Gnome Display Manager Reference Manual

Martin K. Petersen

mkp@mkp.net

George Lebl

jirka@5z.com

Brian Cameron

Brian.Cameron@Sun.COM

Bill Haneman

Bill.Haneman@Sun.COM

Table of Contents

Terms and Conventions Used in This Manual

This manual describes version 2.6.0.4 of the GNOME Display Manager. It was last updated on 7/14/2004.

GDM - Gnome Display Manager. Used to describe the software package as a whole. Sometimes also referred to as GDM2.

gdm - The Gnome Display Manager daemon (`gdm`).

Greeter - The graphical login window (`gdmlogin` or `gdmgreeter`).

Standard Greeter - The standard login window (`gdmlogin`).

Graphical Greeter - The themable login window (`gdmgreeter`).

Chooser - The host chooser which appears on remote displays sending INDIRECT queries (`gdmchooser`).

Configurator - The configuration program (`gdmsetup`).

Paths that start with a word in angle brackets are relative to the installation prefix. I.e. `<share>/pixmaps/` refers to `/usr/share/pixmaps` if GDM was configured with `--prefix=/usr`. Normally also note that GDM is installed with `--sysconfigdir=/etc/X11`, meaning any path to which we refer to as `<etc>/gdm/PreSession` usually means `/etc/X11/gdm/PreSession`. Note that for interoperability it is recommended that you use a prefix of `/usr` and a sysconfdir of `/etc/X11`.

Overview

Introduction

GDM is a replacement for XDM, the X Display Manager. Unlike its competitors (X3DM, KDM, WDM) GDM was written from scratch and does not contain any original XDM / X Consortium code.

For further information about GDM, see the the GDM website[2].

The GDM Daemon

GDM was written with simplicity and security in mind. The overall design concept is this:

Upon startup the `gdm` daemon parses its config file `gdm.conf`. For each of the local displays `gdm` forks an Xserver and a slave process. The main `gdm` process will then listen to XDMCP requests, if so configured, from remote displays and monitor the local display sessions. The main daemon process will also allow starting of on new local Xservers on demand using the `gdmflexiserver` command.

The `gdm` slave process opens the display and starts `gdmlogin`, the graphical login program. `gdmlogin` runs as a dedicated user and communicates asynchronously with the slave process through a pipe. Alternatively `gdmgreeter` command can be used which is the same as `gdmlogin` but allows greater themability. `gdmgreeter` is referred to as the Graphical Greeter, while `gdmlogin` is refereed to as the Standard Greeter.

GDM relies heavily on the presence of PAM, Pluggable Authentication Modules, but supports regular crypt() and shadow passwords on legacy systems.

Remote displays can connect to the XDMCP port on the GDM host. `gdm` will grant access to hosts specified in the gdm service section in your TCP Wrappers configuration file. GDM does not support remote display access control on systems without TCP Wrappers. XDMCP support can be turned off completely, however.

GDM includes several measures making it more resistant to denial of service attacks on the XDMCP service. A lot of the protocol parameters, handshaking timeouts etc. can be fine tuned. The defaults should work for most systems, however. Don't change them unless you know what you are doing.

In general GDM is very reluctant regarding reading/writing of user files. For instance it refuses to touch anything but regular files. Links, sockets and devices are ignored. The value of the RelaxPermissions parameter determines whether GDM should accept files writable by the user's group or others. These are ignored by default.

All operations on user files are done with the effective user id of the user. If the sanity check fails on the user's `.Xauthority` file, a fallback cookie is created in `/tmp`.

Note that normally it is assumed that the home directory is only readable by the user. However NFS traffic really goes "over the wire" and thus can be snooped. For setups with NFS directories you should really use the `UserAuthDir` and set it to some local directory such as `/tmp`. GDM will try to open the normal authorization file for reading as root, and if it fails, then it will conclude that it is on an NFS mount and it will automatically use `UserAuthFBDir`, which is usually `/tmp`. This can be changed by setting `NeverPlaceCookiesOnNFS` in the `[security]` section to false.

GDM implements only the MIT-MAGIC-COOKIE-1 authorization scheme, see the XDMCP section for more information about this, especially relating to using X over the network.

Finally, the sysadmin can specify the maximum file size GDM should accept, and, if the face browser is enabled, a tunable maximum icon size is also enforced. On large systems it is still advised to turn off the face browser for performance reasons. Looking up icons in homedirs, scaling and rendering face icons can take quite a long time. YMMV.

GDM also has a unix domain socket which can be used to control certain aspects of behavior, or to query information about running servers or logged in users. This is the `/tmp/.gdm_socket` and the protocol is described in the sources in the `daemon/gdm.h` header file. The `gdmflexiserver` command uses this for example to launch on demand X servers for the user.

Different Display Types

GDM allows 3 different display types. First local static X servers. These are always run, and when they die or are killed, they are restarted. GDM can run as many of these as needed. GDM can also manage servers on which it does not manage a login itself, thus allowing GDM to be used when building X terminals.

Next GDM supports flexible or on demand servers. These are run by requesting one using the socket protocol. There is a command, `gdmflexiserver`, which can do this for the user. For standard X servers the user must be logged in from a console, on one of the servers that GDM has run. This command can however also launch nested `Xnest` servers which can be started even from non-console logins. This is generally done by running `gdmflexiserver -n`. These servers are not restarted when the user session ends. `gdmflexiserver` normally also locks the users screen before running a new server with xscreensaver.

Last display type is the XDMCP remote displays that are described in the next section. Remote hosts can connect to GDM and present the login screen if this is enabled. Some things may be different for these sessions, such as the Actions menu which allows you to shut down, reboot, or configure GDM will not be shown.

XDMCP

GDM also supports the X Display Manager Protocol (XDMCP) for managing remote displays.

GDM listens to UDP port 177 and will respond to QUERY and BROADCAST_QUERY requests by sending a WILLING packet to the originator.

GDM can also be configured to honor INDIRECT queries and present a host chooser to the remote display. GDM will remember the user's choice and forward subsequent requests to the chosen manager. GDM also supports an extension to the protocol which will make it forget the redirection once the user's connection succeeds. This extension is only supported if both daemons are GDM. It is transparent and will be ignored by XDM or other daemons that implement XDMCP.

GDM only supports the MIT-MAGIC-COOKIE-1 authentication system. Normally little is gained from the other schemes, and no effort has been made to implement them so far. Because of this the cookies go over the wire as clear text, and thus you should be careful about what network you use this on. That is, you should be careful about through where your XDMCP connection is going. Note that obviously if snooping is possible, then the attacker could just snoop your password as you log in, so a better XDMCP authentication wouldn't help you much anyway. If snooping is possible and undesirable, then you had better use ssh for tunneling an X connection anyway rather then using GDM's XDMCP. You could think of XDMCP as a sort of graphical telnet, having the same security issues.

On the upside, GDM's random number generation is very anal and GDM goes to extraordinary measures to truly get a 128 bit random number, using hardware random number generators if available, plus the current time (in microsecond precision), a 20 byte array of pseudorandom numbers, process pid's, plus other random information (possibly using `/dev/audio` or `/dev/mem` if hardware random generators are not available) to create a large buffer and then run MD5 digest on this. Obviously, all this work is wasted if you send this cookie over an open network or store it on an NFS directory (see `UserAuthDir` configuration key). So be careful about where you use remote X display.

Since it is fairly easy to do denial of service attacks on the XDMCP service, GDM incorporates a few features to guard against attacks. Please read the XDMCP reference section below for more information.

Even though GDM tries to outsmart potential attackers, it is still advised that you block UDP port 177 on your firewall unless you really need it. GDM guards against DoS attacks, but the X protocol is still inherently insecure and should only be used in controlled environments. Also each remote connection takes up lots of resources, so it is much easier to to DoS an XDMCP server then say a webserver.

In addition to UDP port 177, you should also block all the X server ports (TCP ports 6000 + display number) on the firewall as well. Do note that various places in GDM will use display numbers 20 and higher (for example the on demand server stuff). X is not a very safe protocol for leaving on the net, and XDMCP is even less safe.

Even though your display is protected by cookies the XEvents and thus the keystrokes typed when entering passwords will still go over the wire in clear text. It is trivial to capture these. You should also be aware that cookies, if placed on an NFS mounted directory, are prone to eavesdropping too. In case of NFS home directories you should really use the `UserAuthDir` and set it to some local temporary directory.

XDMCP is primarily useful for running thin clients such as in terminal labs. Those thin clients will only ever need the network to access the server, and so it seems like the best policy securitywise to have those thin clients on a separate network that cannot be accessed by the outside world, and can only connect to the server. The only point from which you need to access outside is the server.

XDMCP Access Control

XDMCP access control is done using TCP wrappers. It is possible to compile GDM without TCP wrappers however, so you should test your configuration to see if they work.

You should use the daemon name `gdm` in the `/etc/hosts.allow` and `/etc/hosts.deny` files. For example to deny computers from `.evil.domain` from logging in, then add

```
gdm: .evil.domain
```

to `/etc/hosts.deny`. You may also need to add

```
gdm: .your.domain
```

to your `/etc/hosts.allow` if you normally disallow all services from all hosts. See the hosts.allow(5)[3] man page for details.

Even though GDM now tries very hard to ignore things coming from banned hosts you should not rely on the TCP Wrappers for complete protection. It is really best to block UDP port 177 (and all the X ports which are TCP ports 6000 + the display number of course) on your firewall.

The Standard Greeter

The Standard Greeter is the default graphical user interface that is presented to the user. The greeter contains a menu at the top, an optional face browser, an optional logo and a text entry widget.

The text entry field is used for entering logins, passwords, passphrases etc. `gdmlogin` is controlled by the underlying daemon and is basically stateless. The daemon controls the greeter through a simple protocol where it can ask the greeter for a text string with echo turned on or off. Similarly, the daemon can change the label above the text entry widget to correspond to the value the authentication system wants the user to enter.

The menu bar in the top of the greeter enables the user to select the requested session type/desktop environment, select an appropriate locale/language and optionally shutdown/reboot/suspend the machine, configure GDM (given the user knows the root password), change the GTK+ theme, or start an XDMCP chooser.

Optionally the greeter can provide a face browser containing icons for all the users on a system. The icons can be installed globally by the sysadmin or in the users' home directories. If installed globally they should be in the `<share>/faces/` directory (though this can be configured with the `GlobalFaceDir` configuration option) and the filename should be the name of the user, optionally with a `.png` appended.

The users can place their icons in a file called `~/.face`, and they can use the program `gdmphotosetup` to graphically configure this.

Face icons placed in the global face directory must be readable to the gdm user. However, the daemon, proxies user pictures to the greeter and thus those don't have be be readable by the gdm user, but root.

Please note that loading and scaling face icons located in user home directories can be a very time consuming task. Especially on large systems or systems running NIS. The browser feature is only intended for systems with relatively few users. Also if home directories are on an on demand mounted filesystem like AFS, then GDM may mount all the home directories just to check for pictures if the face browser is on. GDM will try to give up after 5 seconds of activity however and only display the users whoose pictures it has gotten so far.

To filter out unwanted user names in the browser, an exclude option is implemented. The greeter will automatically ignore usernames listed in the `Exclude` statement in the config file, and furthermore exclude users whose UIDs are lower then `MinimalUID`.

When the browser is turned on, valid usernames on the machine are inherently exposed to a potential intruder. This may be a bad idea if you don't know who can

get to a login screen. This is especially true if you run XDMCP. However you should never run XDMCP on an open network anyway.

The greeter can optionally display a logo in the login window. The image must be in a format readable to the gdk-pixbuf library (GIF, JPG, PNG, TIFF, XPM and possibly others), and it must be readable to the gdm user. See the `Logo` option in the reference section below for details.

The Graphical Greeter

The Graphical Greeter is a greeter interface that takes up the whole screen and is very themable. Themes can be selected and new themes can be installed by the Configuration program or by setting the `GraphicalTheme` configuration key.

