Graviton Administrators Manual

1. Introduction

The Graviton server provides you with a single configuration point for all hosts in your enterprise. No longer will you have to update every host individually.

Gravity Edge Systems Graviton™ provides a way for IT departments to meet many parts of your company's security policy. Graviton allows you to keep all your computer systems synchronized, clean, up-to-date and makes it incredibly easy to recover an entire system or an entire enterprise of systems in case of disaster.

Graviton also audits the entire static portion of your file system and backs up the variable portion with an easy to use command line interface. Graviton is great for those who want to deploy systems and maintain them easily once they are in the field.

The framework allows you to update a single package and have all the hosts that are subscribed to that package, receive that updated version. You may use Graviton to control any host and to any degree.

2. Directory Structure

Graviton's directory structure is divided into 2 primary sections, *local* and *shared*. Directories and files located beneath the *local* section are items which should remain on the local file system and are in most cases specific to the local system. Directories and files located beneath the *shared* section are items which may be shared or synchronized with other Graviton servers.

Here is an example of what a typical Graviton directory structure looks like:

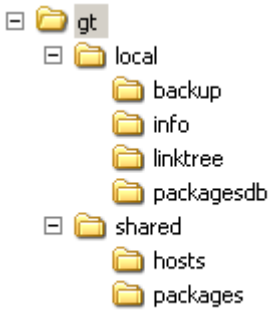

Figure 1 - Directory Structure

2.1 Local

Directories and files found beneath *local* are largely informational and primarily used for Graviton's internal processes.

An exception to this is the *backup* directory which is located beneath *local*. The *backup* directory contains the contents of backups performed by the command line utility 'gt' in conjunction with the '-b' flag. This data is useful and may be copied elsewhere for backup purposes.

2.2 Shared

The *shared* directory is where a Graviton administrator will spend most of their time. This directory contains all of the host and package configuration and data. All of the content located beneath the *shared* directory is suitable for synchronizing to other Graviton servers for load balancing or to another medium for backup purposes.

3. Hosts

3.1 Host Structure

Hosts are located under the *gt/shared/hosts* directory.

A host in Graviton is defined as a directory which contains the following:

- a directory named files
- a file named host.conf

Typically hosts are sorted by DNS hierarchy in a directory structure that matches your DNS structure.

Here is an example:

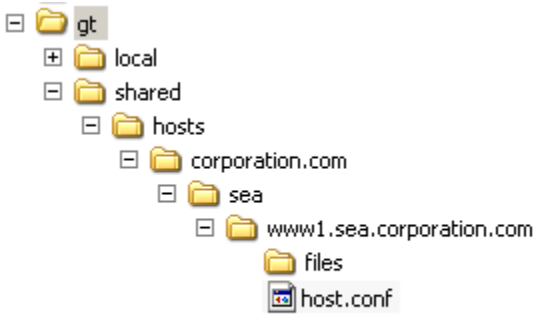

Figure 2 - Host example

Here, the actual host is *www1.sea.corporation.com*, because it contains a directory named *files* and a file named *host.conf*. The directory *corporation.com* is only organizational as is the directory *sea* beneath it. You may organize your

hosts any way you see fit. The entire *hostdir* directory as defined in */etc/graviton/gt.conf* will be searched for all valid hosts when the *gt* CLI command is run.

The *files* directory becomes '/' or 'root' on the destination host. For example, if you were to put a file named 'foo' under *files*, the file 'foo' would be pushed down to '/' on www1.sea.corporation.com. So if you did an 'ls' in '/' on www1.sea.corporation.com you would see 'foo' listed with the other files and directories.

The purpose of the *files* directory is to override or provide a place to put files and directories that are specific to that host. Files and directories placed in the *files* directory will take precedence over files and directories from packages you include that contain the same file and directory structure.

For example, in Debian, /etc/network/interfaces holds the network configuration parameters for the host. Since almost every host has different network configuration, you may place a copy of that hosts interfaces file in *etc/network* inside the *files* directory.

Now we have:

Figure 3 - Files example

Now when the host www1.sea.corporation.com is synchronized by Graviton, the directory structure and file will be pushed down to the host.

3.2 Host Creation

There is one way to create a host for Graviton to use. This is to use the normal Linux command line utilities to create the directories as laid out in section 'Host Structure' and 'Host Configuration'.

