

# **GenieGate - User Management Software**



# Table of Contents

<b>1. Introduction.....</b>	<b>1</b>
<b>2. Installation.....</b>	<b>2</b>
2.1. Directory Structure.....	2
2.2. Quick Start .....	3
2.2.1. Install database .....	3
2.2.2. Edit ~/geniegate.ini.....	4
2.2.3. Run setup.php .....	5
2.3. Basic Apache Usage.....	5
<b>3. Where to get help .....</b>	<b>7</b>
<b>4. Administrative functions.....</b>	<b>8</b>
4.1. Sign On .....	8
4.2. Main Screen .....	8
4.3. User Screen .....	9
4.3.1. Create User .....	10
4.3.2. Search Users .....	11
4.4. Groups .....	12
4.5. Properties .....	13
4.6. Export (and import).....	13
<b>5. User level functions.....</b>	<b>15</b>
5.1. The web/user/ files .....	15
5.2. User sign up .....	15
5.3. Change Password .....	16
5.4. Re-send confirmation code.....	17
5.5. Log Out .....	17
5.6. Logins.....	18
5.7. Forgotten Passwords .....	19
5.8. Personal settings.....	20
<b>6. Customization .....</b>	<b>21</b>
6.1. About the HTML .....	21
<b>7. User Templates.....</b>	<b>22</b>
7.1. shell.php - Shell template.....	22
7.2. Overview of the templates.....	22
7.3. email templates .....	24
<b>8. Programmers API.....</b>	<b>26</b>
8.1. Login via the API.....	26
8.2. "Passive" login .....	26
8.3. Plug ins.....	26
8.4. Creating/Modifying the GenieGate libraries.....	27
<b>A. Caveats and things to note .....</b>	<b>28</b>
A.1. HTTP Authentication and Log outs .....	28
<b>B. vCard implementation.....</b>	<b>29</b>
<b>C. Plugins.....</b>	<b>30</b>
C.1. ApacheDBM large scale htaccess. ....	30

<b>D. XML export/import .....</b>	<b>31</b>
D.1. Overview of the format .....	31
D.2. The DTD for the XML format .....	32
<b>E. License.....</b>	<b>34</b>
<b>F. MySQL tables.....</b>	<b>36</b>

# List of Tables

- 5-1. Form variables for sign up..... 15
- 5-2. Form variables for password change ..... 16
- 5-3. Form variables for confirmation code reminder ..... 17
- 5-4. Form variables for password change ..... 17
- 5-5. Form variables for Login..... 19
- 5-6. Form variables for Password reminder..... 19
- 5-7. Form variables for changing properties..... 20
- B-1. vCard fields and properties ..... 29
- F-1. MySQL tables ..... 36

# List of Examples

- 2-1. Configuring basic apache authentication..... 6
- C-1. Configuring for ApacheDBM with GNU dbm support ..... 30
- D-1. Example XML document ..... 31

# Chapter 1. Introduction

GenieGate is a flexible PHP application for managing user accounts, featuring a plugin model allowing it to interact with other applications.

GenieGate is well documented. This manual is large to provide documentation in specific areas most people will not need. Sections that are optional usually have a note at the beginning explaining that it can be skimmed over. The programmers API documentation is covered separately.

**Tip:** For your convenience, this manual is indexed at GenieGate (<http://www.geniegate.com/>). Please feel free to make use of the web site. It will contain the most up to date information available.

GenieGate supports Apache style groups and passwords along with PHP based solutions. It supports exporting and importing data in an XML format for backups and batch oriented interaction with other applications.

Wherever possible, it *stays out of your way* so you can design your web site in a way that fits your style. The user level functions support a `LOCATION` parameter which allows you control over where the user is directed after completing an operation. With this `LOCATION` variable, users may not even be aware they are using an account manager. To them, it may appear as though they're simply using your web site.

Through it's plugin API, it is possible to integrate GenieGate with other applications requiring user access controls. This allows you to have *one* user name and password for several cgi scripts and applications.

Through it's properties, GenieGate is capable of storing arbitrary *Key - Value* sets of data, classified in groups. This flexible property management allows additional properties to be added as time goes on, multiple classifications allow applications to utilize "pr. user" properties without interfering with GenieGate

GenieGate handles groups, which provides flexibility in access control. This means that certain users can have access to areas other users cannot.

Apart from being an application it is also an *API* for other applications to utilize.

# Chapter 2. Installation

In any flexible application designed to suite a wide variety of requirements several techniques of deployment are possible. GenieGate is no exception.

This chapter will describe the overall process. However each case is unique and there may be deviations from this guide.

## 2.1. Directory Structure

GenieGate is structured using a central directory for it's library files, separate directories for each deployment and web files located where it is convenient for the web master to place them.

- `~/geniegate/`

Each instance has it's own `~/geniegate` directory. This directory contains configuration files, templates and data required for it's own particular setup.

**Tip:** In your `geniegate.ini` configuration file, this is referred to as *WORKDIR*.

It is *recommended* that you place this directory in your HOME directory, (`$HOME/geniegate/` for UNIX users)

The various `common.php` files that reside under `web/` assume `~/geniegate/` is in your *UNIX home directory* If this is not the case, you will have to edit `common.php` and specify the `GENIEGATE_CONF_FILE` as well as `GENIEGATE_HOME`.

- `lib/`

The `lib/` directory contains the global files that are used by default. Files in this directory should never be edited, instead they should be copied to the appropriate subdirectory in `~/geniegate` and edited there. (very much like personalized settings in your `$HOME` directory for other applications such as emacs)

- `web/`

**Important:** It is *extremely* important that your private `~/geniegate` is not accessible by the web server

This directory will contain passwords and other information that should not be made public. It is recommended that this be in your UNIX HOME directory.

The `web/` directory refers to any files that are web accessible, typically some place in your DocumentRoot.

Before we proceed to the "Quick Start" section, it is assumed that your web directory is some place in web space, and can be reached through a web browser, that your `~/geniegate/` has been copied some place outside web space.

## 2.2. Quick Start

This section attempts to quickly walk through the process of installation and setup, it does not cover the Apache plugin configuration. If you have skipped through the directory structure, you should return to it now.

### 2.2.1. Install database

GenieGate requires access to a MySQL database.

#### Setup database

1. Create database **mysql> CREATE DATABASE dbname;**

**Tip:** It is recommended that GenieGate be given it's own database. (in cases where multiple instances of `~/geniegate` exist, separate databases are *required*)

2. Setup a user and grant permissions. **mysql> GRANT ALL PRIVILEGES ON dbname TO username IDENTIFIED BY 'secret';**

Where *dbname* is the name of the database you are creating, *username* is the username for connecting to the database and *secret* is the password used in connecting to your database.

3. Flush the privileges so that you may connect.

**mysql> FLUSH PRIVILEGES;**

**Tip:** Your internet service provider may have provided you with tools to manage mysql databases. You should check with your ISP for the final word on creating and granting access to databases. In many cases, these steps will have already been done for you.