The look and feel of this greeter is really controlled by the theme and so the user interface elements that are present may be different. The only thing that must always be present is the text entry field as described above in the Standard Greeter.

You can always get a menu of available actions by pressing the F10 key. This can be useful if the theme doesn't provide certain buttons when you really wish to do some action.

Logging

GDM itself will use syslog to log errors or status. It can also log debugging information, but this is not generally useful unless something is very wrong, and this must be enabled in the configuration file.

Output from the various X servers is stored in the GDM log directory, which is configurable, but is usually `<var>/log/gdm/`. The output from the session can be found in a file called `<display>.log`. Four older files are also stored with `.1` through `.4` appended. These will be rotated as new sessions on that display are started. You can use these logs to view what the X server said when it started up.

The output from the user session is redirected to `~/.xsession-errors` before even the `PreSession` script is started. So it is not really necessary to redirect this again in the session setup script. As is usually done. If the user session lasted less then 10 seconds, GDM assumes that the session crashed and allows the user to view this file in a dialog before returning to the login screen. This way the user can view the session errors from the last session and correct the problem this way.

You can suppress the 10 second warning by returning code 66 from the `Xsession`script or from your session binary (the default `Xsession` script propagates those codes back). This is useful if you have some sort of special logins for which it is not an error to return less then 10 seconds later, or if you setup the session to already display some error message and the gdm message would be confusing and redundant.

The session output is piped through the GDM daemon and so the `~/.xsession-errors` file is capped at about 200 kilobytes by GDM to prevent a possible denial of service attack on the session. An app could perhaps on reading some wrong data print out warnings or errors on the stderr or stdout. This could perhaps fill up the users home directory who would then have to log out and log back in to clear this. This could be especially nasty if quotas are set. GDM also correctly traps the XFSZ signal and stops writing the file, which would lead to killed sessions if the file was redirected in the old fashioned way from the script.

Note that some distributors seem to override the `~/.xsession-errors` redirection and do it themselves in their own Xsession script (set by the `BaseXsession` configuration key) which means that GDM will not be able to trap the output and cap this file. You also lose output from the `PreSession` script which can make debugging things harder to figure out as perhaps useful output of what is wrong will not be

printed out. See the description of the BaseXsession configuration key for more information, especially on how to handle multiple display managers using the same script.

Note that if the session is a failsafe session, or if GDM can't open this file for some reason, then a fallback file will be created in the /tmp directory named /tmp/xses-<user>.XXXXXX where the XXXXXX are some random characters.

If you run a system with quotas set, it would be good to delete the ~/.xsession-errors in the PostSession script. Such that this log file doesn't unnecessarily stay around.

Security and the GDM User

The GDM daemon normally runs as root, as does the slave. However GDM should also have a dedicated user id and a group id which it uses for its graphical interfaces such as gdmgreeter and gdmlogin. You can choose the name of this user and group in the [daemon] section of the configuration file.

The GDM user, and group, which are normally just gdm should not be user or group of any particular privilage. The reason for using them is to have the user interface run as a user without privilages so that in the unlikely case that someone finds a weakness in the GUI, they cannot access root on the machine.

It should however be noted that the GDM user and group have some privilages that make them somewhat dangerous. For one they have access to the server authorization directory (the ServAuthDir), which contains all the X server authorization files and other private information. This means that someone who gains the GDM user/group privilages can then connect to any session. So you should not, under any circumstances, make this some user/group which may be easy to get access to, such as the user nobody.

The server authorization directory (the ServAuthDir) is used for a host of random internal data in addition to the X server authorization files, and the naming is really a relic of history. GDM daemon enforces this directory to be owned by root.gdm with the permissions of 1770. This way, only root and the GDM group have write access to this directory, but the GDM group cannot remove the root owned files from this directory, such as the X server authorization files.

GDM by default doesn't trust the server authorization directory and treats it in the same way as the temporary directory with respect to creating files. This way someone breaking the GDM user cannot mount attacks by creating links in this directory. Similarly the X server log directory is treated safely, but that directory should really be owned and writable only by root.

Anybody found not using a dedicated user for GDM should be whacked over the head with a large, blunt, heavy and rusty object, although the rusty requirement may be dropped if there is not enough time to have the object develop rust.

Configuration

This section will cover the configuration of GDM and the format of the configuration file. However you can use the gdmsetup binary to configure GDM from a graphical environment. The configuration program does not however let you configure every aspect of GDM however, so if the setup program does not cover your needs you may find information in this section.

The configuration files, and especially the gdm.conf file, contain lots of useful comments, so also read these when changing your setup.

Some keys in the configuration file as shipped are commented out while others are set. This is done so that defaults can be easily changed in the future for some keys. If you wish to set such a key you must first remove the leading hash mark that marks it as a comment.

The configuration files for GDM are located in the `<etc>/gdm/` directory. And some which can be shared among other display managers are located in the `<etc>/dm/` directory.

This is a listing of the config directory contents:

```
Init/
PostLogin/
PostSession/
PreSession/
modules/
gdm.conf
factory-gdm.conf
locale.alias
Xsession
XKeepsCrashing
```

`gdm.conf` is the main GDM configuration file. The options will be described later in this section. `factory-gdm.conf` is the configuration file as shipped with the daemon. This can be useful if you wish to revert to the default configuration.

`locale.alias` is a file which looks much like the system locale alias but in fact it is not the same. These are the languages that are available on your system. All the languages are still tested to see if they actually exist before presenting them to the user.

`Xsession` is a script which sets up a user session and then executes the users choice of session.

`XKeepsCrashing` is a script which gets run when the X server keeps crashing and we cannot recover. The shipped default script will work with most Linux distributions and can run the X configuration program provided the person on the console knows the root password.

`gdm.conf` is configuration file for both `gdm`, `gdmgreeter`, `gdmlogin`, and `gdmchooser` since a lot of parameters overlap.

Accessibility modules are configured in the `modules/` subdirectory, and are a separate topic. Read the default files provided, they have adequate documentation. Again normally the default install is given in the files with `factory` in their name, and those files are not read, they are just there for you so you can always revert to default config.

The remaining configuration is done by dropping scripts in the subdirectories of the `<etc>/gdm/` folder or dropping `.desktop`-style files in `/etc/X11/sessions/`. The latter is also read by KDM for common configuration. Next the directory `<share>/gdm/BuiltInSessions/` is read for GDM specific built in sessions (KDM hardcodes these at time of this writing). Also the default setup will also read `<share>/xsessions/` (which should be `/usr/share/xsessions/` if you really wish to cooperate with KDM) where desktop packages can install their session files. The directories under the `/etc` should be reserved for configuration. This approach makes it easy for package management systems to install window managers and different session types without requiring the sysadmin to edit files. See the `SessionDesktopDir` configuration key for changing the paths. It used to be that GDM stored its built in sessions in `<etc>/dm/Sessions/` but this is now deprecated as of 2.5.90.0. Note that prior to version 2.4.4.2 only the `<etc>/dm/Sessions/` was being read.

A session can be disabled (if it was installed in `/usr/share/xsessions/`) by adding an identically named `.desktop` to one of the directories earlier in the path (likely `/etc/X11/sessions`) and using `Hidden=true` in that file.

The Script Directories

In this section we will explain the `Init`, `PostLogin`, `PreSession` and `PostSession` directories as they are very similar.

When the X server has been successfully started, GDM will try to run the script called `Init/<displayname>`. I.e. `Init/:0` for the first local display. If this file is not found, GDM will attempt to to run `Init/<hostname>`. I.e. `Init/somehost`. If this still is not found, GDM will try `Init/XDMCP` for all XDMCP logins or `Init/Flexi` for all on demand flexible servers. If none of the above were found, GDM will run `Init/Default`. The script will be run as root and GDM blocks until it terminates. Use the `Init/*` script for programs that are supposed to run alongside with the GDM login window. xconsole for instance. Commands to set the background etc. goes in this file too.

It is up to the sysadmin to decide whether clients started by the Init script should be killed before starting the user session. This is controlled with the `KillInitClients` option in `gdm.conf`.

When the user has been successfully authenticated GDM tries the scripts in the `PostLogin` directory in the same manner as for the `Init` directory. This is done before any session setup is done, and so this would be the script where you might setup the home directory if you need to (though you should use the `pam_mount` module if you can for this). You have the `$USER` and `$DISPLAY` environment variables set for this script, and again it is run as root. The script should return 0 on success as otherwise the user won't be logged in. This is not true for failsafe session however.

After the user session has been setup from the GDM side of things, GDM will run the scripts in the `PreSession` directory, again in the same manner as the `Init` directory. Use this script for local session management or accounting stuff. The `$USER` environment variable contains the login of the authenticated user and `$DISPLAY` is set to the current display. The script should return 0 on success. Any other value will cause GDM to terminate the current login process. This is not true for failsafe sessions however. Also `$X_SERVERS` environmental variable is set and this points to a fake generated x servers file for use with the sessreg accounting program.

After this the base `Xsession` script is run with the selected session executable as the first argument. This is run as the user, and really this is the user session. The available session executables are taken from the `Exec=` line in the `.desktop` files in the path specified by `SessionDesktopDir`. Usually this path is `/etc/X11/sessions/:<etc>/dm/Sessions:/usr/share/xsessions/`. The first found file is used. The user either picks from these sessions or GDM will look inside the file `~/.dmrc` for the stored preference.

This script should really load the users profile and generally do all the voodoo that is needed to launch a session. Since many systems reset the language selections done by GDM, GDM will also set the `$GDM_LANG` variable to the selected language. You can use this to reset the language environmental variables after you run the users profile. If the user elected to use the system language, then `$GDM_LANG` is not set.

When the user terminates his session, the `PostSession` script will be run. Again operation is similar to `Init`, `PostLogin` and `PreSession`. Again the script will be run with root privileges, the slave daemon will block and the `$USER` environment variable will contain the name of the user who just logged out and `$DISPLAY` will be set to the display the user used, however note that the X server for this display may already be dead and so you shouldn't try to access it. Also `$X_SERVERS` environmental variable is set and this points to a fake generated x servers file for use with the sessreg accounting program.

Note that the `PostSession` script will be run even when the display fails to respond due to an I/O error or similar. Thus, there is no guarantee that X applications will work during script execution.

Except for the `Xsession` script all of these scripts will also have the environment variable `$RUNNING_UNDER_GDM` set to `yes`, so that you could perhaps use similar scripts

for different display managers. The `Xsession` will always have the `$GDMSESSION` set to the basename of the session that the user chose to run without the `.desktop` extension. In addition `$DESKTOP_SESSION` is also set to the same value and in fact this will also be set by KDM in future versions.

Neither of the `Init`, `PostLogin`, `PreSession` or `PostSession` scripts are necessary and can be left out. The `Xsession` script is however required as well as at least one session `.desktop` file.

The Configuration File - `gdm.conf`

The daemon and the accompanying utilities share a common configuration file: `<etc>/gdm/gdm.conf`.

The configuration file is divided into sections each containing variables that define the behavior for a specific part of the GDM suite. The file is fairly well commented as well.

`gdm.conf` follows the standard `.ini` style configuration file syntax. Keywords in brackets define sections, strings before an equal sign (=) are variables and the data after equal sign represents their value. Empty lines or lines starting with the hash mark (#) are ignored. The graphical configurator will try to preserve both comments (lines with a hash mark) and the overall structure of the file so you can intermix using the GUI or hand editing the configuration file.

Daemon Configuration

[daemon]

AddGtkModules

`AddGtkModules=false`

If true, then enables gdmgreeter/gdmlogin to be launched with additional Gtk+ modules. This is useful when extra features are required such as accessible login. Note that only "trusted" modules should be used to minimize security issues.

Usually this is used for accessibility modules. The modules which are loaded are specified with the `GtkModulesList` key.

