Software Built With You In Mind

MGI

page planet software inc.

MGI 2 Guides

MGI Admin Guide

In the MGI Admin Guide, you will find information about the system requirements for running MGI, procedures for installing MGI, the MGI server admin, the MGI domain admin, and applicable licensing agreements.

MGI User Guide

In the MGI User Guide, you will find the Understanding MGI guide to MGI basics, the Using MGI step-by-step tutorials, and the Referencing MGI technical tag syntax, description, examples, and suggested usage for each MGI tag.

MGI Admin Guide

In the MGI Admin Guide, you will understand the system requirements to run MGI, how to install MGI, and how to use the web-based MGI server and domain admins.

MGI System Requirements

Macintosh

To successfully run MGI on Macintosh, we recommend the following:

- 4D's WebSTAR 3.02 or higher (WebSTAR version 4.4 is recommended).
- System 8.1 or higher (9.1 is recommended).
- Minimum 5 MB of free RAM with additional allocation as needed for additional concurrent threads. Since MGI is allocated memory directly from the web server according to the number of concurrent threads available for processing, the best performance will be achieved with sufficient memory dedicated to the web server. We recommend a minimum of 80 MB of RAM total for the web server.
- Minimum of 5 megs free hard drive space. Additional hard drive space needed for files created by MGI such as databases in the MGI Files folder.

Windows

To successfully run MGI on Windows, we recommend the following:

- Windows NT 4.0 Server running Internet Information Server 4.0 OR Windows 2000 Professional or Server running Internet Information Server 5.0
- Sufficient RAM for IIS.
- Minimum of 5 megs free hard drive space. Additional hard drive space needed for files created by MGI such as databases in the MGI Files folder.

[System Requirements] [Installation] [Domain Configuration] [Server Admin] [Domain Admin] [Licensing]

[MGI Guides Main Menu] [Admin Guide Main Menu]

MGI Installation Procedures

WebSTAR 3.x/4.x Configuration on Macintosh

1. Quit WebSTAR if the application is running.
2. Increase the memory allocation for the WebSTAR server. We recommend a total of 80 MB of RAM allocated to WebSTAR (increase memory by 5 MB minimum).
3. Double-click the MGI VISE installer.
   - Click continue.
   - Read and accept the MGI License Agreement.
   - If multiple copies of WebSTAR are installed on your server, select the copy of WebSTAR where MGI should be installed.
   - Enter the owner name (User name), owner organization, and serial number/authorization code. Enter the serial number and authorization code without spaces. Your existing owner name, owner organization and serial number/authorization code are pre-filled from your MGI preferences. If the serial number/authorization code is left blank, MGI will install in demo mode. Demo mode displays a header and footer on each page that is processed and served by MGI.
   - Enter your Username and Password for the MGI Server Admin. Your existing username is pre-filled and your existing password, although not pre-filled, will be used if another password is not entered. If the password is left blank during an original installation, the password defaults to "MGI" (case-sensitive).
4. Restart the WebSTAR application. Check the log window to make sure the MGI modules were initialized properly.
5. Launch the WebSTAR Admin application and open the Server Settings.
   - Under the "Suffix Mapping" option, configure the suffix mappings for any suffix that you want MGI to process and serve. You can have any suffix such as ".mgi", ".mgi2", ".html", or ".htm" processed by MGI. However, only one processor such as MGI can process any given suffix. In addition, you may have better server performance overall if MGI only processes pages that contain MGI tags rather than every page. The settings for an MGI-enabled suffix should be as follows: Action (MGI2), Type (*), Creator (*), MIME Type (text/html). Note that the ".MGIMODULE" suffix is pre-configured. Click the "Save" button to save the suffix mappings.
   - If you wish to use MGI to determine the default index pages for your server (i.e., to serve an index page when one is not specifically given in a URL request such as "http://www.domain.com/" actually serves "http://www.domain.com/index.mgi"), enter a page name with an MGI suffix as

the Default index name under the "File Names" option. For example, if the ".mgi" suffix is mapped to MGI2, then you might enter "index.mgi" as your default index page in the Server Settings. Click the "Save" button to save the settings.

  - We do not recommend that you run caches on a production server. We recommend that you turn all cache settings off in the WebSTAR Admin. Click the "Save" button to save the settings
  - Quit the WebSTAR Admin.

6. Set up web site accounts as usual. That is, put a web site folder or alias to a web site folder in the WebSTAR root folder, configure virtual hosts for domain accounts using an application such as WebSTAR Virtual Host or Welcome, and configure related services such as FTP and secure services.
7. If accounts are not set up in the MGI Server Admin, then any domain using MGI functions will do so under the Default Domain. The consequences of multiple accounts using MGI functions under the Default Domain is shared database access, shared module access, etc. To avoid shared access, set up each web site in the MGI Server Admin. We also recommend that you disable modules under the Default Domain if you are running a shared server environment (e.g., hosting services, etc.). Read additional instructions for setting up domains and regions in the MGI Server Admin.