More information about mysql, including security policies can be found at <http://www.mysql.com/>

## 2.2.2. Edit `~/geniegate.ini`

Next, edit your local `geniegate.ini` configuration file. In particular, make sure the `[databases]` section is accurate.

`~/geniegate/conf/geniegate.ini`

### LIBRARY\_PATH

Path to libraries, this must point to the global list of libraries.

### WORKDIR

This points to the local `~/geniegate` directory.

### PLUGINS

This is a colon delimited list of *plugins*

The remaining configuration values are divided into sections, these sections are divided by `[SectionName]` values. (this is a standard `.ini` configuration file)

### **[database]**

#### HOST

The host name mysql is running on.

#### USER

This is the username to connect to MySQL

#### PASS

The password to connect to the database.

#### DATABASE

The name of the database to connect to.

**Tip:** Other areas will need to be configured too, but they are dependent on the type of setup you will be doing. At first it's best to get the database working and return later for the other areas.

### 2.2.3. Run setup.php

In order to setup the database tables, run the *setup.php* script in your browser. It is important to note that this script *will only* give you your root login/password once. Running it a second time *will not* provide you with a password. (this is for security.)

After setup.php has completed, you should be able to login to your control panel. You should change your password as soon as possible.

You can, and probably should disable `web/admin/setup.php` after you are done with it.

Congratulations, you now have a minimal installation completed. You can login, have a look around and experiment with different things. Be aware that account creation may not work if the plugins are not configured.

## 2.3. Basic Apache Usage

If you are only using PHP based accounts, this section can be skimmed over. You should disable any plugins that make use of it. (Comment out the PLUGINS section in your `~/geniegate/conf/geniegate.ini`)

This section will quickly go through using the Apache plugin here. We assume you have a working knowledge of basic authentication and have used the **htpasswd** program that ships with Apache.

To get Apache password style authentication you need your configuration (`geniegate.ini`) PLUGINS to contain `GenieGate/Api/Plugin/Apache.php`

### Configuring [ApachePlugin]

#### AuthUserFile

Filename that contains Apache style users. (the same used by the **htpasswd**)

This file, and the directory it resides in *must be writable* by the server. If the server is running as user nobody, then you'll probably have to change the permissions on that directory to `777`.

Simply changing the permissions on the file won't work.

This file (and any files in it's directory) should NOT be web accessible.

#### AuthGroupFile

This is the filename of apache style groups.

Just as in the password file, the server needs write access to the directory.

### Example 2-1. Configuring basic apache authentication

```
PLUGINS=GenieGate/Api/Plugin/Apache.php

[ApachePlugin]
;; The password and group files for Apache regular authentication.

AuthUserFile=/home/joe/geniegate/conf/apache/passwords.txt
AuthGroupFile=/home/joe/geniegate/conf/apache/groups.txt
```

That should be it, GenieGate will now be able to update these two files each time a user account is created or modified. It is still up to you to configure your server to use these files.

If you do not need additional functionality, (such as users being able to sign themselves up) this may be all that is needed.

**Tip:** For Apache the relevant directives are `AuthType` , `AuthName`, `AuthUserFile` , `AuthGroupFile` , `Require` (for *Require group*) see your web servers documentation about how to create these files.

# Chapter 3. Where to get help

It is our business to help you. Please contact us (<http://www.geniegate.com/contact.php>) with any questions or concerns you may have.

Although clients receive priority help, free help is also available.

Although email support is the preferred method, telephone consulting is also available. In some circumstances, consulting is available in person as well. (you must be willing to pay for transportation expenses, this type of consulting is best reserved for extreme cases)

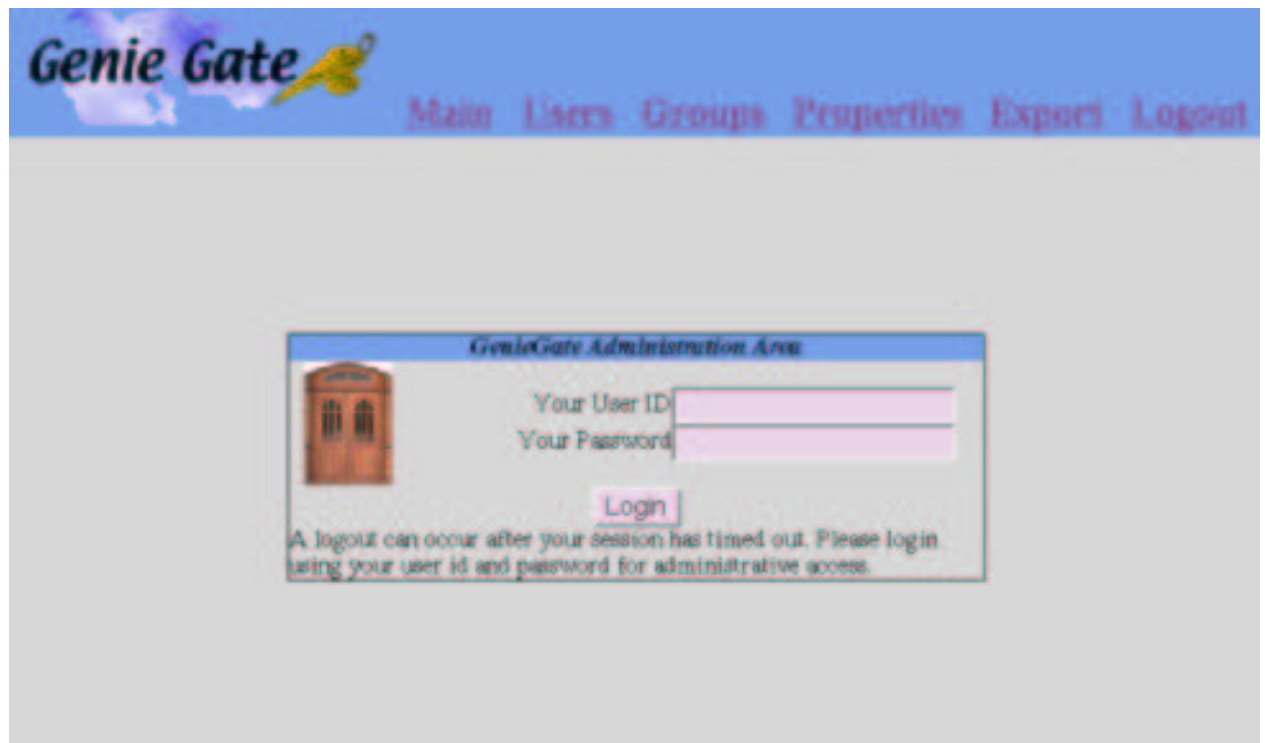
Please feel free to contact us through the contact form at GenieGate (<http://www.geniegate.com/>) we would like to know of any concerns, problems or questions you may have which might enable us to build a better product.