AlwaysRestartServer

`AlwaysRestartServer=false`

If true, then gdm never tries to reuse existing X servers by reinitializing them. It will just kill the existing server and start over. Normally, just reinitializing is a nicer way to go but if the X server memory usage keeps growing this may be a safer option.

AutomaticLoginEnable

`AutomaticLoginEnable=false`

If the user given in AutomaticLogin should be logged in upon first bootup. No password will be asked. This is useful for single user workstations where local console security is not an issue. Also could be useful for public terminals, although there see `TimedLogin`.

AutomaticLogin

`AutomaticLogin=`

This user should be automatically logged in on first bootup. AutomaticLoginEnable must be true and this must be a valid user for this to happen. "root" can never be autologged in however and gdm will just refuse to do it even if you set it up.

The following control chars are recognized within the specified name:

%% — the '%' character

%d — display's name

%h — display's hostname

Alternatively, the name may end with a vertical bar `|`, the pipe symbol. The name is then used as a program to execute which returns the desired username on standard output. If an empty or otherwise invalid username is returned, automatic login is not performed. This feature is typically used when several remote displays are used as internet kiosks, with a specific user to automatically login for each display.

BaseXsession

`BaseXsession=<etc>/gdm/Xsession`

This is the base X session file. When a user logs in, this script will be run with the selected session as the first argument. The selected session will be the `Exec=` from the `.desktop` file of the session.

If you wish to use the same script for several different display managers, and wish to have some of the script run only for GDM, then you can check the presence of the `GDMSESSION` environmental variable. This will always be set to the basename of `.desktop` (without the extension) file that is being used for this session, and will only be set for GDM sessions. Previously some scripts were checking for `GDM_LANG`, but that is only set when the user picks a non-system default language.

This script should take care of doing the "login" for the user and so it should source the `/etc/profile` and friends. The standard script shipped with GDM sources the files in this order: `/etc/profile` then `~/.profile` then `/etc/xprofile` and finally `~/.xprofile`. Note that different distributions may change this however. Sometimes users personal setup will be in `~/.bash_profile`, however broken that is.

Chooser

`Chooser=<bin>/gdmchooser`

Full path and name of the chooser executable followed by optional arguments.

Configurator

`Configurator=<bin>/gdmsetup --disable-sound --disable-crash-dialog`

The pathname to the configurator binary. If the greeter `ConfigAvailable` option is set to true then run this binary when somebody chooses Configuration from the Actions menu. Of course GDM will first ask for root password however. And it will never allow this to happen from a remote display.

ConsoleCannotHandle

`ConsoleCannotHandle=am,ar,az,bn,el,fa,gu,hi,ja,ko,ml,mr,pa,ta,zh`

These are the languages that the console cannot handle because of font issues. Here we mean the text console, not X. This is only used when there are errors to report and we cannot start X.

DefaultPath

`DefaultPath=/bin:/usr/bin:/usr/bin/X11:/usr/local/bin`

Specifies the path which will be set in the user's session.

DefaultSession

`DefaultSession=gnome.desktop`

The session that is used by default if the user does not have a saved preference and has picked 'Last' from the list of sessions. Note that 'Last' need not be displayed, see the `ShowLastSession` key.

DisplayInitDir

`DisplayInitDir=<etc>/gdm/Init`

Directory containing the display init scripts. See the "The Script Directories" section for more info.

DisplayLastLogin

`DisplayLastLogin=true`

If true then the last login information is printed to the user before being prompted for password. While this gives away some info on what users are on a system, it on the other hand should give the user an idea of when they logged in and if it doesn't seem kosher to them, they can just abort the login and contact the sysadmin (avoids running malicious startup scripts). This was added in version 2.5.90.0.

This is for making GDM conformant to CSC-STD-002-85, although that is purely theoretical now. Someone should read that spec and ensure that this actually conforms (in addition to other places in GDM). See `http://www.radium.ncsc.mil/tpep/library/rainbow/CSC-STD-002-85.html` for more info.

DoubleLoginWarning

`DoubleLoginWarning=true`

If true, GDM will warn the user if they are already logged in on another virtual terminal. On systems where GDM supports checking the X virtual terminals, GDM will let the user switch to the previous login virtual terminal instead of logging in.

FailsafeXServer

`FailsafeXServer=`

An X command line in case we can't start the normal X server. should probably be some sort of a script that runs an appropriate low resolution server that will just work. This is tried before the `XKeepsCrashing` script is run.

FirstVT

`FirstVT=7`

On systems where GDM supports automatic VT (virtual terminal) allocation, this is the first vt to try. Usually standard text logins are run on the lower vts. See also `VTAllocation`.

FlexibleXServers

`FlexibleXServers=5`

The maximum number of allowed flexible servers. These are servers that can be run using the `/tmp/.gdm_socket` socket connection. This is used for both full servers and for Xnest servers.

FlexiReapDelayMinutes

`FlexiReapDelayMinutes=5`

After how many minutes of inactivity at the login screen should a flexi server be reaped. This is only in effect before a user logs in. Also it does not affect the Xnest flexiservers. To turn off this behaviour set this value to 0. This was added in version 2.5.90.0.

Greeter

`Greeter=<bin>/gdmlogin`

Full path and name of the greeter executable followed by optional arguments. This is the greeter used for all servers except for the XDMCP remote servers. See also `RemoteGreeter`

Group

`Group=gdm`

The group name under which `gdmlogin`, `gdmgreeter`, `gdmchooser` and the internal failsafe GTK+ dialogs are run. Also see `User`. This user will have access to all the X authorization files, and perhaps to other internal GDM data and it should not therefore be a user such as nobody, but rather a dedicated user. The `ServAuthDir` is owned by this group. The ownership and permissions of `ServAuthDir` should be `root.gdm` and 1770.

GtkModulesList

`GtkModulesList=module-1:module-2:...`

A colon separated list of Gtk+ modules that gdmgreeter/gdmlogin will be invoked with if `AddGtkModules` is true. The format is the same as the standard Gtk+ module interface.

HaltCommand

`HaltCommand=/sbin/shutdown -h now`

Full path and arguments to command to be executed when user selects Shutdown from the Actions menu. This can be a ';' separated list of commands to try. If a value is missing, the shutdown command is not available. Note that the default for this value is not empty so to disable shutdown you must set this explicitly to an empty value.

KillInitClients

`KillInitClients=true`

Determines whether GDM should kill X clients started by the init scripts when the user logs in.

LogDir

`LogDir=<var>/log/gdm`

Directory containing the log files for the individual displays. By default this is the same as the ServAuthDir.

PidFile

`PidFile=<var>/run/gdm.pid`

Name of the file containing the `gdm` process id.

PostLoginScriptDir

`PostLoginScriptDir=<etc>/gdm/PostLogin`

Directory containing the scripts run right after the user logs in, but before any session setup is done. See the "The Script Directories" section for more info.

PostSessionScriptDir

`PostSessionScriptDir=<etc>/gdm/PostSession`

Directory containing the scripts run after the user logs out. See the "The Script Directories" section for more info.

PreSessionScriptDir

`PreSessionScriptDir=<etc>/gdm/PreSession`

Directory containing the scripts run before the user logs in. See the "The Script Directories" section for more info.

RebootCommand

`RebootCommand=/sbin/shutdown -r now`

Full path and optional arguments to the program to be executed when user selects Reboot from the Actions menu. This can be a ';' separated list of commands to try. If missing, the reboot command is not available. Note that the default for this value is not empty so to disable reboot you must set this explicitly to an empty value.

RemoteGreeter

```
RemoteGreeter=<bin>/gdmlogin
```

Full path and name of the greeter executable followed by optional arguments. This is used for all remote XDMCP sessions. It is useful to have the less graphically demanding greeter here if you use the Graphical Greeter for your main greeter. See also the `Greeter` key.

RootPath

```
RootPath=/sbin:/usr/sbin:/bin:/usr/bin:/usr/bin/X11:/usr/local/bin
```

Specifies the path which will be set in the root's session and the {Init,PostLogin,PreSession,PostSession} scripts executed by GDM.

ServAuthDir

```
ServAuthDir=<var>/gdm
```

Directory containing the X authentication files for the individual displays. Should be owned by `root.gdm` with permissions 1770, where `gdm` is the GDM group as defined by the `Group` option. That is should be owned by root, with `gdm` group having full write permissions and the directory should be sticky and others should have no permission to the directory. This way the gdm user can't remove files owned by root in that directory, while still being able to write its own files there. GDM will attempt to change permissions for you when it's first run if the permissions are not the above. This directory is also used for other private files that the daemon needs to store. Other users should not have any way to get into this directory and read/change it's contents. Anybody who can read this directory can connect to any display on this machine.

SessionDesktopDir

```
SessionDesktopDir=/etc/X11/sessions/:<etc>/dm/Sessions/:/usr/share/xsessions/
```

Directory containing the `.desktop` files which are the available sessions on the system. Since 2.4.4.2 this is treated like a PATH type variable and the first file found is used.

SoundProgram

```
SoundProgram=/usr/bin/play
```

Program to use when playing a sound. Currently used for playing the login sound, see the `SoundOnLoginFile` key. Supported since 2.5.90.0.

StandardXServer

```
StandardXServer=/usr/X11R6/bin/X
```

Full path and arguments to the standard X server command. This is used when gdm cannot find any other definition, and it's used as the default and failsafe fallback in a number of places. This should be able to run some sort of X server.

SuspendCommand

`SuspendCommand=`

Full path and arguments to command to be executed when user selects Suspend from the Actions menu. If empty there is no such menu item. Note that the default for this value is not empty so to disable suspend you must set this explicitly to an empty value.

TimedLoginEnable

`TimedLoginEnable=false`

If the user given in `TimedLogin` should be logged in after a number of seconds (set with `TimedLoginDelay`) of inactivity on the login screen. This is useful for public access terminals or perhaps even home use. If the user uses the keyboard or browses the menus, the timeout will be reset to `TimedLoginDelay` or 30 seconds, whichever is higher. Note that no password will be asked for this user so you should be careful.

TimedLogin

`TimedLogin=`

This is the user that should be logged in after a specified number of seconds of inactivity. This can never be "root" and gdm will refuse to log in root this way. The same features as for `AutomaticLogin` are supported. The same control chars and piping to a program are supported.

TimedLoginDelay

`TimedLoginDelay=30`

This is the delay before the `TimedLogin` user will be logged in. It must be greater then or equal to 10.

User

`User=gdm`

The username under which `gdmlogin`, `gdmgreeter`, `gdmchooser` and the internal failsafe GTK+ dialogs are run. Also see `Group`. This user will have access to all the X authorization files, and perhaps to other internal GDM data and it should not therefore be a user such as nobody, but rather a dedicated user.

UserAuthDir

`UserAuthDir=`

The directory where user's `.Xauthority` file should be saved. When nothing is specified the user's home directory is used. This is tilde expanded so you can set it to things like: `~/authdir/`.

If you do not use the tilde expansion, then the filename created will be random, like in `UserAuthFBDir`. This way many users can have the same authentication directory. For example you might want to set this to `/tmp` when user has the home directory on NFS, since you really don't want cookie files to go over the wire. The users should really have write privilages to this directory, and this directory should really be sticky and all that, just like the `/tmp` directory.

Normally if this is the users home directory GDM will still refuse to put cookies there if it thinks it is NFS (by testing root-squashing). This can be changed by setting `NeverPlaceCookiesOnNFS` in the `[security]` section to false.

UserAuthFBDir

`UserAuthFBDir=/tmp`

If GDM fails to update the user's `.Xauthority` file a fallback cookie is created in this directory.

UserAuthFile

`UserAuthFile=.Xauthority`

Name of the file used for storing user cookies.