IIS 4.0/5.0 Configuration on NT

1. Double-click the MGI VISE installer.
   - Click continue.
   - Read and accept the MGI License Agreement.
   - Enter the owner name (User name), owner organization, and serial number/authorization code. Enter the serial number and authorization code without spaces. Your existing owner name, owner organization and serial number/authorization code are pre-filled from your MGI preferences. If the serial number/authorization code is left blank, MGI will install in demo mode. Demo mode displays a header and footer on each page that is processed and served by MGI.
   - Enter your Username and Password for the MGI Server Admin. Your existing username is pre-filled and your existing password, although not pre-filled, will be used if another password is not entered. If the password is left blank during an original installation, the password defaults to "MGI" (case-sensitive).
2. Open the Internet Services Manager. The following steps may be made to the Master Properties which apply to the entire server or may be made to the properties of selected web sites.
   - Open the Master Properties or the Properties of a selected web site.
   - Click the "Home Directory" tab (where the virtual root directory is configured). Click the "Configuration..." button. This is where you configure the extension

mappings for MGI. First, add the required extension mapping for ".mgimodule". To add an extension mapping, click the "Add" button. In the "Executable" field, enter or browse the path to the file "mgi2.dll". **Note: The path CANNOT have spaces in it.** In the "Extension" field, enter the suffix that is mapped to MGI, ".mgimodule" in this case. Leave all other settings as they default and click the "OK" button to add the extension. Add additional extension mappings for any extension you want MGI to process and serve. You can have any extension such as ".mgi", ".mgi2", ".html", or ".htm" processed by MGI. However, only one processor such as MGI can process any given extension. In addition, you may have better server performance overall if MGI only processes pages that contain MGI tags rather than every page. Click "OK" to save the extension mappings.

- Click the "Documents" tab. Check the "Enable Default Document" checkbox to list default index pages that will server for all web sites or the selected web site. To add a default page name, click the "Add" button, enter the default index page (e.g., "index.mgi") and click the "OK" button. Default pages are served and listed in the order entered. To rearrange the order of default index pages, select a default index page and use the up and down arrow buttons.
- Click the "Directory Security" tab. Click the "Edit..." button in the "Anonymous access and authentication control" section. Leave "Anonymous access" checked, but uncheck all others (basic, NTLM, digest authentication, and especially Integrated Windows authentication). This is required for MGI to be able to authenticate the MGI Server Admin, the MGI Domain Admin and any page with MGI authentication tags. IIS will intercept the authentication before MGI gets a chance to process it if any of these options are enabled. Click "OK" to save the authentication settings.
- Although virtual hosts are used by MGI to detect which domain settings to use, you must still configure virtual hosts in the Internet Services Manager for each web site. As usual, virtual hosts are configured by clicking the "Web Site" tab. On the "Web Site" tab, click the "Advanced..." button to view the virtual host list. To add a virtual host, click the "Add..." button, enter the TCP Port as "80" and enter the virtual host name (e.g., "www.domain.com") in the "Host Header Name" field. Click the "OK" button to save the virtual host. Click "OK" again to save the Advanced Web Site settings.
- To save all changes in the Master Properties or selected domain properties, click the "Apply" button and close the window or click the "OK" button.

3. If accounts are not set up in the MGI Server Admin, then any domain using MGI functions will do so under the Default Domain. The consequences of multiple accounts using MGI functions under the Default Domain is shared database access, shared module access, etc. To avoid shared access, set up each web site in the MGI Server Admin. We also recommend that you disable modules under the Default Domain if you are running a shared server environment (e.g., hosting services, etc.). Read additional instructions for setting up domains and regions in the MGI Server Admin.

Installing Module Updates

During the course of MGI development, PagePlanet Software releases updates to modules which contain bug fixes and new features. You can download module updates from the PagePlanet Software web site or via the Version Tracker feature of the MGI Server Admin.

Use these instructions to install a new module.

1. Download the module from the PagePlanet Software web site or via Version Tracker.
2. Transfer the module to the server and uncompress the module file.
3. Access the MGI Server Admin.
4. Under Module Management, unload the module you are replacing.
5. Copy the new module into the MGI File/Modules/ folder.
6. Under Module Management in the MGI Server Admin, reload the module. The new module will be active on your server as soon as it is reloaded.

[System Requirements] [Installation] [Domain Configuration] [Server Admin] [Domain Admin] [Licensing]

[MGI Guides Main Menu] [Admin Guide Main Menu]

Configuring Domains and Regions in the MGI Server Admin

Before proceeding, you should have installed MGI and set up any web site account on your machine (including any entries for virtual hosting).