# Chapter 4. Administrative functions

You may safely skip this chapter, it is a reference for the user administrative interface, which is more or less self explanatory.

The administrative interface is GenieGate's "control panel" It allows you to add and remove user accounts, create and modify groups, change passwords, perform queries and other administrative tasks.

## 4.1. Sign On

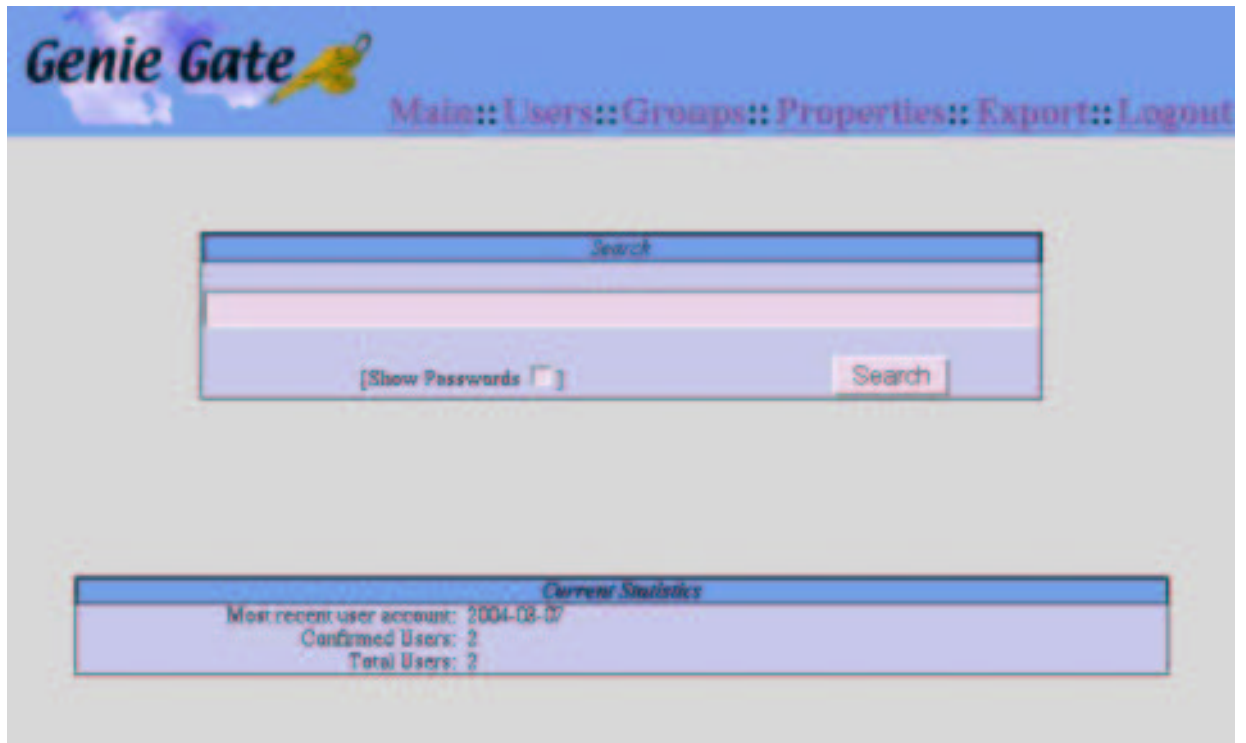


The screenshot shows the GenieGate Administration Area login interface. At the top, there is a blue header with the Genie Gate logo (a yellow key) and a navigation menu with links: Main, Users, Groups, Properties, Export, and Logout. Below the header is a central login form titled "GenieGate Administration Area". The form contains a small icon of a wooden door on the left. To the right of the icon are two input fields: "Your User ID" and "Your Password". Below these fields is a "Login" button. At the bottom of the form, there is a message: "A logout can occur after your session has timed out. Please login using your user id and password for administrative access."

Before you can begin using GenieGate you must *sign in*. Only those who are in the *admin* group may sign in to this form.

When you setup the database you created a special user named *root* who is already a member of the admin group.

## 4.2. Main Screen



This is the main screen, it shows you some statistics about the number of users and provides a "quick search" box where you can search for users based on their names, email addresses or user ID's. (Wild cards are permitted in this form)

## 4.3. User Screen

The screenshot displays the Genie Gate administrative interface with a blue header bar. The header contains the logo 'Genie Gate' with a yellow bird icon and a navigation menu: 'Main:: Users:: Groups:: Properties:: Export:: Logout'. Below the header are three distinct panels:

- Search for Users:** A form with input fields for 'Email', 'Name', and 'Joined'. The 'Joined' field is split into 'From:' (containing '2004-02-27') and 'To:' (containing '2004-03-07'). There is a checkbox for '(Show Passwords)' and a 'Search' button.
- Create New User:** A form with a 'User ID' input field and a 'Create user Account' button.
- Expunge:** A text-based panel with the following text: 'Periodically, you should expunge accounts that have not been validated. (people who did not follow the link in email)'. Below this, it says: 'You may use this function to purge this information, people who are in the process of signing up (today's date) will not be affected.' At the bottom is an 'Expunge old non-validated accounts' button.

This menu allows you to create, query or remove users. It provides an "Expunge old accounts" option. This should be run periodically to clear out the pending accounts. (Situations where people have signed up but did not confirm their email)

The search box is more specific than the one on the main screen, it allows you to query by specific field and to limit the search to accounts created within a range of dates.

### 4.3.1. Create User

The screenshot shows the 'Genie Gate' administrative interface. At the top, there is a navigation menu with links: 'Main:: Users:: Groups:: Properties:: Export:: Logout'. Below this is a form titled 'User Info' with the following fields:

- User ID: sample
- Name: [text input]
- Email: [text input]
- Password: [text input]
- Groups: User can access control panel, use wisely! Members

Below the 'User Info' section is a 'Properties' section with two sub-sections: 'Regular' and 'Internal'.

- Regular:** Telephone Number, Postal code, and Province.
- Internal:** Comments.

After entering a *user id* in the Users menu, this form will be presented allowing you to create a new user.

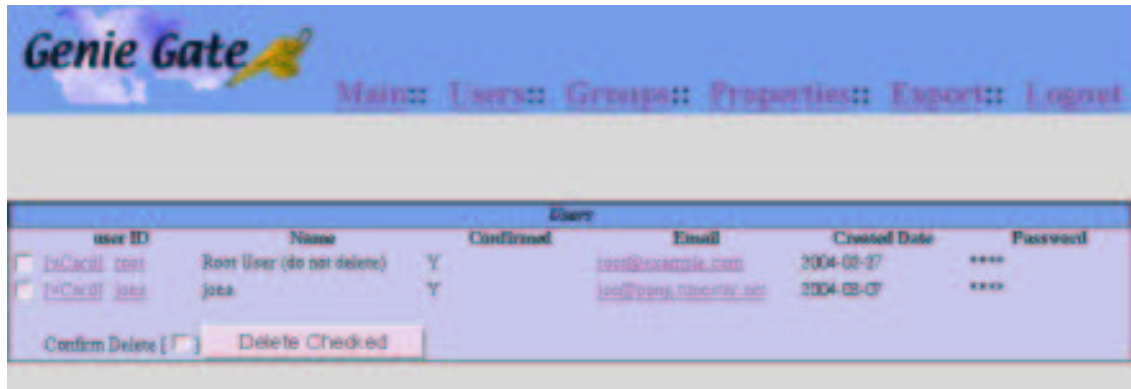
The Groups is a multiple option select box containing any group definitions you may have. Users can be assigned to more than one group, typically by holding the **CONTROL** key while selecting groups with the mouse.