VTAllocation

`VTAllocation=true`

On systems where GDM supports automatic VT (virtual terminal) allocation (currently Linux and FreeBSD only), you can have GDM automatically append the vt argument to the X server executable. This way races that come up from each X server managing it's own vt allocation can be avoided. See also `FirstVT`.

XKeepsCrashing

`XKeepsCrashing=<etc>/gdm/XKeepsCrashing`

A script to run in case X keeps crashing. This is for running An X configuration or whatever else to make the X configuration work. See the script that came with the distribution for an example. The distributed `XKeepsCrashing` script is tested on Red Hat, but may work elsewhere. Your system integrator should make sure this script is up to date for your particular system.

In case `FailsafeXServer` is setup, that will be tried first. and this only used as a backup if even that server keeps crashing.

Xnest

`Xnest=/usr/bin/X11/Xnest`

The full path and arguments to the Xnest command. This is used for the flexible Xnest servers. This way the user can start new login screens in a nested window. Of course you must have the Xnest server from your X server packages installed for this to work.

Security Options

[security]

AllowRoot

`AllowRoot=true`

Allow root (privileged user) to log in through GDM. Set this to false if you want to disallow such logins.

On systems that support PAM, this parameter is not as useful as you can use PAM to do the same thing, and in fact do even more. However it is still followed, so you should probably leave it true for PAM systems.

AllowRemoteRoot

`AllowRemoteRoot=true`

Allow root (privileged user) to log in remotely through GDM. Set this to false if you want to disallow such logins. Remote logins are any logins that come in through the xdmcp.

On systems that support PAM, this parameter is not as useful as you can use PAM to do the same thing, and in fact do even more. However it is still followed, so you should probably leave it true for PAM systems.

AllowRemoteAutoLogin

`AllowRemoteAutoLogin=false`

Allow the timed login to work remotely. That is, remote connections through XDMCP will be allowed to log into the "TimedLogin" user by letting the login window time out, just like the local user on the first console.

Note that this can make a system quite insecure, and thus is off by default.

CheckDirOwner

`CheckDirOwner=true`

By default GDM checks the ownership of the home directories before writing to them, this prevents security issues in case of bad setup. However in some instances home directories will be owned by a different user and in this case it is necessary to turn this option on. You will also most likely have to turn the `RelaxPermissions` key to at least value 1 since in such a scenario home directories are likely to be group writable. Supported since 2.6.0.4.

DisallowTCP

`DisallowTCP=true`

If true, then always append `-nolisten tcp` to the command line of local X servers, thus disallowing TCP connection. This is useful if you do not care for allowing remote connections, since the X protocol could really be potentially a security hazard to leave open, even though no known security problems exist.

NeverPlaceCookiesOnNFS

`NeverPlaceCookiesOnNFS=true`

Normally if this is true (which is by default), GDM will not place cookies into the users home directory if this directory is on NFS. Well, GDM will consider any filesystem with root-squashing an NFS filesystem. Sometimes however the remote file system can have root squashing and be safe (perhaps by using encryption). In this case set this to 'false'. Note that this option appeared in version 2.4.4.4 and is ignored in previous versions.

RelaxPermissions

`RelaxPermissions=0`

By default GDM ignores files and directories writable to other users than the owner.

Changing the value of RelaxPermissions makes it possible to alter this behavior:

0 - Paranoia option. Only accepts user owned files and directories.

1 - Allow group writable files and directories.

2 - Allow world writable files and directories.

RetryDelay

`RetryDelay=1`

The number of seconds GDM should wait before reactivating the entry field after a failed login.

UserMaxFile

`UserMaxFile=65536`

GDM will refuse to read/write files bigger than this number (specified in bytes).

In addition to the size check GDM is extremely picky about accessing files in user directories. It will not follow symlinks and can optionally refuse to read files and directories writable by other than the owner. See the `RelaxPermissions` option for more info.

XDCMP Support

[xdmcp]

DisplaysPerHost

`DisplaysPerHost=1`

To prevent attackers from filling up the pending queue, GDM will only allow one connection for each remote machine. If you want to provide display services to machines with more than one screen, you should increase the `DisplaysPerHost` value accordingly.

Note that the number of connections from the local machine is unlimited. Only remote connections are limited by this number.

Enable

`Enable=false`

Setting this to true enables XDMCP support allowing remote displays/X terminals to be managed by GDM.

`gdm` listens for requests on UDP port 177. See the Port option for more information.

If GDM is compiled to support it, access from remote displays can be controlled using the TCP Wrappers library. The service name is `gdm`

You should add

`gdm:.my.domain`

to your `/etc/hosts.allow`, depending on your TCP Wrappers configuration. See the hosts.allow(5)[4] man page for details.

Please note that XDMCP is not a particularly secure protocol and that it is a good idea to block UDP port 177 on your firewall unless you really need it.

HonorIndirect

`HonorIndirect=true`

Enables XDMCP INDIRECT choosing (i.e. remote execution of `gdmchooser`) for X-terminals which don't supply their own display browser.

MaxPending

`MaxPending=4`

To avoid denial of service attacks, GDM has fixed size queue of pending connections. Only MaxPending displays can start at the same time.

Please note that this parameter does *not* limit the number of remote displays which can be managed. It only limits the number of displays initiating a connection simultaneously.

MaxPendingIndirect

`MaxPendingIndirect=4`

GDM will only provide `MaxPendingIndirect` displays with host choosers simultaneously. If more queries from different hosts come in, the oldest ones will be forgotten.

MaxSessions

`MaxSessions=16`

Determines the maximum number of remote display connections which will be managed simultaneously. I.e. the total number of remote displays that can use your host.

MaxWait

`MaxWait=30`

When GDM is ready to manage a display an ACCEPT packet is sent to it containing a unique session id which will be used in future XDMCP conversations.

GDM will then place the session id in the pending queue waiting for the display to respond with a MANAGE request.

If no response is received within MaxWait seconds, GDM will declare the display dead and erase it from the pending queue freeing up the slot for other displays.

MaxWaitIndirect

`MaxWaitIndirect=30`

The MaxWaitIndirect parameter determines the maximum number of seconds between the time where a user chooses a host and the subsequent indirect query

where the user is connected to the host. When the timeout is exceeded, the information about the chosen host is forgotten and the indirect slot freed up for other displays. The information may be forgotten earlier if there are more hosts trying to send indirect queries then `MaxPendingIndirect`.

Port

`Port=177`

The UDP port number `gdm` should listen to for XDMCP requests. Don't change this unless you know what you are doing.

PingIntervalSeconds

`PingIntervalSeconds=15`

Interval in which to ping the X server in seconds. If the X server doesn't return before the next time we ping it, the connection is stopped and the session ended. This is a combination of the XDM PingInterval and PingTimeout, but in seconds.

Note that GDM in the past used to have a `PingInterval` configuration key which was also in minutes. For most purposes you'd want this setting to be lower then one minute however since in most cases where XDMCP would be used (such as terminal labs), a lag of more than 15 or so seconds would really mean that the terminal was turned off or rebooted and you would want to end the session.

Willing

`Willing=<etc>/gdm/Xwilling`

When the server sends a WILLING packet back after a QUERY it sends a string that gives the current status of this server. The default message is the system ID, but it is possible to create a script that displays customized message. If this script doesn't exist or this key is empty the default message is sent. If this script succeeds and produces some output, the first line of it's output is sent (and only the first line). It runs at most once every 3 seconds to prevent possible denial of service by flooding the server with QUERY packets.

Common GUI Configuration Options

[gui]

AllowGtkThemeChange

`AllowGtkThemeChange=true`

If to allow changing the GTK+ (widget) theme from the greeter. Currently this only affects the standard greeter as the graphical greeter does not yet have this ability. The theme will stay in effect on this display until changed and will affect all the other windows that are put up by GDM. Supported since 2.5.90.2.

GtkRC

`GtkRC=`

Path to a `gtkrc` to read when GDM puts up a window. You should really now use the `GtkTheme` key for just setting a theme.

GtkTheme

```
GtkTheme=Default
```

A name of an installed theme to use by default. It will be used in the greeter, chooser and all other GUI windows put up by GDM. Supported since 2.5.90.2.

GtkThemesToAllow

```
GtkThemesToAllow=all
```

Comma separated list of themes to allow. These must be the names of the themes installed in the standard locations for gtk themes. You can also specify 'all' to allow all installed themes. This is related to the `AllowGtkThemeChange` key. Supported since 2.5.90.2.

MaxIconWidth

```
MaxIconWidth=128
```

Specifies the maximum icon width (in pixels) that the face browser will display. Icons larger than this will be scaled. This also affects icons in the XDMCP chooser.

MaxIconHeight

```
MaxIconHeight=128
```

Specifies the maximum icon height (in pixels) that the face browser will display. Icons larger than this will be scaled. This also affects icons in the XDMCP chooser.

Greeter Configuration

[greeter]

BackgroundColor

```
BackgroundColor=#007777
```

If the BackgroundType is 2, use this color in the background of the greeter. Also use it as the back of transparent images set on the background and if the BackgroundRemoteOnlyColor is set and this is a remote display. This only affects the Standard Greeter.

BackgroundImage

```
BackgroundImage=somefile.png
```

If the BackgroundType is 1, then display this file as the background in the greeter. This only affects the Standard Greeter.

BackgroundProgram

```
BackgroundProgram=/usr/bin/xeyes
```

If set this program will be run in the background while the login window is being displayed. Note that not all programs will run this way, since gdm does not usually have a home directory. You could set up home directory for the gdm user if you wish to run applications which require it. This only affects the Standard Greeter.

BackgroundRemoteOnlyColor

```
BackgroundRemoteOnlyColor=true
```

On remote displays only set the color background. This is to make network load lighter. The `BackgroundProgram` is also not run. This only affects the Standard Greeter.

BackgroundScaleToFit

```
BackgroundScaleToFit=true
```

Scale background image to fit the screen. This only affects the Standard Greeter.

BackgroundType

```
BackgroundType=2
```

The type of background to set. 0 is none, 1 is image and 2 is color. This only affects the Standard Greeter.

Browser

```
Browser=true
```

Set to true to enable the face browser. See the "The Standard Greeter" section for more information on the face browser. This option only works for the Standard Greeter. For the Graphical Greeter, the face browser is enabled by choosing a theme which includes a face browser

ChooserButton

```
ChooserButton=true
```

If true, add a chooser button to the Actions menu that will restart the current server with a chooser. XDMCP does not need to be enabled on the local machine for this to work.

ConfigAvailable

```
ConfigAvailable=true
```

Allow the configurator to be run from the greeter. Note that the user will need to type in the root password before the configurator is run however. See the `Configurator` option in the daemon section.

DefaultFace

`DefaultFace=<share>/pixmaps/nophoto.png`

Default icon file for users without a personal picture in `~/gnome/photo`. The image must be in an gdk-pixbuf supported format and the file must be readable for the gdm user.

Exclude

`Exclude=bin,daemon,adm,lp,sync,shutdown,halt,mail,...`

Comma-separated list of usernames to exclude from the face browser. The excluded users will still be able to log in. See also `MinimalUID`.

GlobalFaceDir

`GlobalFaceDir=<share>/faces/`

Systemwide directory for face files. The sysadmin can place icons for users here without touching their homedirs. Faces are named after their users' logins.

I.e. `<GlobalFaceDir>/johndoe` would contain the face icon for the user "johndoe". No image format extension should be specified.

The face images must be stored in gdk-pixbuf supported formats and they must be readable for the GDM user.

A user's own icon file will always take precedence over the sysadmin provided one.

GraphicalTheme

`GraphicalTheme=circles`

The graphical theme that the Graphical Greeter should use. it should refer to a directory in the theme directory set by `GraphicalThemeDir`.

GraphicalThemeDir

`GraphicalThemeDir=<share>/gdm/themes/`

The directory where themes for the Graphical Greeter are installed.