3.3 Host Configuration (host.conf)

The host configuration file *host.conf* is where Graviton looks to gather all the subscription information for a host. Any Unix text editor may be used to edit this file, don't use an editor which places DOS line ends in the file.

The following is an example of a typical *host.conf* file.

```
base debian-ix86-sarge
linktag current
pkgtype deb
groups all

include meta/base26
include kernels/kernel-2.6.12.5-i686-p4
include meta/developer
include meta/auth-scheme/ad
```

Figure 4 - host.conf example

3.3.1 base (context: host.conf)

The *base* directive lets Graviton know which package tree base to *include* packages from (refer to section 'Packages' for more information).

Here is an example:

base redhat-ix86-rhel4as

3.3.2 linktag (context: host.conf)

The *linktag* directive lets Graviton know which symbolic link to use when the dependency engine processes all of the includes from the packages subscribed to in the *host.conf* (refer to section 'Packages' for more information).

Here is an example:

linktag testing

3.3.3 pkgtype (context: host.conf)

The *pkgtype* directive lets Graviton know which package type this host is using. The package type is typically tied in with the *base* that the host is subscribed to. In the above example, this host is going to use the *debian-ix86-sarge* base. It just so happens that a majority of the packages located under *debian-ix86-sarge* are Debian (.deb) style packages that were placed there by the CLI utility *deb2gt*. Graviton uses this flag in combination with *debian_status* (/etc/graviton/gt.conf) and *redhat_rpmdb* (/etc/graviton/gt.conf) to determine whether or not it should build a */var/lib/dpkg/status* file on deb based hosts or build *gt/local/packagesdb/<hostname>/status.tmp* for rpm based hosts.

On rpm based machines, it's required that all of the rpm files are present on the remote system to rebuild the rpm databases from scratch (refer to section 'Maintenance' for more information). On deb based hosts, the status file is built by Graviton and placed automatically into the host's *files* directory in the appropriate location (/var/lib/dpkg).

It should be noted that while every effort was made to reconstruct the package database files for deb and rpm based hosts, they may not be perfect and may contain errors.

Normally, you won't need these package databases because Graviton is handling package deployment, however if you wish to add a package on the fly without using Graviton, these package databases will be used by the different package management utilities. The package installation will most likely fail or generate a massive amount of errors if these databases are missing.

Here is an example:

pkgtype rpm

3.3.4 groups (context: host.conf)

The *groups* directive lets Graviton know which host groups this host belongs to. Groups are used so that you may run the *gt* CLI utility with the *–g* option and provide it a comma delimited list of hosts you wish to synchronize. Group names should have no spaces and be lower case. If you want to place a machine into more than one group, just place a ',' (comma) after the first group name and enter your second group name.

Here is an example:

groups sea,web,all

or

groups sea

An example of *gt* utilizing one of the above examples:

gt –g web –v

3.3.5 include (context: host.conf, package.conf)

The *include* directive lets Graviton know which packages the host should be subscribed to.

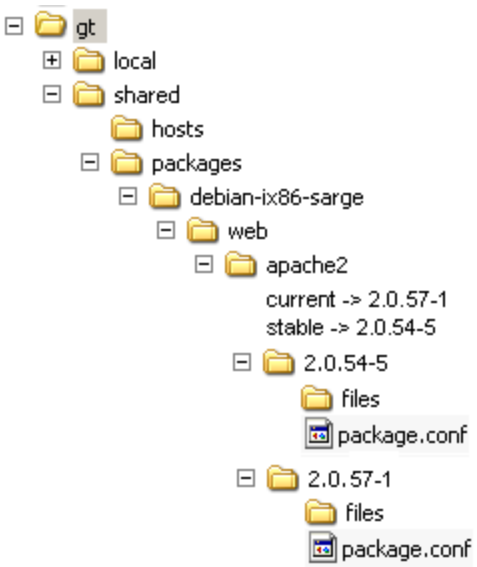

Figure 5 – Multiple packages with multiple symbolic links example

In the above image, there are 2 packages; 2.0.54-5 and 2.0.57-1. There are also 2 symbolic links in the package name directory (apache2), current and stable. The *linktag* defined in the *host.conf* (refer to section B. Host Configuration) determines which symbolic link will be used if the full package name *web/apache2* were to be listed in a host's *host.conf* as an *include*.