The name, email and password are required fields.

The properties are optional, (in fact, you can define your own properties in the Properties menu)

After you have filled in the information, you may hit the submit button (not shown) to create this user.

### 4.3.2. Search Users



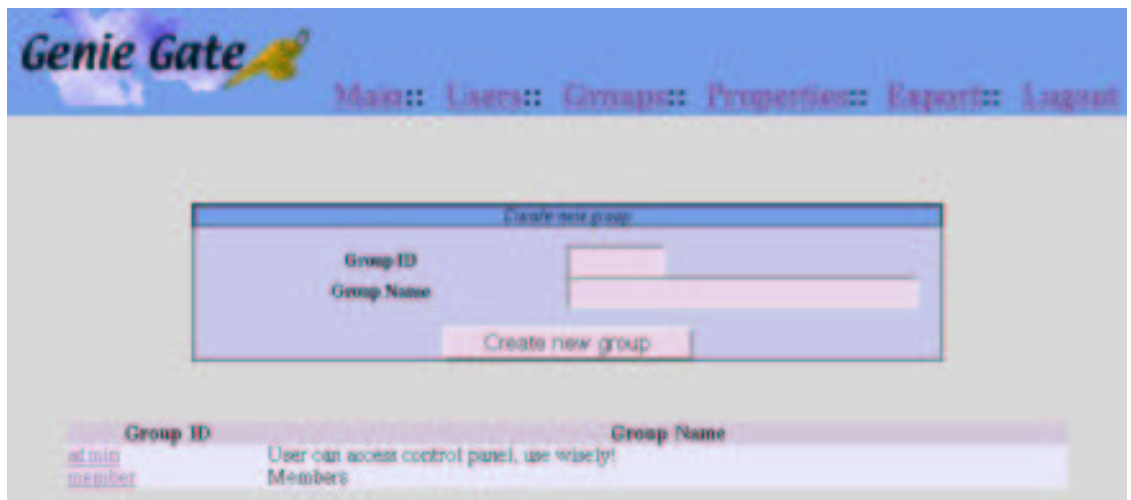
This figure is an example search results page, providing links to a *vCard* and a link to an edit screen.

The *vCard* is a way of exporting this users information into your address and depends on certain key properties to be filled in.

Clicking on the email address allows you to email this user.

By default, passwords are not shown unless the button "show passwords" has been selected on the search box.

## 4.4. Groups



Groups provide access level functionality.

Members may *belong to* 1 or more groups. Access levels may be set depending on which group(s) a given user belongs to. In this way, it is possible to provide restrictions that effectively say *Only admin users allowed here*

You may create, modify or remove groups from this menu. Be aware however that the group ID's may have significant meaning to one or more *plug ins* For this reason, groups should be alphanumeric.

Clicking on a group ID provides a list of members who *are in that group* as well as a form for editing the groups label or removing it.

## 4.5. Properties

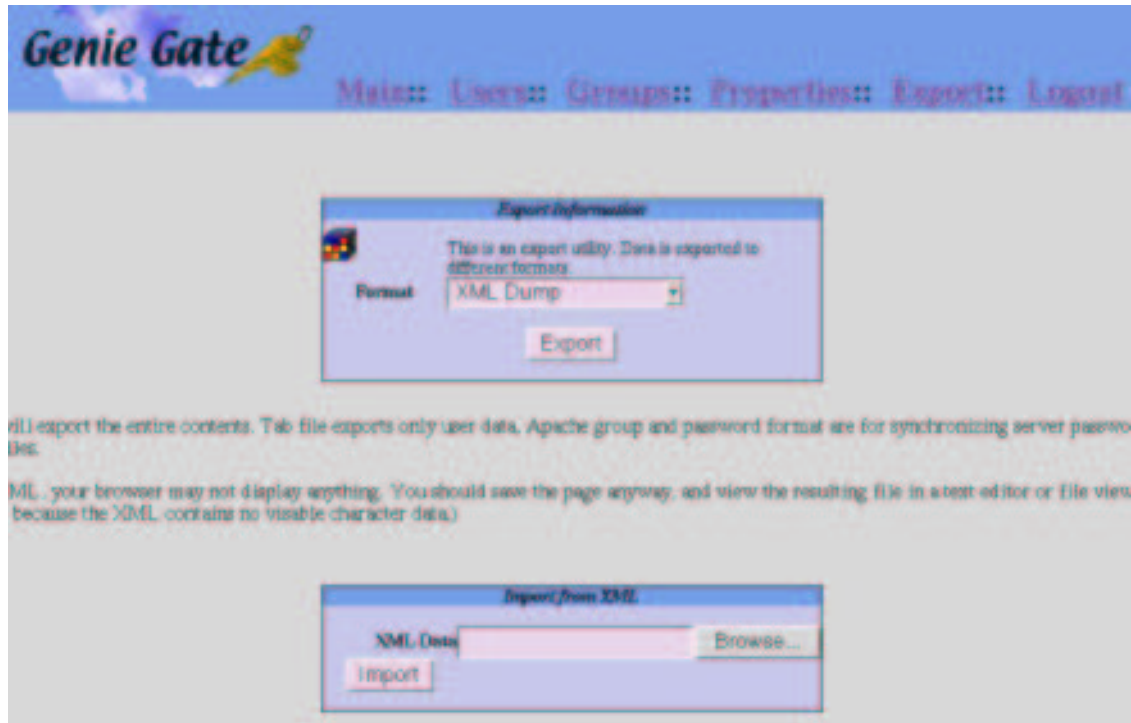
The screenshot shows the Genie Gate web interface. At the top, there is a navigation menu with links: Home, Users, Groups, Properties, Export, and Logout. Below the menu is a form titled 'Create new property'. The form has three input fields: 'Property ID', 'Section' (a dropdown menu with 'Private, Internal' selected), and 'Property Label'. A 'Create new property' button is located below the form. Below the form is a section titled 'Manage Properties'. This section is divided into two columns: 'Public' and 'Internal'. The 'Public' column contains a list of property types: Telephone Number, Postal code, State/Province, Country, First Name, Last Name, FAX number, and City. The 'Internal' column contains a list of property types: Comments and Comments.