InfoMsgFile

`InfoMsgFile=/path/to/infofile`

If present and /path/to/infofile specifies an existing and readable text file (e.g. /etc/infomsg.txt) the contents of the file will be displayed in a modal dialog box before the user is allowed to login. This works both with the standard and the themable greeters.

InfoMsgFont

`InfoMsgFont=fontspec`

If present and InfoMsgFile (see above) is used, this specifies the font to use when displaying the contents of the InfoMsgFile text file. For example fontspec could be Sans 24 to get a sans serif font of size 24 points. This works both with the standard and the themable greeters.

LocaleFile

`LocaleFile=<etc>/gdm/locale.alias`

File in format similar to the GNU locale format with entries for all supported languages on the system. The format is described above or in a comment inside that file.

LockPosition

`LockPosition=true`

If true the position of the login window of the Standard Greeter cannot be changed even if the title bar is turned on.

Logo

`Logo=<share>/pixmaps/gnome-logo-large.png`

Image file to display in the logo box. The file must be in an gdk-pixbuf supported format and it must be readable by the GDM user. If no file is specified the logo feature is disabled. This only affects the Standard Greeter.

MinimalUID

`MinimalUID=100`

The minimal UID that gdm should consider a user. All users with a lower UID will be excluded from the face browser. See also `Exclude`.

PositionX

`PositionX=200`

The horizontal position of the login window of the Standard Greeter.

PositionY

`PositionY=100`

The vertical position of the login window of the Standard Greeter.

Quiver

`Quiver=true`

Controls whether `gdmlogin` should shake the display when an incorrect username/password is entered. This only affects the Standard Greeter.

RemoteWelcome

`RemoteWelcome=Welcome to %n`

Controls which text to display next to the logo image in the greeter for remote XDMCP sessions. The same expansion is done here as in the `Welcome` string.

RunBackgroundProgramAlways

`RunBackgroundProgramAlways=false`

If this is true then the background program is run always, otherwise it is only run when the `BackgroundType` is 0 (None) This only affects the Standard Greeter.

SetPosition

`SetPosition=true`

If true the position of the login window of the Standard Greeter is determined by `PositionX` / `PositionY`.

ShowGnomeFailsafeSession

`ShowGnomeFailsafeSession=true`

Should the greeter show the Gnome Failsafe session in th sessions list.

ShowLastSession

`ShowLastSession=true`

Should the greeter show the 'Last' session in the session list. If this is off, then GDM is in the so called 'switchdesk' mode which for example Red Hat uses. That is, the users can't pick the last session and will just then get the default session (see `DefaultSession`) unless then pick something else for this session only. So if this is off, this really circumvents saving of the last session.

ShowXtermFailsafeSession

`ShowXtermFailsafeSession=true`

Should the greeter show the Xterm Failsafe session in the sessions list.

SoundOnLogin

`SoundOnLogin=true`

If true, the greeter will play a sound or beep when it is ready for a login. See also the `SoundOnLoginFile` key. Supported since 2.5.90.0.

SoundOnLoginFile

`SoundOnLogin=/path/to/sound.wav`

The file that will be played using the specified sound program (by default that is `/usr/bin/play`) instead of a beep when the greeter is ready for a login. See also the `SoundOnLogin` key and the `SoundProgram` key. Supported since 2.5.90.0.

SystemMenu

`SystemMenu=true`

Turns the Actions menu (which used to be called System menu) on or off. If this is off then one of the actions will be available anywhere. These actions include Shutdown, Reboot, Configure, XDMCP chooser and such. All of those can however be turned off individually. Shutdown, Reboot and Suspend can be turned off by just setting the corresponding keys to empty. Note that the actions menu

is only shown on local logins as it would not be safe or even desirable on remote logins, so you don't have to worry about remote users having any sort of console privilages.

Note that if this is off none of the actions will be available even if a theme for a graphical greeter mistakenly shows them. Also note that sometimes a graphical theme may not show all the available actions as buttons and you may have to press F10 to see the menu.

TitleBar

`TitleBar=true`

Display the title bar in the greeter. This only affects the Standard Greeter.

Use24Clock

`Use24Clock=false`

Force the use of 24 hour clock even if the locale would default to a 12 hour clock. In some locales that normally use 24 hour format (like czech, that is `cs_CZ`) this setting has no effect.

UseCirclesInEntry

`UseCirclesInEntry=false`

Use circles instead of asterisks in the password entry. This may not work with all fonts however.

Welcome

`Welcome=Welcome`

Controls which text to display next to the logo image in the standard greeter. The following control chars are supported:

%% — the '%' character

%d — display's hostname

%h — Fully qualified hostname

%m — machine (processor type)

%n — Nodename (i.e. hostname without .domain)

%r — release (OS version)

%s — sysname (i.e. OS)

This string is only used for local logins. For remote XDMCP logins we use `RemoteWelcome`.

In the Graphical Greeter the location of this text depends on the theme. Unless the theme uses the stock welcome string somewhere this string will not be displayed at all.

XineramaScreen

`XineramaScreen=0`

If the Xinerama extension is active the login window will be centered on this physical screen (use 0 for the first screen, 1 for the second...).

XDCMP Chooser Options

[chooser]

AllowAdd

`AllowAdd=true`

If true, allow the user to add arbitrary hosts to the chooser. This way the user could connect to any host that responds to XDMCP queries from the chooser.

Broadcast

`Broadcast=true`

If true, the chooser will broadcast a query to the local network and collect responses. This way the chooser will always show all available managers on the network. If you need to add some hosts not local to this network, or if you don't want to use a broadcast, you can list them explicitly in the `Hosts` key.

Multicast

`Multicast=true`

If true and IPv6 is enabled, the chooser will send a multicast query to the local network and collect responses from the hosts who have joined multicast group. If you don't want to send a multicast, you can specify IPv6 address in the `Hosts` key. The host will respond if it is listening to XDMCP requests and IPv6 is enabled there.

MulticastAddr

`MulticastAddr=ff02::1`

This is the Link-local Multicast address and is hardcoded here.

DefaultHostImage

`DefaultHostImage=<share>/pixmaps/nohost.png`

File name for the default host icon. This image will be displayed if no icon is specified for a given host. The file must be in an gdk-pixbuf supported format and it must be readable for the GDM user.

HostImageDir

`HostImageDir=<share>/hosts`

Repository for host icon files. The sysadmin can place icons for remote hosts here and they will appear in `gdmchooser`.

The file name must match the fully qualified name (FQDN) for the host. The icons must be stored in gdk-pixbuf supported formats and they must be readable to the gdm user.

Hosts

```
Hosts=host1,host2
```

The hosts which should be listed in the chooser. The chooser will only list them if they respond. This is done in addition to broadcast (if `Broadcast` is set), so you need not list hosts on the local network. This is useful if your networking setup doesn't allow all hosts to be reachable by a broadcast packet.

ScanTime

```
ScanTime=4
```

Specifies how many seconds the chooser should wait for replies to its BROADCAST_QUERY. Really this is only the time in which we expect a reply. We will still add hosts to the list even if they reply after this time.

X Server definitions

To set up X servers, you need to provide gdm with information about the installed X servers. You can have as many different definitions as you wish, each identified with a unique name. The name `Standard` is required. If you do not specify this server, gdm will assume default values for a 'Standard' server and the path given by `daemon/StandardXServer`. `Standard` is used as the default, in situations when no other server has been defined.

Servers are defined by sections named `server-` followed by the identifier of this server. This should be a simple ASCII string with no spaces. If you use the GUI configurator, it will use random words for these. These will not be user visible, they are just needed to uniquely identify the server.

[server-Standard]

name

```
name=Standard server
```

The name that will be displayed to the user.

command

```
command=/usr/bin/X11/X
```

The command to execute, with full path to the binary of the X server, and any extra arguments needed.

flexible

```
flexible=true
```

Indicates if this server is available as a choice when a user wishes to run a flexible, on demand server.

handled

```
handled=true
```

Indicates that GDM should run the login window on this server and allow a user to log in. If set to false, then GDM will just run this server and wait for it to

terminate. This can be useful to run an X terminal using GDM. When this is done you should normally also add `-terminate` to the command line of the server to make the server terminate after each session. otherwise the control of the slave will never come back to GDM and for example soft restarts won't work, since GDM assumes there is a login in progress for the entire time this server is active.

chooser

`chooser=false`

Indicates that GDM should instead of a login window run a chooser on this window and allow the user to choose which server to log into.

Local Static X Server Configuration

The static X servers are servers that are always running, when the server ever dies, it is restarted. You can have as many local static servers as you wish. Each key in the `[servers]` section corresponds to the display number of the server to run. For example normally there is only one key, which is the key `0`, which corresponds to the display `:0`.

[servers]

<display number>

`0=Standard`

Control section for local X servers. Each line indicates the local display number and the command that needs to be run to start the X server(s).

The command can either be a path to an X executable, or a name of one of the server definitions. This can be followed by some arguments that should be passed to the X server when executed.

The gdm daemon doesn't enforce the numbers to be in order or for them to be "packed". However when you use the GUI configurator, the servers will always start from 0 and go up by 1. That is, leaving no holes.

GDM will splice "`-auth <ServAuthDir>/:n.Xauth :n`", where n is the display number. Inside the command line before all other arguments before running the server.

On some systems it is necessary for gdm to know on which virtual consoles to run the X server. In this case, (if running XFree86) add "vt7" to the command line for example to run on virtual console 7. However on Linux and FreeBSD this is normally done automatically if `VTAllocation` key is set.

Normally you do not need to add a `-nolisten tcp` flag as this is added automatically for local servers when the `DisallowTCP` option is set.

Per User Configuration

There are some per user configuration settings that control how GDM behaves. Firstly there is the `~/.dmrc` file. In theory this file should be shared between GDM and KDM, so users only have to configure things once. This is a standard `.ini` style configuration file. It has one section called `[Desktop]` and can have two keys, `Session`, which is the basename of the session `.desktop` file that the user wishes to

normally use (but without the `.desktop` extension) and a `Language` key that is the language that the user wishes to use. If either of these keys is missing, the system default is used. The file would normally look as follows:

```
[Desktop]
Session=gnome
Language=cs_CZ.UTF-8
```

The user can also configure a face image for the face browser. This is done by copying an image into a file called `~/.face`. This should be a standard image that GTK+ can read, such as PNG.

Face images can also be placed in the global face directory, which is normally `<share>/faces/` (though this can be configured with the `GlobalFaceDir` configuration option) and the filename should be the name of the user, optionally with a `.png` appended.

Controlling GDM

You can control GDM behavior during runtime in several different ways. You can either run certain commands, or you can talk to GDM using either a unix socket protocol, or a FIFO protocol.

Commands

To stop GDM, you can either send the TERM signal to the main daemon or run the `gdm-stop` command which is in the `<sbin>/` directory. To restart GDM, you can either send the HUP signal to the main daemon or run the `gdm-restart` command which is also in the `<sbin>/` directory. To restart GDM but only after all the users have logged out, you can either send the USR1 signal to the main daemon or run the `gdm-safe-restart` command which is in the `<sbin>/` directory as well.

The `gdmflexiserver` command can be used to start new flexible (on demand) servers. Run `gdmflexiserver --help` to get a listing of possible options. This command will also lock the current screen with xscreensaver so that the user can safely walk away from the machine and let someone else log in. XFree86 will automatically switch back to the same virtual terminal (if your operating system supports it), after the new session has ended, so this should work quite transparently to the users.

The FIFO protocol

GDM also provides a FIFO called `.gdmfifo` in the `ServAuthDir` directory (usually `<var>/gdm/.gdmfifo`). You must be root to use this protocol, and it is mostly used for internal GDM chatter. It is a very simple protocol where you just echo a command on a single line to this file. It can be used to tell GDM things such as restart, suspend the machine, or restart all X servers next time it has a chance (which would be useful from an X configuration program).