Here is an example line in a *host.conf*:

include web/apache2

Another value may be placed after the package name delimited by a space to choose a specific version if you do not want the one that *linktag* defines. Let's assume the value of *linktag* is 'stable'.

include web/apache2 current

or

include web/apache2 2.0.57-1

Both of the above includes will subscribe the host to the specified versions instead of the one defined by the *linktag* value. WARNING: Even though you may have included a specific version by using this method, if another package has an include for *web/apache2* and the value of *linktag* is 'stable', the 'stable' version will also be included and may cause problems. Graviton will issue a warning anytime it sees two or more packages of different versions included with the same package name on the same host.

The only limitation to how many *include* statements you may have in a *host.conf* or *package.conf* is based on your systems ram and hard drive space.

3.3.6 exclude (context: host.conf, package.conf)

The *exclude* directive lets Graviton know which patterns should be excluded when synchronizing the remote host. This is commonly used to prevent Graviton from wiping out variable data such as log files, pid files, caches, database files, and other transient and variable data. A more specific an *exclude* pattern will make it easier to maintain a package tree than broad sweeping patterns which cover many files. Best practice is to make each package *exclude* transient data which it generates. Here is an example *package.conf* for the apache2 package which has several *exclude* statements.

```
include libs/libpcre3
include libs/libaprutil1
include libs/libpq4
include libs/libsqlite3-0
include libs/libdb4.4
include web/apache2.2-common
include libs/libldap2
include libs/libapr1
include libs/libuuid1
include libs/libexpat1
include libs/libc6

exclude /var/log/apache2/access.log*
exclude /var/log/apache2/error.log*
exclude /var/run/apache2.pid

install-execute "/etc/init.d/apache2 start"
execute "/etc/init.d/apache2 reload"
remove-execute "/etc/init.d/apache2 stop"
```

Figure 6 - package.conf example

3.3.7 exclude-backup (context: host.conf, package.conf)

The *exclude-backup* directive lets Graviton know which files you would like to 'exclude' from being automatically put in the backup list when you use the 'exclude' directive. For instance, you may want to 'exclude' all of the files ending in .pyo, so you used a statement like this in the package.conf for python2.4 "exclude *.pyo" to exclude all *.pyo from being touched by the system. When you did that, all "*.pyo" files were automatically added to the backup list. If you really don't want to back those files up, you may use an exclude-backup statement.

Here is an example:

exclude-backup *.pyo

3.3.8 block (context: host.conf, package.conf)

The *block* directive lets Graviton know which packages the host should not be subscribed to. Blocks are most commonly used to prevent the inclusion of a specific package even if it's depended upon by an already included package.

The *block* directive should be used extremely sparingly within any *package.conf*. Blocks placed within a *package.conf* may adversely affect other hosts which depend on the blocked package. The use of *block* should primarily be used within a *host.conf* where the scope of the *block* statement is limited to just that host.

When a *block* is used, not only will the package being blocked not be included, but all dependencies located in that package's *package*.conf (unless otherwise included from a separate and non-blocked package).

When using *block*, only the name of the package need be mentioned and not the section in which it resides

Here is an example:

Correct

block apache2

Incorrect

block web/apache2

.

4. Packages

4.1 Package Structure

Packages are located under the *gt/shared*/*packages* directory.

A package in Graviton is defined as a directory (typically a version number) which contains the following:

- a directory named files
- a file named package.conf

Typically packages are sorted by their function under a base directory that is usually named after an operating system.