Properties are pieces of data associated to users. An example property would be someone's telephone number or zip code.

You may define or remove any properties you wish, however be aware that plugins or other applications may depend on them.

**Creating a new property.** Involves selecting deciding if it should be a *public* or *private* property. Then, make sure to select the right classification. (under section) The public/private status cannot be adjusted.

## 4.6. Export (and import)



GenieGate supports importing and exporting your data. It will export in a variety of formats including apache style groups, passwords and GenieGate's own XML format.

Importing is only supported with XML files.

# Chapter 5. User level functions

This section deals with user level functions. (sign up, password reminder, change password, etc..)

If you are using GenieGate for manual account creations (through the administration interface) and do not wish your users to have such functionality, you can skip this chapter.

## 5.1. The web/user/ files

The web/users/ directory contains PHP code that is designed for users to run. web/users/index.php Could run every function, however for convenience there are several distinct scripts that run particular functions. This provides a way to link to them using conventional HTML anchors.

These scripts (as well as the admin scripts) make assumptions about where your ~/geniegate/ resides. (See: Section 2.1)

## 5.2. User sign up

This section has to do with allowing new members to sign themselves up automatically. The process involves filling out a form, checking their email and returning with a confirmation code.

The file web/users/signup.php contains the URL for this function, (however it does not actually produce the HTML, it is equivalent to /path/index.php?DO\_SHOW\_SIGNUP\_FORM=1 )

The form requires a name, userID, password, email address and confirm address. The form variables are:

This form must be POST (GET won't work for security reasons)

**Table 5-1. Form variables for sign up**

Form Variable	Label	What it does
DO_PROCESS_SIGNUP		Tells the application to process a sign up. (Hidden form variable)
USERID	Users ID to use	
NAME	Name	Supplies the persons name.
EMAIL	Email	Supplies the persons email address.
PASSWORD	Password	Supplies the users password.
PASSWORDC	Password Confirm	Supplies a password for confirmation (must match PASSWORD)

Form Variable	Label	What it does
LOCATION		Optional, if provided, must be an <i>absolute</i> url to display after submit.

After submitting this form, the user is asked to check his or her email for further instructions. An email is sent to the users email address, with a temporary confirmation code and url used to validate the email address. (so it is essential to use valid email addresses when testing this)

When the user checks their email, they should get a message with a url that looks something like this:

`http://www.example.com/path/to/web/users/confirm.php?C=163&U=userid`

**Tip:** The text of this email is configurable by creating your own `mail-confirm.php`

Clicking the link the generated link will *activate* their account, and they will be given the opportunity to login.

The Login page *should be specified* in your `~/geniegate/conf/geniegate.ini` In the [Url] section under `StartPage` By default it is `http://localhost/`. Which is almost certainly not what you want.

## 5.3. Change Password

Users can change their own passwords, the process involves filling out a form with their existing password, creating a new password and repeating the new password (for confirmation)

Like all the other user functions, `web/users/index.php` can be used to run the function, for convenience `web/users/change_password.php` can be used to show the form.

It is assumed that you will want to tie this functionality into an existing web page the form variables for doing so are listed here.

The url the form is posted to is `web/users/index.php`

Note that the only accepted method is POST. (for security reasons)

**Table 5-2. Form variables for password change**

Form Variable	Label	What it does
PASSWORD	Password	Provides existing password.
NEW_PASS	New Password	Revised password
CONFIRM_PASS	Password Confirm	Confirmation

Form Variable	Label	What it does
DO_CHANGE_PASSWORD		Hidden, action directive
LOCATION		Optional, redirection url.

**Tip:** The screen that is shown after a password change isn't very nice looking. In most cases the user should be directed back to what they were doing in the first place.

You can supply a hidden *LOCATION* form variable to a url you would prefer them to be directed to.

## 5.4. Re-send confirmation code

This function allows a user to retrieve their confirmation code, this is useful in the event they've lost their email, or need to be reminded.

The process is similar to password retrieval, in that either *UID* or *EMAIL* form variable can be used to retrieve the information. Instead of being sent a password reminder, they are sent a URL and a confirmation code to re-enable their account.

**Table 5-3. Form variables for confirmation code reminder**

Form Variable	Label	What it does
DO_SHOW_REMIND_FORM		Control directive, displays remind-confno
DO_RESEND_CONFIRM		Control directive, sends reminder.
LOCATION		Optional, url to be directed to.
UID		Their user ID
EMAIL		Their email address

## 5.5. Log Out

**One never really logged in.** First off, one never actually "logs in" to any web page. Through the magic of basic cookies and session management it can appear that way but in fact, this is not the case. (with stateless connections, all web sites have this same caveat)

In GenieGate logout generally means clearing session data that relates to logins residing on the server. It *is not effective* for standard HTTP based authentication (See: Section A.1 for important notes about this)

This form works with the GET method.

**Table 5-4. Form variables for password change**

Form Variable	Label	What it does
DO_LOGOUT		Control directive, only needed for index.php
LOCATION		Optional, url to be directed to.

You may choose `web/users/logout.php` instead of `index.php`. The form variable `DO_LOGOUT` is not required if you do this.

It is *recommended* that you supply a `LOCATION` parameter, this should be a complete url to where you would like the user to be directed to after the logout.

If the logout function does not work for you, it may be due to the HTTP based authentication, (perhaps carried over from the admin function) You may need to restart your browser. If you are certain you are NOT using HTTP authentication, you may have found a bug. Please submit a report so we can get it corrected.

## 5.6. Logins

**The 2 methods of login.** GenieGate supports 2 methods of authentication. One is HTTP based, the other is session based.

HTTP authentication has the advantage of working with almost any web resource; static pages and cgi scripts. It is the standard approach to web based logins and is part of the core HTTP protocol. However, it has the disadvantage in that many people don't like the look of the sign up box. With standard HTTP based authentication you relinquish a certain amount of control; it is impossible to provide a reliable logout function using this approach.

Session based login has the advantage that it gives you more control of the session and can provide more control over what the login box looks like. However, it has the disadvantage of requiring cookies or passing session ID's. It is not as standard, requires co-operation from every web resource and generally doesn't work with static HTML pages.

**Tip:** Apache supports a wide variety of directives. Some of them can be used with PHP to create session based logins for resources. `mod_rewrite` is such a module. See <http://www.apache.org/> for more details on what apache can provide.

To use session based authentication with GenieGate you need to access it on an *API* level. This is not quite as difficult as it seems, it is slightly more involved but is generally more suitable.

The `web/users/index.php` (with no parameters) uses the session based approach, while `web/users/login.php` uses HTTP based and then defaults to session based if the user hits cancel on the password dialog box. (If `index.php` is passed `DO_LOGIN=1` it is equivalent to `login.php`)

**Tip:** A much better approach to logging in is to use GenieGate on an *API level*. You'll get much better control over the process that way. See `web/example/login/` for an easier approach that gives you better control over what the forms look like.