Full and up to date documentation of the commands and their use is contained in the GDM source tree in the file `daemon/gdm.h`. Look for the defines starting with `GDM_SOP_`. The commands which require the pid of the slave as an argument are the ones that are really used for internal communication of the slave with the master and should not be used.

Socket Protocol

GDM provides a unix domain socket for communication at `/tmp/.gdm_socket`. Using this you can check if GDM is running, the version of the daemon, the current servers that are running and who is logged in on them, and if GDM supports it on your operating system, also the virtual terminals of all the console logins. You can also request new flexible servers to be run with this protocol, but this is best done with the `gdmflexiserver` command.

gdmflexiserver accepts the following commands with the --command option:

```
VERSION
AUTH_LOCAL
FLEXI_XSERVER
FLEXI_XNEST
CONSOLE_SERVERS
ALL_SERVERS
UPDATE_CONFIG
GREETERPIDS
QUERY_LOGOUT_ACTION
SET_LOGOUT_ACTION
SET_SAFE_LOGOUT_ACTION
QUERY_VT
SET_VT
CLOSE
```

These are described in detail below, including required arguments, response format, and return codes.

VERSION

```
VERSION: Query version
Supported since: 2.2.4.0
Arguments:  None
Answers:
  GDM <gdm version>
  ERROR <err number> <english error description>
     200 = Too many messages
     999 = Unknown error
```

AUTH_LOCAL

```
AUTH_LOCAL: Setup this connection as authenticated for
            FLEXI_SERVER Because all full blown (non-Xnest)
            servers can be started only from users logged in
            locally, and here gdm assumes only users logged
            in from gdm.  They must pass the xauth
            MIT-MAGIC-COOKIE-1 that they were passed before
            the connection is authenticated.
Note:       The AUTH LOCAL command requires the
            --authenticate option, although only
            FLEXI XSERVER uses this currently.
Supported since: 2.2.4.0
Arguments:  <xauth cookie>
  <xauth cookie> is in hex form with no 0x prefix
Answers:
  OK
  ERROR <err number> <english error description>
     0 = Not implemented
     100 = Not authenticated
     200 = Too many messages
```

```
     999 = Unknown error
```

FLEXI_XSERVER

```
FLEXI_XSERVER: Start a new X flexible server
Only supported on connection that passed AUTH_LOCAL
Supported since: 2.2.4.0
Arguments:  <xserver type>
  If no arguments, starts the standard x server
Answers:
  OK <display>
  ERROR <err number> <english error description>
     0 = Not implemented
     1 = No more flexi servers
     2 = Startup errors
     3 = X failed
     4 = X too busy
     6 = No server binary
     100 = Not authenticated
     200 = Too many messages
     999 = Unknown error
```

FLEXI_XNEST

```
FLEXI_XNEXT: Start a new flexible Xnest server
Note:        Supported an older version from 2.2.4.0, later
             2.2.4.2, but since 2.3.90.4 you must supply 4
             arguments or ERROR 100 will be returned.  This
             will start Xnest using the XAUTHORITY file
             supplied and as the uid same as the owner of
             that file (and same as you supply).  You must
             also supply the cookie as the third argument
             for this display, to prove that you indeed are
             this user.  Also this file must be readable
             ONLY by this user, that is have a mode of 0600.
             If this all is not met, ERROR 100 is returned.
Note:        The cookie should be the MIT-MAGIC-COOKIE-1,
             the first one gdm can find in the XAUTHORITY
             file for this display.  If that's not what you
             use you should generate one first.  The cookie
             should be in hex form.
Supported since: 2.3.90.4
Arguments:  <display to run on> <uid of requesting user>
            <xauth cookie for the display> <xauth file>
Answers:
  OK <display>
  ERROR <err number> <english error description>
     0 = Not implemented
     1 = No more flexi servers
     2 = Startup errors
     3 = X failed
     4 = X too busy
     5 = Xnest can't connect
     6 = No server binary
     100 = Not authenticated
     200 = Too many messages
     999 = Unknown error
```

CONSOLE_SERVERS

```
CONSOLE_SERVERS: List all console servers, useful for linux
                 mostly.  Doesn't list xdmcp and xnest
                 non-console servers
Supported since: 2.2.4.0
Arguments:  None
Answers:
  OK <server>;<server>;...

  <server> is <display>,<logged in user>,<vt or xnest display>

  <logged in user> can be empty in case no one logged
  in yet, and <vt> can be -1 if it's not known or not
  supported (on non-linux for example).  If the display is an
  xnest display and is a console one (that is, it is an xnest
  inside another console display) it is listed and instead of
  vt, it lists the parent display in standard form.

  ERROR <err number> <english error description>
     1 = Not implemented
     200 = Too many messages
     999 = Unknown error
```

ALL_SERVERS

```
ALL_SERVERS: List all servers, including console, remote, xnest.
             This can, for example, be useful to figure out if
             the server you are on is managed by the gdm daemon,
             by seeing if it is in the list.  It is also somewhat
             like the 'w' command but for graphical sessions.
Supported since: 2.4.2.96
Arguments:  None
Answers:
  OK <server>;<server>;...

  <server> is <display>,<logged in user>

  <logged in user> can be empty in case no one logged in yet

  ERROR <err number> <english error description>
     0 = Not implemented
     200 = Too many messages
     999 = Unknown error
```

UPDATE_CONFIG

```
UPDATE_CONFIG: Tell the daemon to update config of some key.
               Any user can really request that values are
               re-read but the daemon caches the last date
               of the config file so a user can't actually
               change any values unless they can write the
               config file.
Supported keys:
  security/AllowRoot (2.3.90.2)
  security/AllowRemoteRoot (2.3.90.2)
  security/AllowRemoteAutoLogin (2.3.90.2)
  security/RetryDelay (2.3.90.2)
  security/DisallowTCP (2.4.2.0)
  daemon/Greeter (2.3.90.2)
```

```
  daemon/RemoteGreeter (2.3.90.2)
  xdmcp/Enable (2.3.90.2)
  xdmcp/Port (2.3.90.2)
  xdmcp/PARAMETERS (2.3.90.2) (pseudokey, all the parameters)
         xdmcp/MaxPending
         xdmcp/MaxSessions
         xdmcp/MaxWait
         xdmcp/DisplaysPerHost
         xdmcp/HonorIndirect
         xdmcp/MaxPendingIndirect
         xdmcp/MaxWaitIndirect
         xdmcp/PingIntervalSeconds (only affects new connections)
  daemon/TimedLogin (2.3.90.3)
  daemon/TimedLoginEnable (2.3.90.3)
  daemon/TimedLoginDelay (2.3.90.3)
  greeter/SystemMenu (2.3.90.3)
  greeter/ConfigAvailable (2.3.90.3)
  greeter/ChooserButton (2.4.2.0)
  greeter/SoundOnLoginFile (2.5.90.0)
  daemon/AddGtkModules (2.5.90.0)
  daemon/GtkModulesList (2.5.90.0)
Supported since: 2.3.90.2
Arguments:  <key>
  <key> is just the base part of the key such as
  "security/AllowRemoteRoot"
Answers:
  OK
  ERROR <err number> <english error description>
     0 = Not implemented
     50 = Unsupported key
     200 = Too many messages
     999 = Unknown error
```

GREETERPIDS

```
GREETERPIDS: List all greeter pids so that one can send HUP
             to them for config rereading.  Of course one
             must be root to do that.
Supported since: 2.3.90.2
Arguments:  None
Answers:
  OK <pid>;<pid>;...
  ERROR <err number> <english error description>
     0 = Not implemented
     200 = Too many messages
     999 = Unknown error
```

QUERY_LOGOUT_ACTION

```
QUERY_LOGOUT_ACTION: Query which logout actions are possible
Only supported on connections that passed AUTH_LOCAL.
Supported since: 2.5.90.0
Answers:
  OK <action>;<action>;...
     Where action is one of HALT, REBOOT or SUSPEND.  An
     empty list can also be returned if no action is possible.
     A '!' is appended to an action if it was already set with
     SET_LOGOUT_ACTION or SET_SAFE_LOGOUT_ACTION.  Note that
     SET_LOGOUT_ACTION has precedence over
     SET_SAFE_LOGOUT_ACTION.
```

```
  ERROR <err number> <english error description>
     0 = Not implemented
     100 = Not authenticanted
     200 = Too many messages
     999 = Unknown error
```

SET_LOGOUT_ACTION

```
SET_LOGOUT_ACTION:  Tell the daemon to halt/reboot/suspend
                    after slave process exits.
Only supported on connections that passed AUTH_LOCAL.
Supported since: 2.5.90.0
Arguments:  <action>
  NONE           Set exit action to 'none'
  HALT           Set exit action to 'halt'
  REBOOT         Set exit action to 'reboot'
  SUSPEND        Set exit action to 'suspend'

Answers:
  OK
  ERROR <err number> <english error description>
     0 = Not implemented
     7 = Unknown logout action, or not available
     100 = Not authenticanted
     200 = Too many messages
     999 = Unknown error
```

SET_SAFE_LOGOUT_ACTION

```
SET_SAFE_LOGOUT_ACTION:  Tell the daemon to halt/reboot/suspend
                         after everybody logs out.  If only
                         one person logs out, then this is
                         obviously the same as the
                         SET_LOGOUT_ACTION.  Note that
                         SET_LOGOUT_ACTION has precendence over
                         SET_SAFE_LOGOUT_ACTION if it is set to
                         something other then NONE.  If no one
                         is logged in, then the action takes
                         effect immediately.
Only supported on connections that passed AUTH_LOCAL.
Supported since: 2.5.90.0
Arguments:  <action>
  NONE           Set exit action to 'none'
  HALT           Set exit action to 'halt'
  REBOOT         Set exit action to 'reboot'
  SUSPEND        Set exit action to 'suspend'

Answers:
  OK
  ERROR <err number> <english error description>
     0 = Not implemented
     7 = Unknown logout action, or not available
     100 = Not authenticanted
     200 = Too many messages
     999 = Unknown error
```

QUERY_VT

```
QUERY_VT:  Ask the daemon about which VT we are currently on.
           This is useful for logins which don't own
           /dev/console but are still console logins.  Only
           supported on Linux currently, other places will
           just get ERROR 8.  This is also the way to query
           if VT support is available in the daemon in the
           first place.
Only supported on connections that passed AUTH_LOCAL.
Supported since: 2.5.90.0
Arguments:  None
Answers:
  OK <vt number>
  ERROR <err number> <english error description>
     0 = Not implemented
     8 = Virtual terminals not supported
     100 = Not authenticanted
     200 = Too many messages
     999 = Unknown error
```

SET_VT

```
SET_VT:  Change to the specified virtual terminal.
         This is useful for logins which don't own /dev/console
         but are still console logins.  Only supported on Linux
         currently, other places will just get ERROR 8.
Only supported on connections that passed AUTH_LOCAL.
Supported since: 2.5.90.0
Arguments:  <vt>
Answers:
  OK
  ERROR <err number> <english error description>
     0 = Not implemented
     8 = Virtual terminals not supported
     9 = Invalid virtual terminal number
     100 = Not authenticanted
     200 = Too many messages
     999 = Unknown error
```

CLOSE

```
CLOSE Answers: None
Supported since: 2.2.4.0
```

GDM Commands

The GDM package provides quite a few different commands.

`gdm` and `gdm-binary` Command Line Options

The `gdm` command is really just a script which runs the `gdm-binary`, passing along any options. Before launching gdm-binary, the gdm wrapper script will source the `/etc/profile` file to set the standard system environment variables. In order to better support internationalization, it will also set the LC_MESSAGES environment variable to LANG if neither LC_MESSAGES or LC_ALL are set. If you really need to set some additional environment before launching GDM, you can do so in this file.

`gdm` and `gdm-binary` Command Line Options

--help

Gives a brief overview of the command line options.

-nodaemon

If this option is specified, then gdm does not fork into the background when run. You can use just a single dash with this option to preserve compatibility with XDM.