Here is an example:

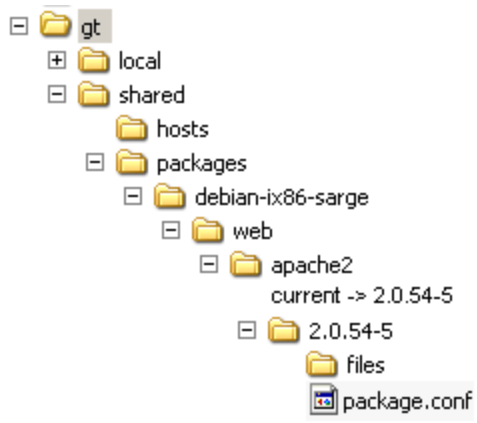

Figure 7 - package example

Here, the actual package is *2.0.54-5*, because it contains a directory named *files* and a file named *package.conf*. The directory *debian-ix86-sarge* is a *base directory*. The directory *web* is a section that the Debian distribution sorts packages into and *apache2* is the name of the Debian package. For all intents and purposes within Graviton, this package will be accessed from a host's *host.conf* and other package's *package.conf* files in an *include* statement like this:

include web/apache2

The *current* symbolic link points to a version to be used in the *current* tree. The directory *files* contains all of the files for the package. Anything put into the *files* directory will be transferred to '/' or 'root' on the client host.

The file *package.conf* is similar to a hosts *host.conf* in that it holds dependency or subscription information.

4.2 Package Creation

There are two ways to create a package for Graviton to use. The first is to let a Graviton utility (deb2gt or rpm2gt) do it for you or manually create it yourself.

4.2.1 deb2gt and rpm2gt

These two utilities unpack .deb and .rpm packages and place them into a Graviton package tree. The utilities make no effort to ascertain which distribution version these packages should belong to (e.g. Sarge, Sid, Fedora 7.1, Red Hat Enterprise AS 4, SUSE 9, etc). However the utilities do let you pass an

argument to them which allow you to place the packages into a package tree of your choice. Please refer to the manual pages for more information on using these utilities.

4.2.2 manual package creation

Using the information in sections ‘Package Structure’ and ‘Package Configuration’, you may use normal Linux command line utilities to create the required directory structure and files which make up a Graviton package.

4.3 Package Configuration (package.conf)

A *package.conf* is the brains of a package. It may contain the following directives: include, exclude, block, install-execute, execute, and remove-execute.

4.3.1 include (context: host.conf, package.conf)

See section 3.2.5

4.3.2 exclude (context: host.conf, package.conf)

See section 3.2.6

4.3.3 block (context: host.conf, package.conf)

See section 3.2.7

4.3.4 install-execute (context: package.conf)

The *install-*execute directive lets you specify a command you'd like to run when a package is installed. An *install-execute* runs after the Rsync process occurs. Use quotes ("") around any command you'd like to run.

Here is an example:

install-execute "/etc/init.d/apache-ssl start"

You may also stack commands by placing a ';' between them. You may notice that some commands won't run unless you specify their full path. This is because root doesn't have a proper environment on the remote host. Chances are the PATH environment variable isn’t set properly on the remote host when Graviton tries to SSH in a run the command. This is a problem with the ‘root’ accounts environment on the remote host and not a problem with Graviton.

Example:

install-execute "touch /var/log/db.log ; /etc/init.d/db start"

You may also use multiple *install-execute* lines.

Example:

install-execute "su - postgres ; initdb ; createdb db ; createuser dbuser"
install-execute "/etc/init.d/postgresql start"

4.3.5 execute (context: package.conf)

The execute directive lets you specify a command you'd like to run every time a host is synchronized by Graviton. An *execute* runs after the Rsync process occurs. Use quotes ("") around any command you'd like to run.

Example:

execute "/etc/init.d/postfix reload"

4.3.6 remove-execute (context: package.conf)

The *remove*-execute directive lets you specify a command you'd like to run when a package is removed. A *remove-execute* runs before the Rsync process occurs. Use quotes ("") around any command you'd like to run.

Example:

remove-execute "/etc/init.d/apache2 stop"

5. Command Line (CLI) Tools

5.1 gt

This is the primary utility in the Graviton suite. Please refer to the man (manual) page that is distributed with the software for instructions on its usage.

5.2 gtdbdump

This utility dumps the contents of any .db file Graviton creates. Please refer to the man (manual) page that is distributed with the software for instructions on its usage.

5.3 gtdbgen

This utility will regenerate some database files that Graviton relies upon. It's a good idea to run this nightly while no other Graviton processes are running. Please refer to the man (manual) page that is distributed with the software for instructions on its usage.

5.4 deb2gt

This utility will process .deb style packages and place them into the package tree. Please refer to the man (manual) page that is distributed with the software for instructions on its usage.