**Table 5-5. Form variables for Login**

Form Variable	Label	What it does
DO_LOGIN		Directive, optional with login.php
LOCATION		Optional, redirect after login.
GG_USERID	User ID	User ID to login as.
GG_PASSWORD	Password	Users password

You may notice that the page shown after login is not particularly appealing or useful. It is recommended that you provide 1 or more login forms, with the LOCATION set to where you would like your visitor to go after signing on.

## 5.7. Forgotten Passwords

The process of reminding a user of his or her password involves filling out a form with their email address OR their user ID, GenieGate will then lookup their email address and send the password to the email address they used *when signing up*

This function works only with the POST method, the correct script is `web/users/index.php`

**Table 5-6. Form variables for Password reminder**

Form Variable	Label	What it does
DO_SEND_PASSWORD		Directive, what to do.
LOCATION		Optional, redirect page.
UID	User ID	Users ID
EMAIL	Email	Accounts email address

Either the user ID or the email address can be used. (In the event the user forgot his/her userID.) It is not an error to supply both. If they do supply both, first the userID is checked and if not found, the email address is checked.

It is important to note that it *will not* send the account information to any address they type in. The email address is used only to lookup their account information.

**Tip:** The email that is sent to them can be tailored by creating `~/geniegate/lib/views/users/mail-password.php` and changing it.

## 5.8. Personal settings

GenieGate has the ability to store and maintain arbitrary personal settings. Typical uses are postal addresses, zip codes, favorite colors and things like that.

Personal settings are used in GenieGate's *vCard* export implementation.

If you are not using personal settings, you may skip over this section.

Unlike other profile managers, *you do not need to know in advance* which settings are required. You may add them through the administration interface.

This section covers properties from the perspective of user access.

GenieGate distinguishes between *internal* and *external* properties. External properties may be changed by the user, while internal properties are reserved for the administrators use.

Furthermore, GenieGate allows multiple classifications of properties on an *API* level. This provides 3rd party applications and custom software with an ability to store properties unique for it's purposes separate from the main ones used by GenieGate.

**About `web/users/properties.php`.** There is a provided script that shows the HTML required to set properties, it primarily serves as an example of the required HTML, and should probably not be used in a production web site. (It shows ALL public properties on one page in no particular order)

There is another file in `web/example/forms/properties.php` that demonstrates a more uniform, orderly approach. The example enables a user to view and change their address. It needs to be edited before it can work. (`$LIB` and `$POST_URL` will need to be set to their proper locations)

**Table 5-7. Form variables for changing properties**

Form Variable	What it does
<code>DO_UPDATE_SETTINGS</code>	Directive, tells <code>index.php</code> to update the personal settings.
<code>LOCATION</code>	Optional, redirected page after completion.
<code>PROP:propId</code>	The property ID to change. <code>PROP:propId</code>

The `PROP:propId` may need further explanation; anything after the "PROP:" is interpreted to mean a property ID. So, to set the zip code, one would use `<INPUT NAME="PROP:postalCode" ... >`

# Chapter 6. Customization

A large problem with complex web sites and infrastructure is the lack of a *consistent* user profile area. When working with multiple web applications you are often forced to maintain several sets of username/password combinations.

GenieGate is not just a PHP application, it also contains a programmers API and was designed for complete customization through the use of templates and the *model-view-controller* design pattern.

## 6.1. About the HTML

Wherever possible, a hidden LOCATION parameter can be used to determine where the user is directed after completing something. For the most part, editing the HTML in GenieGate should not be required. In the cases where it is desired, the design supports something known as "views" this is more or less a fancy way of saying it's all template based.

Editing the templates directly is *strongly discouraged* instead, you should copy the template to `~/geniegate/lib/GenieGate/views/users/` (or `~/geniegate/lib/GenieGate/views/admin/`) and edit it there.

Doing this will allow you the ability to run multiple copies of GenieGate and shield you from loosing changes during a possible upgrade. (See: Section 2.1 for a more detailed explanation.)

The default template set contains the filename at the bottom of the screen, this can be used as a guide for which template to edit.

# Chapter 7. User Templates

The output from GenieGate can be altered by editing templates. (Or in many cases, supplying a LOCATION variable to a desired page)

## 7.1. shell.php - Shell template

The other templates are "wrapped" in the `shell.php` template, This is the proper place to remove the lower Page: `GenieGate/views/users/new-user.php` piece of information. (It is provided for your reference)

## 7.2. Overview of the templates

You may choose to skip over this portion, it is provided as a reference. The easiest way to customize the templates is to simply locate the screen you wish to modify, view the information displayed at the bottom and use that as a guide.

Each template is displayed on the page it appears. However, if you have modified your `shell.php`, this information may not be visible to you. Simply renaming the copy you made at `~/geniegate/lib/GenieGate/views/users/shell.php` to a temporary value should re-enable this information.

- `GenieGate/views/users/account-confirmed.php`

Shown after the account has been confirmed.

- `GenieGate/views/users/change-password.php`

Show the screen for changing a password.

- `GenieGate/views/users/check-email.php`

Show the message explaining further instructions have been sent.

- `GenieGate/views/users/error-confirm.php`

Shown when there is an error during the confirm stage.

This is expected to be fairly common, provide a form for them to manually key in their userID and confirmation code.

- `GenieGate/views/users/error-general.php`

Show an error.

- `GenieGate/views/users/error-signup.php`

Shown when there is a problem with the new user signup form.

- `GenieGate/views/users/logged-out.php`

Page is shown after logging out.

- `GenieGate/views/users/main.php`

This is the main menu for the persons account.

- `GenieGate/views/users/new-user.php`

New user page.

- `GenieGate/views/users/password-changed.php`

This is shown after a user changes their password.

- `GenieGate/views/users/sent-password.php`

User password has been emailed to them.

- `GenieGate/views/users/set-properties.php`

This view is shown after a user sets properties.

- `GenieGate/views/users/shell.php`

Shell wrapper for the interface, this is a handy place to put HTML that appears globally. (headers, footers, etc)

See the beginning of this section for an explanation of how to use it.

- `GenieGate/views/users/show-email-password.php`

I forgot my password form.

- `GenieGate/views/users/show-properties.php`

Very generic "properties" menu, shows all the users properties and provides a form for them to edit them.

- `GenieGate/views/users/stylesheet.php`

This is the stylesheet included in shell.php

- `GenieGate/views/users/user-signup.php`

This shows the user signup form.

- `GenieGate/views/users/remind-confno.php`

Displays form for sending confirmation code reminder.

- `GenieGate/views/users/check-email-remind.php`

Screen shown after confirmation code had been sent.

## 7.3. email templates

Email is also template based.

**Confirmation Email.** The file `GenieGate/views/users/mail-confirm.php` creates the text of the information emailed to a new user instructing them to confirm their account.

**Forgotten Password.** The text of the email sent to a user when he or she chooses to be reminded of their password is in `GenieGate/views/users/mail-password.php`.