--no-console

Tell the daemon that it should not run anything on the console. This means that none of the local servers from the `[servers]` section will be run, and the console will not be used for communicating errors to the user. An empty `[servers]` section automatically implies this option.

--preserve-ld-vars

When clearing the environment internally, preserve all variables starting with LD_. This is mostly for debugging purposes.

--version

Print the version of the GDM daemon.

--wait-for-go

If started with this option, gdm will init, but only start the first local display and then wait for a GO message in the fifo protocol. No greeter will be shown until the GO message is sent. Also flexiserver requests will be denied and XDMCP will not be started until GO is given. This is useful for initialization scripts which wish to start X early, but where you don't yet want the user to start logging in. So the script would send the GO to the fifo once it is ready and GDM will then continue. This functionality was added in version 2.5.90.0.

`gdmchooser` and `gdmlogin` Command Line Options

The `gdmgreeter` and `gdmlogin` are two different login programs, either can be used by GDM. `gdmgreeter` is themeable with GDM themes while `gdmlogin` is themable with GTK+ themes. These programs are normally executed by the GDM daemon. Both programs support standard GNOME options.

`gdmchooser` Command Line Options

The `gdmchooser` is the XDMCP chooser application. The `gdmchooser` is normally executed by the GDM daemon. It supports the following options for XDM compatibility. This program supports standard GNOME options.

`gdmchooser` Command Line Options

-xdmaddress=SOCKET

Socket for XDM communication.

--clientaddress=ADDRESS

Client address to return in response to XDM. This option is for running gdmchooser with XDM, and is not used within GDM.

-connectionType=TYPE

Connection type to return in response to XDM. This option is for running gdmchooser with XDM, and is not used within GDM.

`gdmconfig` and `gdmsetup` Command Line Options

`gdmconfig` is a wrapper script for running `gdmsetup` for binary name compatibility. `gdmconfig` is installed to the `<sbin>/` directory.

gdmsetup runs a graphical program for modifying the GDM configuration file, gdm.conf. Normally on systems that support the PAM userhelper, this is setup such that when you run `gdmsetup` as an ordinary user, it will first ask you for your root password before starting. Otherwise, this program may only be run as root. This program supports standard GNOME options.

`gdmXnestchooser` and `gdmXnest` Command Line Options

The `gdmXnestchooser` command automatically gets the correct display number, sets up access, and runs `Xnest` with -indirect localhost. This way you get an XDMCP chooser provided by your machine. You can also supply as an argument the hostname whose chooser should be displayed, so `gdmXnestchooser somehost` will run the XDMCP chooser from host `somehost` inside an Xnest session. You can make this command do a direct query instead by passing the `-d` option as well. In addition to the following options, this program also supports standard GNOME options.

`gdmXnest` is a symbolic link to `gdmXnestchooser` and is the same as using the --no-query and --no-gdm-check options with `gdmXnestchooser`. It is useful for running `Xnest` without actually connecting somewhere. It will print out the display setting on standard output that you can use to connect to this server. This is useful mostly for developers who perhaps wish to test their apps running on a completely separate server.

`gdmXnestchooser` and `gdmXnest` Command Line Options

-x, --xnest=STRING

Xnest command line, default is "Xnest"

-o, --xnest-extra-options=OPTIONS

Extra options for Xnest, default is no options.

-n, --no-query

Just run Xnest, no query (no chooser)

-d, --direct

Do direct query instead of indirect (chooser)

-B, --broadcast

Run broadcast instead of indirect (chooser)

-b, --background

Run in background

--no-gdm-check

Don't check for running GDM

`gdmflexichooser` Command Line Options

The `gdmflexiserver` command which runs flexible (on demand) X servers.

`gdmflexiserver` lets a user log in once and then quits. This is, for example useful if you are logged in as user A, and user B wants to log in quickly but user A does not wish to log out. The X server takes care of the virtual terminal switching so it works transparently. There is also a flexi server as an Xnest, that is an X server in a window. This is requested by running `gdmflexiserver`.

If there is more than one server defined with flexible=true, then the user is given a dialog with those choices upon running gdmflexiserver

The `gdmflexiserver` command option provides a way to send arbitrary commands to GDM and can be used to debug problems or in scripts, although `gdmflexiserver` does require X to be running.

In addition to the following options, `gdmflexiserver` also supports standard GNOME options.

`gdmflexichooser` Command Line Options

-c, --command=COMMAND

Send the specified protocol command to gdm

-n, --xnest

Xnest mode

-l, --no-lock

Do not lock current screen

-d, --debug

Debugging output

-a, --authenticate

Authenticate before running --command

`gdmphotosetup` Command Line Options

Allows the user to select an image that will be used as the user's photo by GDM's face browser, if enabled by GDM. The selected file is stored as ~/.face. This program accepts standard GNOME options.

`gdmthemetester` Command Line Options

`gdmthemetester` takes two parameters. The first parameter specifies the environment and the second parameter specifies the path name or the name of a theme to view. This is a tool for viewing a theme outside of GDM. It is useful for testing or viewing themes. `gdmthemetester` requires that the system support `gdmXnest`. Note that themes can display differently depending on the theme's "Show mode". `gdmthemetester` allows viewing the themes in different modes via the environment option. Valid environment values and their meanings follow:

```
console       - In console mode.
console-timed - In console non-flexi mode.
flexi         - In flexi mode.
xdmcp         - In remote (XDMCP) mode.
remote-flexi  - In remote (XDMCP) & flexi mode.
```

`gdm-restart` Command Line Options

`gdm-restart` stops and restarts GDM by sending the GDM daemon a HUP signal. This command will immediately terminate all sessions and log out users currently logged in with GDM. `gdm-restart` is installed to the `<sbin>/` directory.

`gdm-safe-restart` Command Line Options

`gdm-safe-restart` stops and restarts GDM by sending the GDM daemon a USR1 signal. GDM will be restarted as soon as all users log out. `gdm-safe-restart` is installed to the `<sbin>/` directory.

`gdm-stop` Command Line Options

`gdm-stop` stops GDM by sending the GDM daemon a TERM signal. `gdm-stop` is installed to the `<sbin>/` directory.

Graphical Greeter Themes

This section describes the creation of themes for the Graphical Greeter. For examples including screenshots, see the standard installed themes and the themes from the theme website[5].

Theme Overview

GDM Themes can be created by creating an XML file that follows the specification in gui/greeter/greeter.dtd. Theme files are stored in the directory `<share>/gdm/themes/<theme_name>`. Usually this would be under `/usr/share`. The theme directory should contain a file called `GdmGreeterTheme.desktop` which has similar format to other .desktop files and looks like:

```
[GdmGreeterTheme]
Encoding=UTF-8
Greeter=circles.xml
Name=Circles
Description=Theme with blue circles
Author=Bond, James Bond
```

```
Copyright=(c) 2002 Bond, James Bond
Screenshot=screenshot.png
```

The Name, Description, Author and Copyright fields can be translated just like the other `.desktop`files. All the files that are mentioned should be in the theme directory itself. The Screenshot field points to a file which should be a 200x150 screenshot of the theme in action (it is OK not to have one, but it makes it nicer for user). The Greeter field points to an XML file that contains the description of the theme. The description will be given later.

Once you have theme ready and installed you can test it with the installed `gdmthemetester` script. This script assumes that you also have installed the Xnest X server. It takes two arguments, first the environment that should be used. This is one of console, console-timed, flexi, remote-flexi, xdmcp. Where console is a standard console login, console-timed is a console login with a timed login going on, flexi is for any local flexible server, remote-flexi is for flexi server that is not local (such as an Xnest flexiserver run from a remote display) and xdmcp is for remote xdmcp connections. The second argument is the theme name. So for example to test how things look in the xdmcp mode with the circles theme you would run:

```
gdmthemetester xdmcp circles
```

Be sure to test all the environments with your theme, and make sure to test how the caps lock warning looks by pressing caps lock. This is also a good way to take screenshots, just take a screenshot of the Xnest window. This can be done in GNOME by focusing the Xnest window and pressing Alt-PrintScreen.

Once you have all this done, then make a tarball that contains the directory name (so that you could just untar it in the `/usr/share/gdm/themes` directory). And this is the tarball you distribute and people can install from the graphical setup program. You can do this with the commands:

```
cd /usr/share/gdm/themes
tar czvf <theme_name>.tar.gz <theme_name>/
```

Detailed Description of Theme XML format

Box Nodes

Box nodes are container nodes for item nodes. Box nodes are specified as follows:

```
<box orientation="alignment" min-width="num" xpadding="num"
ypadding="num" spacing="num" homogeneous="bool">
```

Where "num" means number and bool means either "true" or "false". The alignment value can be either "horizontal" or "vertical". If you leave any property off it will default to zero or "false" in case of "homogeneous", and "vertical" for the orientation.

If the box is homogeneous then the children are allocated equal amount of space.

The "min-width" must be specified in pixels. Obviously there is also a corresponding "min-height" property as well.

Fixed Nodes

Fixed is a container that has its children scattered about laid out with precise coordinates. The size of this container is the biggest rectangle that contains all the children. Fixed has no extra properties and so you just use:

```
<fixed>
```

Then you put other items with proper position nodes inside this.

The "toplevel" node is really just like a fixed node.

Item Nodes

A GDM Theme is created by specifying a hierarchy of item and box nodes. Item nodes can have the following value for "type":

entry
: Text entry field.

list
: A list widget.

label
: A text label. Must have a "text" node to specify the text.

pixmap
: An pixmap image in a format that gdk-pixbuf supports like PNG, JPEG, Tiff, etc...)

rect
: Rectangle.

svg
: Scaled Vector Graphic image.

For example:

```
<item type="label">
```

Items can specify ID values which gives them a specific look and feel or formatting. Furthermore you can customize the login process by adding custom widgets with custom id's for some items (currently only the list item)

Entry items can have id values as follows:

user-pw-entry
: Entry field for userid and password entry. This is the field used for responses for the PAM/GDM questions (Username, Password, etc..).

List items can have id values as follows:

userlist
: A user browser list, so that users can pick their username by clicking on this instead of typing.

Furthermore, you can have an arbitrary id (I'd recommend starting the id with 'custom' not to conflict with future additions to this spec) and ask extra information of the user. See the section 'Custom Widgetry'

Label items can have id values as follows:

clock
: Label the displays the date and time.

pam-prompt
: Label the displays PAM prompt. This is the prompt that PAM uses to ask for username, password, etc...

pam-error
: Label the displays PAM/GDM error messages. Such as when user can't log in.

pam-message
: Label the displays PAM message. These are messages that PAM/GDM gives about state of the account, help about the prompts and other information.

timed-label
: Label that displays timed login information.

Rectangles can have id values as follows:

caps-lock-warning
: Displays an icon that shows if the CAPS LOCK key is depressed. This rectangle will be hidden/shown appropriately

If an item is of type rect, the item can be a button. Buttons must also include a "button" value as follows:

```
<item type="rect" id="disconnect_button" button="true">.
```

Possible values for button ids are as follows:

chooser_button
: Runs the XDMCP chooser.

config_button
: Runs the GDM Setup program.

disconnect_button
: Disconnect from remote session.

language_button
: Displays the language selection dialog.

halt_button
: Halt (shuts down) the system.

reboot_button
: Reboot the system.

session_button

List and select from available sessions.

suspend_button

Suspend the system.

system_button

Perform halt/reboot/suspend/etc. options (if allowed by gdm configuration). Also allows user to run configurator if user enters root password (again if allowed by gdm configuration). This is usually now labeled Actions, and referred to as the Actions menu.

Position Node

Each item can specify its position and size via the "pos" node. For example:

```
<pos x="0" y="4" width="100%" height="100%"/>
```

Both position and size can be given in percent and it will be taken as the percentage of the size of the current container. For toplevel items it's the percentage of the whole screen.

