command" prompts. The command accepts as input a valid agent path (represented in either UNIX style forward slashes, or Windows style backslashes, identical in both cases.)

The operator may execute "cd /" to jump to the top of the agent directory, may specify a relative path, an absolute path, and paths may contain "/./ and "/../" directives to reference the current directory or the parent directory.

The "cd" command fails with an "Invalid path" message if the specified directory is not supported by the agent, or if the current path is in someway invalid.

The "cd" command, if issued with no arguments, is identical to the "pwd" command. The command accepts an absolute or relative agent directory pathname

`ls (agent-directory)`

The "ls" (list directory) command is used to list the contents of the specified agent path in a fashion similar to the "ls" command of UNIX platforms, or the Windows "dir" command. The command accepts as input a valid agent path (represented in either UNIX style forward slashes, or Windows style backslashes, identical in both cases.)

If no "agent-directory" is specified with the "ls" command, then the current agent directory contents is listed. If an invalid directory is specified, the program returns the message "No list information is available."

If the "ls" command references agent directory containing other sub-directories, the contents of the subdirectories are listed. Conversely, if the "ls" command is in a terminal sub-directory that has read access, the contents of the terminal sub-directory are returned. This provides the central method of acquiring data from the agent.

`ls –b (agent-directory)`

The "ls –b" (list directory brief) command is identical to the standard "ls" command, except does not display blank lines or any annotations in the resulting output. This may be easier for some programs to parse. Except for this, the "ls –b" command behaves in an identical fashion to the "ls" command.

`ls –r (agent-directory)`

The "ls –r" (list directory recursive) command lists directory paths, but not values. All the directory paths beneath the specified agent directory are listed. In particular, to list all the agent paths on the system, supported by the currently connected agent, the operator issues the command "ls –r /". This is useful to the operator when detecting the currently configured capabilities for a connected agent.

`set (agent-directory) = value`

The "set" command is used to set a value for a writable agent. The command syntax requires three components: a writable agent path must be specified (in standard "ls" and "cd" notation, described above), followed by an "=" character, followed by the value to be set (up to 1000 characters in most cases.)

The "agent directory", specified as the first argument, must be settable. Note that not all agent values are writable or settable, and it is up to a particular application on whether the agent directory value can be set or not. Typically, any path that can be set to a value is identified with an access of either "RW", or "-W" or "-W+" in the second column of the listing.

Attempting to set a path that is not settable results in the error "Agent path is not settable". If the object is settable, but for some reason the GMI App disagrees with the type of value set (for example setting an integer value to a text string) the error "Invalid set value for path" is displayed.

Note that the "=" character may be preceded or followed by any number of spaces. Further note that the value (following the "=" character) will have all leading and trailing spaces removed, and multiple spaces compressed. This assists with parsing of values fetched by the user.

`upload (agent-directory) = (localfile)`

The "upload" command is similar to the "set" command, except that values are specified by the contents of a named file, which follows the "=" character in the command. This command permits an operator to upload an entire file to a particular path of the agent. The file can be downloaded via the "download" command, can be listed (in most cases) via the "ls" command, and can be operated on by other "set" commands.

The "upload" command can upload either textual or binary files. The type of data accepted by the agent directory is strictly a function of the GMI App associated with the agent path. Note that not all agent values are uploadable. Typically the uploadable path will be identified with an access of either "RW+" or "-W+" in the second column of the listing. In most cases, the subdirectory name will also end in a "+" character to indicate that the path may be uploaded.

In some situations, a value may be both "settable" as well as "uploadable" (for example the standard "/system/SysInfo" path accepts both a "set" and "upload" value.) The combination available to the operator is strictly a function of the GMI App for the path.

`download (agent-directory) = (localfile)`

The "download" command complements the "upload" command, reading the content of a file to the specified local file. This provides a method of saving binary or textual data to a file. Note that textual items may also be redirected to a file via the ">" and ">>" operators (similar to a UNIX shell or Windows cmd.exe program.) The main use for the "download" command is to acquire the contents of an uploaded binary file or executable.

`install (localfile)`

The "install" command permits the end-user to install new applications at the agent. The command accepts the pathname to a local file on the system, which must be a signed GMI App. If the agent accepts the application, this can open a new set of agent directories at the agent, which can be traversed with the "cd" command, listed with the "ls" command, and which may support "set" and "upload" commands.

The standard "/install" path of the agent can be used to list the contents and the signature information associated with any installed GMI App. When a new App is installed using the "install" command, it is listed in the "/install" path along with the module name, author name, contact information, and other data.

The "install" command is the principle way of adding new functionality to the agent program. More information on GMI Apps is available via web resources.

`uninstall (agent-directory)`

The "uninstall" command permits the end-user to uninstall an existing application, possibly to free up agent resources. The command accepts an agent directory as an argument, where any path within the installed App can be specified.