**Confirmation code reminder.** The text of the email sent to a user when he or she chooses to be reminded of their confirmation code `GenieGate/views/users/mail-confirm-remind.php`. By default, it simply includes *mail-confirm.php*

# Chapter 8. Programmers API

GenieGate is not just an application, it also provides a complete API for programmers to use. Through this API it is possible to store and access properties, perform authentication and listen for *events*. (it is also possible to do a great deal more, you should consult the api documents if you are interested)

People interested in using GenieGate for basic authentication may skip this chapter. However, it should be reassuring to know that it is available should your needs grow beyond what is already provided.

The actual documentation for this API is not covered here, it is generated by a free tool called doxygen (at the time of this writing, doxygen does not support the DocBook format)

## 8.1. Login via the API

The PHP files in your `web/examples/login` directory demonstrate a PHP based approach to password protecting PHP documents. Both form based and HTTP authentication based methods are demonstrated.

**Using the examples involves configuring.** The location of `~/geniegate/conf.php` needs to be set before the examples can be used.

After you've altered them with the correct file location, you may try them out. `basic.php` provides the HTTP (with fall-back to a builtin form) method. It involves adding 1 line to a PHP document you wish to protect.

The `2liner.php` is slightly more elegant, it demonstrates how to use a form based approach, with a *user defined* form for the user to fill out.

## 8.2. "Passive" login

Passive logins are useful in situations where you only need to protect portions of a document, wish to provide separate versions of the same document depending on whether or not a user has signed on.

A classic example would be providing a sign-on form to people who are not signed in.

Passive logins are also useful in situations where you require access to the *properties* of a given user (such as addresses or names) but do not wish your page to fail if he or she is not already a member.

The example `web/example/login/passive.php` demonstrates this functionality. Just as the other examples, it needs to be configured prior to using it.

## 8.3. Plug ins

If you are using Apache style passwords, then you have come into contact with plugins. They involve adding an entry to your `PLUGINS` and optionally setting up some other configuration sections. (This is done in `~/geniegate/conf/geniegate.ini`)

Plugins are php classes that "listen" to events such as new user creation or account changes. They are typically designed to maintain synchronization with other applications that may need to know about these events.

To create a plug in, you should extend `GenieGate_Api_Plugin` and provide methods for the events you are interested in.

**The logger plug in.** For a simple example plug in, `GenieGate_Api_Plugin_Logger` is a good place to start. It will log each event (and it's parameters) to a file. It is a useful plugin to assist in determining those events you are interested in.

## 8.4. Creating/Modifying the GenieGate libraries

Generally you should never need to modify the libraries of GenieGate. Doing so is not officially recommended, but it is possible with minimal hassles.

Please do NOT change the files in the global `lib/` directory. You should copy them to your `~/geniegate/lib/` directory and edit them there.

# Appendix A. Caveats and things to note

## A.1. HTTP Authentication and Log outs

When using basic authentication apache style (or any web server using basic HTTP authentication) the login credentials are stored on the *browser* this has the unpleasant side effect that it is simply not possible to provide an *effective* "logout" function.

The only way to actually log out of such a site is to shut your browser down or if the browser supports it, clear the login credentials. (Even this is not always reliable because the browser may not know which cookies/credentials/form variables to clear)

Workarounds for this style of authentication include things like sending the client browser a header stating their credentials were not accepted, but the user could simply cancel and sign on to another page anyway.

When using sites that use this type of authentication it is usually best to restart the browser when done, to ensure your password is not preserved. (even then, some browsers may choose to cache this information)

If a logout function is imperative you can restrict your application(s) to using GenieGate on an API level, this will work (since the login credentials are stored on the server).

There is nothing server-side that can deal with this problem reliably. If a web application claims to offer this functionality (with HTTP authentication) it is *strongly recommended* that you question it. There is almost certainly a way to undermine it.

# Appendix B. vCard implementation

vCards are a common interchange format for PIM's and address books, GenieGate supports downloading of vCard's for users in the search result page.

vCards have only been tested with the Linux GnomeCard program. It should work with most any PIM since vCards are a standard file format.

In order for vCards to work, certain properties need to be available. (Unset properties are simply skipped over, but the property definition must exist.)

**Table B-1. vCard fields and properties**

Property/Field	Description
Fields	
email	Email address
name	Users name
Public properties	
tel	Telephone number
street	Street address
city	City
state	State / province
postalCode	country
fname	First Name
lname	Last name
fax	Fax number
title	Job title
organization	Organization/Company
homePage	Home Page
dob	Date of Birth
wphone	Work phone
Private properties	
comments	Personal comment field

You may need to set your browser up to handle the mime type of text/directory to work with your favorite PIM.

# Appendix C. Plugins

If you are using basic apache authentication, you are already familiar with GenieGate's plug-in model.

How you use a plugin will vary on what it does, but it will involve editing your `~/geniegate/conf/geniegate.ini` so that GenieGate will utilize them.

The critical configuration variable is `PLUGINS` which contains a colon delimited list of plugin PHP files.

**Tip:** You should remove any plugins you are not using because they could slow down your web site.

## C.1. ApacheDBM large scale htaccess.

Apache supports what is known as *DBM* authentication. This type of authentication is ideal for high traffic web sites where a large number of users and/or a large number of requests pr. minute are common.

Be aware, however that both PHP AND Apache must be compiled with the correct *dbm support* in order for this plugin to work.

### Example C-1. Configuring for ApacheDBM with GNU dbm support

```
PLUGINS=GenieGate/Api/Plugin/Apache.php:GenieGate/Api/Plugin/ApacheDBM.php  
[ApacheDBM]
```

```
; in your .htaccess set AuthDBMGroupFile and AuthDBMUserFile to the same  
; filename.  
;  
AuthDBMFile=/home/joe/geniegate/conf/apache/authenticate.dbm
```

```
; The DBM Handler, this will vary depending on what is supported on your system.  
; See: http://www.php.net/manual/en/ref.dba.php for what the values mean.  
;  
; Options are gdbm,dbm,ndbm,db2,db3  
; (NOTE that the value selected has to work with both Apache AND php)  
DBMHandler=gdbm
```

# Appendix D. XML export/import

GenieGate supports importing and exporting of *XML* format files. These are given the mime type of `application/x-geniegate-xml`.

Many other user management applications support features that aren't directly related to the management of user access. Things such as newsletters, photographs, emails, etc..

XML is GenieGate's answer to desired features that are batch oriented in nature. One could, for example use the XML to produce a list of users who have the property "WantEmail" to produce a monthly newsletter. Another application might apply a stylesheet that generates an HTML report in a format that you find useful. It works the other way around, too. If you have a list of users in some other format and wish to give them access, the answer is in getting that format into the type of XML which GenieGate understands.

## D.1. Overview of the format

The format consists of group definitions, property definitions and users. The best way to become familiar with it is to view one of the files that have been exported.