For x and y, you can also specify a negative position which means position from the right or bottom edge. But this only applies with absolute coordinates. With percentage you can specify negative position and it will be still from the same edge.

The position also specifies the anchor of the item, this can be "n", "ne", "e", "se", "s", "sw", "w" and "nw" or "center" which stand for the different edges/corners or "center" for center. For example:

```
<pos x="10%" y="50%" anchor="w" width="80%" height="95"/>
```

If the item contains a box, you can specify width and height to be "box" to mean that they are supposed to be the width and height of the box, that is the items in the box plus the padding.

You can also specify an "expand" property to either be "true" or false. If true then the child will be expanded in the box as much as possible (that is it will be given more space if available).

There are two extra properties you can specify (as of 2.4.4.3) for labels (and labels only). The first is "max-width" which will specify the maximum width of the label in pixels. And the second is "max-screen-percent-width" which specifies the maximum percentage of the screen width that the label can occupy. By default no label will occupy more then 90% of the screen by width. An example may be:

```
<item type="label">
<pos x="10%" max-screen-percent-width="50%"/>
```

Show Node

Some items may only display in certain modes, like when doing a remote display. Multiple values can be specified and must be separated with commas. The following values are possible:

`console` - In console mode.

`console-fixed` - In console non-flexi mode.

`console-flexi` - In console & flexi mode.

`flexi` - In flexi mode.

`remote` - In remote mode.

`remote-flexi` - In remote & flexi mode.

For example:

```
<show modes="flexi,remote"/>
```

You can also specify the "type" value to indicate that certain items should only be displayed if the type is true. Valid values include the following:

`chooser`, if ChooserButton is set to "true" in `gdm.conf` file.

`config`, if ConfigAvailable is set to "true" in `gdm.conf` file.

`halt`, if HaltDaemon is specified in `gdm.conf` file.

`reboot`, if RebootCommand is specified in `gdm.conf` file.

`suspend`, if SuspendCommand is specified in `gdm.conf` file.

`system`, if SystemMenu is specified in `gdm.conf` file

`timed`, if TimedLoginEnabled is set to "true" in `gdm.conf` file.

For example:

```
<show modes="console" type="system"/>
```

Note that if SystemMenu is off then all of halt, reboot, suspend, chooser and config will not show, so this is a global toggle for them all. See some of the standard themes for how the show modes are used.

Normal/Active/Prelight Nodes

Depending on the item type, it can specify its color, font, or image via the following tags:

`normal` - normal state.

`active` - when the item has active focus.

`prelight` - when the mouse is hovering over the item.

When item is "rect" (alpha can be omitted and defaults to 0.0):

```
<normal color="#ffffff" alpha="0.0">
```

When item is "label":

```
<normal color="#ffffff" font="Sans 14"/>
```

When the item type is "pixmap" or "SVG", then the normal, active, and prelight tags specify the images to use as follows:

```
<normal file="picture.png" tint="#dddddd"/>
```

Note that relative pathnames are assumed to be in the same directory as the theme `.xml` file in `<share>/gdm/themes/<theme_name>`.

Text Node

Text tags are used by labels. They can be used to display localized text as follows (if the "xml:lang" attribute is omitted, the C locale is assumed):

```
<text xml:lang="fr">Option</text>
```

You can include pango markup in the text nodes for labels, however you must encode it. So for example to have the label of "foo^{bar}", you must type:

```
<text">foo&lt;sup&gt;bar&lt;/sup&gt;</text>
```

Stock

Certain common localized labels can be specified via the stock tags. The "text" tag is ignored if the "stock" tag is used. You should really use the stock labels rather then just putting all the translations into the themes. This gives faster load times and likely better translations. The following values are valid:

`caps-lock-warning`, _("You've got capslock on!")

`chooser`, _("_XDMCP Chooser")

`disconnect`, _("D_isconnect")

`halt`, _("Shut_down")

`language`, _("_Language")

`quit`, _("_Quit")

`reboot`, _("_Reboot")

`session`, _("_Session")

`suspend`, _("Sus_pend")

`system`, _("_Actions") (Formerly "S_ystem")

`timed-label`, _("User %s will login in %d seconds")

`username-label`, _("Username:")

`welcome-label`, _("Welcome to %h")

For example:

```
<stock type="welcome-label"/>
```

Custom Widgetry

Currently there is one item which can be customizable and this is the list item. If you need to ask the user extra things, such as to pick from a list of places to log into, or set of custom login sessions you can setup the list item and add listitem children that describe the choices. Each listitem must have an id and a text child.

The choice will be recorded in the file `<ServAuthDir>/<display>.GreeterInfo` as `<list id>=<listitem id>`.

For example suppose we are on display :0, `ServAuthDir` is `/var/gdm` and we have the following in the theme:

```
<item type="list" id="custom-config">
<pos anchor="nw" x="1" y="1" height="200" width="100"/>
<listitem id="foo">
<text>Foo</text>
</listitem>
<listitem id="bar">
<text>Bar</text>
</listitem>
</item>
```

Then if the user chooses 'Foo' then `/var/gdm/:0.GreeterInfo` will contain:

```
custom-config=foo
```

Accessibility

GDM supports "Accessible Login" to allow users to log in to their desktop session even if they cannot easily use the screen, mouse, or keyboard in the usual way. This feature allows the user to launch assistive technologies at login time by means of special "gestures" from the standard keyboard and from a keyboard, pointing device, or switch device attached to the USB or PS/2 mouse port. It also allows the user to change the visual appearance of the login UI before logging in, for instance to use a higher-contrast color scheme for better visibility. GDM only supports accessibility with the Standard Greeter, so the "Greeter" parameter in gdm.conf must be set to the Standard Greeter "gdmlogin".

Accessibility Configuration

In order to enable Accessible Login, the system administrator must make some changes to the default login configuration by manually modifying three human-readable configuration files, stored in gdm.conf, AccessKeyMouseEvents and AccessDwellMouseEvents.

In order to allow users to change the color and contrast scheme of the login dialog, make sure the "AllowThemeChange" parameter in gdm.conf is set to "true".

To restrict user changes to the visual appearance to a subset of available themes, the "GtkThemesToAllow" parameter in gdm.conf can be set to a list of acceptable themes separated by commas. For example:

```
GtkThemesToAllow=HighContrast,HighContrastInverse
```

To enable the use of assistive technologies such as the Onscreen Keyboard, Screen Reader, or Magnifier, the "AddGtkModules" parameter in gdm.conf must be uncommented and set to "true". Also the "GtkModulesList" parameter must be uncommented and set as follows:

```
GtkModulesList=gail:atk-bridge:dwellmouselistener:keymouselistener
```

System administrators may wish to load only the minimum subset of these modules which is required to support their user base. Depending on the end-user needs,

not all of the above GtkModules may need to be loaded. If your end-users need the integrated Screen Reader and Magnifier, you must include "gail" and "atk-bridge". If your end-users will be using a pointing device without buttons or switches, include "dwellmouselistener". If some of your users will use pointing devices with switches, alternative physical keyboards, or switch/button devices, include "keymouselistener". Including all four is suitable for most system configurations. The Onscreen Keyboard can operate without gail and atk-bridge, but with a reduced feature set; for optimum accessibility we recommend including gail and atk-bridge.

Once "keymouselistener" and/or "dwellmouselistener" have been added to the GtkModules loaded by GDM, you can assign end-user actions with the launching of specific assistive technologies. These gesture associations are contained in files AccessKeyMouseEvents and AccessDwellMouseEvents, respectively. The gesture format is described in the two configuration files.

The AccessKeyMouseEvents file controls the keymouselistener Gesture Listener and is used to define key-press, mouse button, or XInput device sequences that can be used to launch programs needed for accessibility. In order to reduce the likelihood of unintentional launch, these 'gestures' may be associated with multiple switch presses and/or minimum durations.

The DwellKeyMouseEvents file controls the dwellmouselistner and supports gestures that involve only motion of a pointing device such as the system mouse of an alternative pointing device such as a head pointer or trackball may also be defined. All gestures are specified by the same syntax; that is, there is no distinction between a 'core mouse' gesture and motion from an alternate input device.

Motion gestures are defined as "crossing events" into and out of the login dialog window. If the 'dwellmouselistener' GtkModule is loaded, alternative pointing devices are temporarily "latched" to the core pointer, such that motion from alternative devices results in movement of the onscreen pointer.

In order to use text-to-speech services at login time (for instance, when using the Screen Reader in speech mode) on some operating systems, the gdm user must be made a member of the "audio" group

Example Configurations

This section has some example configurations that are useful for various setups.

Terminal Lab With One Server

Suppose you want to make a lab full of X terminals that all connect to one main server. So let's call one X terminal `xterminal` and lets call the server `appserver`. You install GDM on both.

On `appserver` you enable XDMCP, so you have

```
[xdmcp]
Enable=true
```

If you want no local screens here, you can then make the `[servers]` section empty.

On the `xterminal` you disable XDMCP (you don't want anyone to connect to the xterminal really). You will add a server type perhaps called `Terminal` as follows:

```
[server-Terminal]
name=Terminal server
command=/usr/X11R6/bin/X -terminate
flexible=false
handled=false
```

This definition should in fact be included in the standard configuration file. Notice that we made the `handled` key false since we don't want GDM to handle this server localy. Also note that we have not yet added the `-query` argument, you can add that here, or in the `[servers]` section. We'll define our local servers as follows:

```
[servers]
0=Terminal -query appserver
```

This will run a direct XDMCP query to the server named `appserver`.

Terminal Lab With Two Or More Servers

Suppose you want to make a lab full of X terminals that all connect to some choice of servers. For now let's make it `appserverone` and `appservertwo`. Again we'll call our example X terminal machine `xterminal`. The setup on both servers is the same as with the case of one server in the previous section. You do not need to explicitly enable indirect queries on the server since we'll run the choosers locally on the X terminals.

So on the `xterminal` you again disable XDMCP. You will add a server type perhaps called `Chooser` as follows:

```
[server-Chooser]
name=Chooser server
command=/usr/X11R6/bin/X
flexible=false
chooser=true
```

And again this definition should in fact be included in the standard configuration file. Notice that we made the `chooser` key true here. This will run the XDMCP chooser for this server, and when the user chooses a host GDM will run a query for that host. Then we'll define our local servers as follows:

```
[servers]
0=Chooser
```

The XDMCP chooser on the X terminal will normally give a broadcast query to see which servers exist on the network. If the two servers are not reachable by a broadcast query, you must add them by hand to the configuration file. So in the `[chooser]` section you would have:

```
Hosts=appserverone,appservertwo
```

and any other servers you wish the users to be able to connect to.

Sometimes you may want to run the chooser on the server side however. Then what you want to do is to run a configuration similar to the previous section about the one server configuration with XDMCP indirect queries enabled on `appserver` and on the X terminals you'd have

```
[servers]
0=Terminal -indirect appserver
```

This way for example you only have to maintain one `Hosts` entry. However as a disadvantage then, the `appserver` must then always be available. So it's not good

for situations where you want to have serveral servers and not all of them have to be on all the time. You could also have one of the X terminals handle indirect XDMCP queries and serve up the chooser to the other X terminals.

License

This program is free software; you can redistribute it and/or modify it under the terms of the *GNU General Public License*[6] as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the *GNU General Public License* for more details.

A copy of the *GNU General Public License* is included as an appendix to the *GNOME Users Guide*. You may also obtain a copy of the *GNU General Public License* from the Free Software Foundation by visiting their Web site[7] or by writing to

Free Software Foundation, Inc.
59 Temple Place - Suite 330
Boston, MA 02111-1307
USA

Notes

1. ghelp:fdl
2. http://www.jirka.org/gdm.html
3. man:hosts.allow
4. man:hosts.allow
5. http://art.gnome.org/themes/gdm_greeter/index.php
6. gnome-help:gpl
7. http://www.fsf.org