5.5 rpm2gt

This utility will process .rpm style packages and place them into the package tree. Please refer to the man (manual) page that is distributed with the software for instructions on its usage.

6. Best Practices

6.1 Administrative Directories

The following are all defined in '/etc/graviton/gt.conf'.

packdir	(location of the package trees)
backupdir	(location of backed up data)
hostdir	(location of host configurations)
treedir	(location of the linktree)
dbdir	(location of host database files)
bindir	(location of the Graviton binaries)
infodir	(location of administrative databases)
logdir	(location of Graviton log files)

Here is an example of a gt.conf

```
# GT directory structure options
packdir /gt/shared/packages
backupdir /gt/local/backup
hostdir /gt/shared/hosts
treedir /gt/local/linktree
dbdir /gt/local/packagesdb
bindir /gt/bin
infodir /gt/local/info
logdir /var/log/gt

# Default mail address to send email reports to.
email youremail@yourdomain.com
# If yes, you don't have to give the -m flag for email reports,
# they will just be sent to the default address.
email_alwayson yes

# If set to 'no', nothing will be ignored.  This is used to filter
# out itemized file changes in the GT report.
```

```
# For example:  If you no longer wish to see directories which are
# being updated because only their modify time has changed,
# you could put '\.d\.\.t\.\.\.\.' as the value.  This is a regular
# expression, so you may add more statements by putting
# a | (or) inbetween your patterns, like pattern|pattern.

report_ignore \.d\.\.t\.\.\.\.

# Default package directory for deb2pkg to unpack .deb's into.
deb2gt_defbase debian-ix86-etch
# Default package directory for rpm2pkg to unpack .rpm's into.
rpm2gt_defbase redhat-ix86-rhel4as
# Should GT create the /var/lib/dpkg/status file based on the
# packages your host is subscribed to?
# The resulting file will be placed in {hostdir}/{hostname}/files/var/lib/dpkg
# The status file is used by apt to determine what packages are already
# installed and should only download new or updated packages.
debian_status yes

# Should GT create the /var/lib/rpm/Database files based on the packages
your
# host is subscribed to?
# The resulting files will be placed in {hostdir}/{hostname}/files/var/lib/rpm
redhat_rpmdb yes

# Moving window of days of backups to keep
backup 3
```

6.2 Package Databases in 'infodir' (see /etc/graviton/gt.conf for 'infodir' value)

The package databases should be rebuilt on a nightly basis or when no other Graviton activity (mainly deb2gt and rpm2gt) is going on. The 'gtdbgen' utility will search through all of your package trees and re-build databases that Graviton depends upon. Every database file (.db) in 'infodir' (typically /gt/local/info) may be regenerated by running 'gtdbgen'.

Example nightly cron entry:

```
0 0 * * * root test -x /usr/bin/gtdbgen && /usr/bin/gtdbgen
```

6.3 Backing up Graviton

We recommend that you back up your Graviton installation on a nightly basis.

Example 'rsync' command:

```
cd /usr/local
```

```
rsync -avvz --delete --exclude gt/local/linktree gt \
backuphost.company.com:/backupdir
```

The command above is run from the Graviton server and will backup everything in the Graviton directory structure except ‘gt/local/linktree’ which is under most circumstances is unnecessary to backup. This command also assumes all of your administrative directories are kept under the same root (e.g. /usr/local/gt) directory.

6.4 Running Graviton against a Graviton Server

It is recommended that you put all of your Graviton Servers under control of Graviton. This is no different than putting any other host on Graviton except that you’ll want to exclude certain Graviton directories.

Here is an example package.conf for the Graviton package.

```
exclude /usr/local/gt/shared
exclude /usr/local/gt/local
exclude /var/log/gt/*.log
```

If your Graviton base installation is elsewhere, you’ll need to modify the path information to match your locations. For example, if the gt root directory is in /vol, you’d want to modify ‘/usr/local’ to ‘/vol’.

6.5 Graviton directory layout

Even though all of Graviton’s administrative directories are definable in ‘/etc/graviton/gt.conf’, you should keep ‘packdir’ and ‘treedir’ on the same volume for performance reasons. It’s easiest for backup purposes if all directories are kept under the same root (e.g. /usr/local/gt or /mylargevol/gt) directory.