The root directory for the App is always available via the "/install" directory of the agent program, along with contact information. If the "uninstall" is successful, the App information is removed from the "/install" directory along with any associated agent paths or functions for the App package.

`type (localfile)`

The "type" GMI command is provided for completeness. This internal command operates exactly the same as the Windows "type" command (i.e. displays the contents of a disk file to standard output), but uses the built-in "pipe" capabilities. Therefore, this command can pipe data to the "parser" function, described later in this manual. It is also useful for simply viewing local disk files.

Agent Directory Paths

The particular agent paths, which can be viewed with an "ls" command, or traversed by the "cd" command, are strictly a function of the Apps that are currently installed at the agent. When a user installs a package via the "install" command, this opens up new directories in the agent path hierarchy. The operator can see all paths supported by the currently connected agent by issuing the "ls –r ./" command.

The syntax of the agent path follows closely that found in UNIX and Windows system. For example, the "/" path always references the top-level directory of the agent. The path can be relative to the current working directory (displayed with the "pwd" command.) To list the parent directory, the "ls ../" or simply "ls .." command can be used. To list the current directory, the "." character is used.

For example, the command "ls /install/../install/InstallCount/../.." is identical to the "ls /" command (because the ".." characters reference the parent directory.) This syntax will be extremely familiar to UNIX and Windows cmd.exe users.

The following conventions apply to agent paths:

- Agent paths, while displayed in a certain case, are case-insensitive to the "cd", "ls" and other commands. The user may specify either upper or lower case letters, or any combination thereof.

- Each directory path contains a "Type" a setting, listed as the first column of output of the "ls" command. The directory type will always be "Folder" if the directory contains folders. The directory type will always be "File" if the folder is uploadable. Other than that, the "Type" setting is defined by the Application, typically either "Integer", "Text", "Counter", "Gauge", "IPAddr", "Binary", or some other setting that gives an indication of the path contents.

- Each directory path contains a "Permission" setting, listed in the second column of output of the "ls" command. The permissions of folders is always "R-" (Read only.) The permission of subdirectories will be either: "R-" for read only; "RW" for read and write; "-W" for write only; "RW+" for readable, writable and uploadable objects; or "-W+" for write only uploadable objects. The actual permissions are under the control of the application associated with the path.

- Finally, as a convenience, folder names are typically followed by a "/" character, and uploadable objects are followed by a "+" character. Folder names, by convention, are always in lower case letters, whereas scalar subdirectories (containing information) always begin with a capital letter. Note that application developers should follow these conventions, but no enforcement requires this convention (other than the general usability of the GMI App.)

Installing And Uninstalling Packages

By default, the Agent comes with a limited number of folders, most of which are empty. To add new functions to an agent, the operator will execute the "install" command (documented elsewhere), which uploads the package, installs it, and opens new directories at the agent program to provide control and monitor functions.

The actual functionality of a GMI App is documented by the supplier of the App (which may be open source, or a commercially available package.) Each GMI App is useful ONLY given the accuracy of the documentation that is supplied with it. Additionally, any GMI App may contain functionality that is not compliant or in accordance with the functions of the agent or managed system.

Given that, the GMI system enforces a "certification" process, required by all Apps, that identifies the author and contact information for the App. This prevents a malicious App from being installed without at least knowing the contact information for the App.

If an attempt is made to upload an unsigned and uncertified App, the CO-cmd.exe (and agent) program will block the installation, displaying an "Invalid package" error message.

Executing External Commands

An external command can be executed at the GMI prompt by simply prefixing the command with a "!" exclamation point. This syntax will be familiar to UNIX users, and is the same syntax used in programs like FTP and command line debuggers.

This provides a simple method of passing control to the operating system, such as to run a batch file, editor, or other program in the local system. This function may also be particularly useful for scripting more complex systems, as discussed elsewhere.

Redirecting Output

The GMI-Cmd.exe program supports basic redirection of the "ls" command using standard operators: ">" creates the user specified file; ">>" creates or appends the user specified file; "|" places the standard output of the file to the named program.

For example, the following command lists the contents of the GMI Agent "/system/SysInfo" path, and pipes the output to the standard Windows "more" command:

```
ls /system/SysInfo | More
```

As another example, the following command finds any occurrences of the keyword "Test" in the output of the "/system/SysInfo" path, and places the result in the file "test.txt" in the temp directory of the local system:

```
ls /system/SysInfo | find "test" > c:\temp\test.txt
```

The above command closely follow the syntax familiar to users of UNIX shells or the Windows cmd.exe program.

Piping Data To The Build In Parser

In addition to allowing the user to pipe to external filters such as "more" and "find", the GMI-Cmd.exe program includes a simple built-in parser slightly reminiscent of "awk", which permits the operator to match a pattern, and parse the matched result into fields.

The "parse" function is documented in the GMI Parser reference manual in detail, available from the GMI Foundation website. The syntax of the parser is briefly documented here.