### Example D-1. Example XML document

```
<!DOCTYPE GenieGate SYSTEM "geniegate.dtd">

<GenieGate>
<GroupDefinitions>
<GroupDefinition gid="test" name="My test group" />
</GroupDefinitions>

<PropertyDefinitions>
<PropertySection id="com.example.PhotoAlbum"
  title="Properties of a Photo Album">
<PropertyDefinition property="photo" title="Photograph URL" />
</PropertySection>
</PropertyDefinitions>

<Users>
<User uid="test" name="Test User" password="testing" email="test@example.com" confirm="Y">
<Groups>
<Group gid="test" />
</Groups>
<UserProperties section="com.example.PhotoAlbum">
<UserProperty property="photo">
http://home.example.com/photo.jpg
</UserProperty>
```

```

</UserProperties>
<UserProperties section="genie.form.Public">
<UserProperty property="city">Any Town</UserProperty>
</UserProperties>
</User>
</Users>

</GenieGate>

```

Here we've created a group called " test ". We've also created a *property section* called `com.example.com.PhotoAlbum`. (Presumably this is an application using GenieGate on an API level to manage it's own set of user-level properties.) It contains the property "photo".

We've also created (or updated) the user with the id of test. We've given this person access to the "test" group. Since we've defined the property "photo" for our hypothetical photograph album, we can store the url to this persons online photograph.

We've also set this persons "city" to "Any Town" the property is in the "genie.form.Public" section. Since this property is already defined, we can go ahead and use it.

## D.2. The DTD for the XML format

```

<!-- Root element -->
<!ELEMENT GenieGate (GroupDefinitions?,
PropertyDefinitions?,
Users?)>

<!ELEMENT GroupDefinitions (GroupDefinition*)>
<!ELEMENT GroupDefinition EMPTY>
<!ATTLIST GroupDefinition gid NMTOKEN #REQUIRED>
<!ATTLIST GroupDefinition name CDATA #IMPLIED>
<!-- Flag not used -->
<!ATTLIST GroupDefinition signup (Y|N) #IMPLIED>

<!ELEMENT PropertyDefinitions (PropertySection*)>
<!ELEMENT PropertySection (PropertyDefinition*)>
<!ATTLIST PropertySection id NMTOKEN #REQUIRED>
<!ATTLIST PropertySection title CDATA #IMPLIED>

<!ELEMENT PropertyDefinition EMPTY >
<!ATTLIST PropertyDefinition property NMTOKEN #REQUIRED>
<!ATTLIST PropertyDefinition title CDATA #IMPLIED>

<!-- Container for Users, this should be in another file. -->
<!ELEMENT Users (User*) >

```

```
<!-- Container for a user. -->
<!ELEMENT User (Groups?,UserProperties*)>
<!ATTLIST User uid NMTOKEN #REQUIRED>
<!ATTLIST User email CDATA #REQUIRED>
<!ATTLIST User password CDATA #REQUIRED>
<!-- YYYY-MM-DD -->
<!ATTLIST User created NMTOKEN #IMPLIED>
<!ATTLIST User confirm (Y|U) #REQUIRED>
<!ATTLIST User name CDATA #IMPLIED>

<!-- Groups user is a member of. -->
<!ELEMENT Groups (Group*)>
<!ELEMENT Group EMPTY>
<!ATTLIST Group gid NMTOKEN #REQUIRED>

<!-- Properties grouped by a section. -->
<!ELEMENT UserProperties (UserProperty*)>
<!ATTLIST UserProperties section NMTOKEN #REQUIRED>

<!-- Only element that actually uses #PCDATA. (Since a property could
in theory contain multiple lines) -->
<!ELEMENT UserProperty (#PCDATA)>
<!ATTLIST UserProperty property NMTOKEN #REQUIRED>
```

# Appendix E. License

**Tip:** The main purpose of this license is the disclaimer. In general, just use common sense, meaning don't redistribute it or hold me responsible for any damages. If you are using the free version, you must do some advertising. That means the links must be provided on every signup and login form, with the text "User Management Solutions" as the text of the link.

Web designers or virtual host providers may share the software among multiple clients. (this was, after all, the whole *point* of the ~/geniegate/ directory structure)

The networked file systems clause is provided to make it possible to run multiple copies on ISP's that have several machines networked together. The firewall portion is intended to prevent this clause from being abused.

You may use it for free provided the link <A HREF="http://www.geniegate.com/">GenieGate User Management Solutions</A> anywhere a login box and signup form exists. Additionally, the tagline that is sent out with email messages must also be present.

Copyright © 2003-2004 Jamie Hoglund. All rights reserved.

Purchaser or Partner (Licensee) are granted permission to use the software ("software") under the following conditions:

Redistribution in source or binary forms, with or without modification, is not permitted.

Free versions of GenieGate must provide the link <A HREF="http://www.geniegate.com/">GenieGate User Management Solutions</A> on each login form and signup form. Additionally, the last 3 lines sent in email must not be modified.

Software may be used by multiple person(s) on the same computing platform ("computer") provided that such persons are directly associated to Licensee in a business endeavor, this includes, but is not limited to clients, customers, or employees.

Multiple copies may be used on computers networked in such a manner as are transparent to Licensee, such as networked file systems, where proper software operation is not possible without such network installed, software may be used on multiple computers. All hardware in such an environment must reside behind the same firewall. (generally reside in the same "intranet")

Software may be copied for development or testing by licensee or employees of licensee provided the other conditions in this license are satisfied and such installations are used exclusively for software development.

**THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY**

AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.

IN NO EVENT SHALL THE Jamie Hoglund, OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*Portions of this license borrowed from the Apache license. See: <http://www.apache.org/>*

# Appendix F. MySQL tables

This briefly covers the tables in use by GenieGate.

You'll notice they all begin with the letters *ua\_* this should ease problems sharing GenieGate with other applications on the same database, *unless* the other applications happen to use tables that begin with *ua\_*.

**Table F-1. MySQL tables**

Table	Purpose
ua_group	Group definitions and names
ua_members	Links users with groups.
ua_psect	Property sections
ua_prnam	Property definition
ua_pnam	Property definitions
ua_prop	Stores user properties.
ua_users	User accounts

The actual fields aren't covered here, you should look at the tables directly to see what they are. (if you're reading this, you're probably already doing that)

Most of it is fairly straight-forward. However, multiple sections are supported for properties and might warrant further explanation. *ua\_psect* contains the section definitions. This table contains the field *sid* which maps to a numeric *section ID*. This is used for categorizing properties into sections.

The table *ua\_pnam* contains the actual property names. These have *prid* and *sid*, *sid* is joined with *ua\_psect*, to obtain the section name. The field *prid* is a globally unique number for this property.

Finally, *prid* is used to retrieve the actual users property from *ua\_prop*

The purpose of doing it "the long way" is that numbers can be indexed and stored much more efficiently than strings, it would not make sense to store a long property name several times over for each user.