To send data to the parser, the user simply pipes the output of a GMI command to the "parse" function. The "parse" function accepts either one or two arguments. The syntax of the parse function is as follows:

```
command | parse  "matchpatt" fieldspec
```

Note that the "parse" command reads standard input of a GMI command. It is not usable within the GMI-Cmd.exe program except on the right side of a pipe character as shown above.

The "matchpatt" argument is simply a quoted string containing a keyword in the standard input to the program. Any line that does not match the specified string is rejected. The matchpatt is case-insensitive. If the "matchpatt" argument is not specified, all lines are matched.

The "fieldspec" argument is a selection of fields, similar to "awk" syntax, where $1 is the first word of output, $2 is the second word of output, $3 is the third word of output, etc. If the user specifies a "matchpatt" value, then only matched lines are parsed into fields. If the user does not specify a "matchpatt" value (but only the fieldspec) then all lines are matched, and the specified fields are output.

The operator must supply either a "matchpatt", or a "fieldspec", or both. If the operator includes just a "fieldspec", then all lines are matched. If the operator includes just the "matchpatt" argument, then all lines that match the pattern are returned. Examples are as follows:

1. Return all lines in the listing that match "sys". The command below uses the "parse" function to display any lines in the output of the "system" listing that contain the keyword "sys".

   ```
   ls  /system | parse "sys"
   ```

2. Return the type of the "sysInfo" folder. The command below uses the "parse" function to display the first column of the "sysInfo" line in the "/system" directory listing. As with all GMI folder listings, the type of the object is the first word:

   ```
   ls /system | parse "sysinfo" $1
   ```

3. Return the value of "Bytes_Received" from the "netstat" application. The command below lists the "eth" values (which consists of multiple "label" and "value" pairs) and returns the value for the line that has "Bytes_Received" as the label, i.e. the second word of the line that matches "bytes_received":

   ```
   ls /perf/netstat/eth | parse "bytes_received" $2
   ```

4. Return the names and percent used of all the disks listed in the "diskmon" application. The command below lists the "diskmon" values, and returns the first

column of each listing (which is the disk name), and the fifth column of the listing (which is the disk percent used.) Since no match pattern is specified, all lines are matched.

```
ls /perf/diskmon | parse $1 $5
```

Additional notes regarding the GMI parse function, including information useful to application developers, is documented on the GMI Foundation website. The "parse" command is provided mainly for convenience to more advanced users; detailed understanding of this function is useful but not required.

Note that, as an alternative to using the built-in "parse" command, an operator can also use the "awk" external program, which provides similar functions to the built-in parser, and similar command syntax. The "awk" command is widely documented, and available for a variety of systems.

Batch File Scripting

The GMI-Cmd.exe program does not incorporate a scripting interface, but permits redirection of standard input, and supports a special "-q" argument that will suppress the GMI command line prompts. This permits the operator to run the GMI-Cmd.exe program in batch mode.

For example, the following batch file lists the contents of the "system" directory of an agent executing at address 192.168.1.90:

```
@echo off
(
echo connect 192.168.1.90
echo ls -b /system/sysname
echo ls -b /system/sysdescr
echo ls -b /system/syslocation
echo ls -b /system/syscontact
echo ls -b /system/sysTime
)| .\GMI-cmd.exe –q
```

The above information, when transcribed to a batch file, will query the basic information from the agent described above.

Note that more suitable tools (such as the GMI-XML.exe program) are available for web scripting or special applications of the end user. Consult web resources for more information

Programmer API

All of the various functions of the "GMI-Cmd.exe" program, documented here, are also available to software developers using the GMI-Cmd32.lib link library. Software programmers interested in developing command programs are encouraged to download the freely distributed "GMI-Cmd-API.zip" package, available from the GMI Foundation website.

This package contains link libraries, example programs, and programmer level documentation sufficient to use any of the GMI command functions re in a C / C++ program. This furnishes a simple method of building utilities, command programs, or more complex system software that can communicate with GMI agent programs.

The GMI-Cmd-API.zip file is freely distributed, and can be used by any software developer or organization without limit.

About The GMI Foundation

The GMI Software consists of agents, utilities, documentation and API's intended to promote secure and flexible management of end-points. GMI fully supports Open Source Initiatives, and views its role as one of not-for-profit promotion of software to correct the endemic problems associated with system and software management. More information on the GMI Foundation is available at the following location:

GMI Foundation
http://www.gmi-foundation.org
mailto: info@gmi-foundation.org

About Our Technical Partners

Additional information on the GMI Software, including a selection of useful GMI Apps, as well as professional and support services, is available from various members of the GMI Foundation.

A central clearinghouse for GMI applications is provided by Vallum Software, LLC, which provide support, certification, and development solutions, as well as the "Halo Management System", based on the GMI agent. Visit the link below.

Vallum Software, LLC
http://www.vallumsoftware.com
mailto: info@vallumsoftware.com