The MGI Server Admin has other features such as logging and security, however the instructions below only pertain to allowing a web site access to MGI modules.

1. Open the MGI Server Admin

To use the MGI Server Admin, access the "MGIServerAdmin.XXX" page (where XXX is any suffix ir extension mapped to MGI2) in a web browser using any virtually hosted domain on the server (including the server name or IP address). For example, "http://www.domain.com/mgiserveradmin.mgi".

2. Click the "Domain Management" Button.

3. Create an MGI Domain for Each Web Site

Virtually hosted domains are configured differently from non-dns accounts (i.e., folders at the root of WebSTAR that do not have a domain name).

For Each Virtually Hosted Domain (i.e., Accounts with a Domain Name):

a. Add a Domain. Under "Create new domain" enter the web site's main domain name (e.g., "domain.com") and click the "Create..." button. This name is simply a label for the web site so that you can recognize the web site in the Domain Management list.

b. Configure Domain Preferences. Select the domain in the Domain Management list and click the "Edit" button.

- **Virtual Hosts**. In the text box under Domain Preferences, enter all virtual hosts that resolve to the selected web site (e.g., "domain.com", "www.domain.com", and "third.domain.com"). Enter each host on one line with a return. Be sure you enter the virtual hosts for all domains that resolve to the selected web site including redirects. Click the "Save" button. Preference changes take effect immediately.
- **Module Access for the Domain.** At the bottom of the Domain Preferences window the modules available for the web site are listed. The module list does not reflect modules that are disabled at the server level (under Module Management in the MGI Server Admin). By default the web site has access to all modules. To disable the web site's access to a module, click the "Disable" button beside the module. When disabled the module name displays in plain text

with a line through the name. To enabled a module, click the "Enable" button beside the module. When enabled the module name displays in bold text.

- **Default Domain**. If you are running a shared hosting environment, we recommend that you disable all modules in the Default Domain. This will disallow unauthorized use of MGI on your server and prevent unintended shared databases and other access.

c. Add Regions or Use Default Region. Click the "Regions" link beside "Domain Preferences" to view the "Region Management" list. Regions specified in the Region Management list restrict file paths within the region to the root of the region (i.e., a file path cannot walk "up and out" of a defined region) and restrict database access and other settings to the region. Regions can be created to separate access and settings between sub-folders of a web site. For example, if you need to separate database access for different divisions of a company that are sub-folders of a web site, you could create a region for each division. For the majority of web sites, shared access within a domain is preferred and allowed, therefore the default region (/ - the root level) is used.

To add a region, enter the region (sub-folder) path under "Create new region" and click the "Create..." button. The region path is based on your virtual host mapping. A region is any sub-folder under the mapping level of the domain for the region. For example, if your virtual host "domain.com" maps to the folder "/domain/" and you want to create a region for "/domain/shop/", then you would enter "shop" as the region path. However, if you have the domain "shop.domain.com" mapped to the folder "/domain/shop/", then the "shop" folder is actually the root level ("/") of that domain.

NOTE: If you are using a virtual host application that modifies the URL rather than maintaining separate virtual roots (such as **Welcome in normal mode or HomeDoor** on a Macintosh), you must prepend the region name with the host folder path. For example, the Welcome admin requires a "host folder" for each virtual host. If the host folder for a domain is "/websites/" then the non-dns account region name should be entered as "/websites/accountFolderName/". If you are running a "true" virtual hosting system (such as WebSTAR Virtual Host or Welcome in router mode on a Macintosh or an NT IIS virtual hosting system), there is no need to prepend information to the region name since the URL is not modified.

d. Configure Region Preferences. Settings for the default region will apply to any region (sub-folder) in the web site that is not configured as a separate region. To configure a region's preferences, select the region in the list and click the "Edit" button.

- **Remove Unprocessed Tags**. By default MGI tags that are not recognized or that are not processed are removed from pages before they are served. To leave unrecognized or unprocessed tags in pages, check the box beside "Do Not Remove Unprocessed MGI Tags".

- **Embedded Error Checking**. By default MGI does not check for errors in MGI tags that are embedded in HTML. To have MGI check and report errors in embedded tags, check the box beside the "Report Errors in Embedded MGI Tags". **Warning**: MGI may find embedding errors in scripts such as Javascript which are not actually errors and are acceptable coding. To remedy this problem, add spaces to the script near embedded tags (after open brackets "<") or turn this option off.
- **Default Index Pages**. In WebSTAR (Mac) only, you have the option of using the default index pages defined in the web server admin or using the default index pages defined for the region of the specified web site. By default, the region is set to use the "Web Server-Defined Default Pages". To use the default index pages listed for the region, select the radio button beside "Use". Enter each default index page on one line with a return. For example, you may want the "index.html", "index.htm", and "index.mgi" pages to be served by default when a specific index page is not entered. Default pages are checked and served in the order entered.
- **Custom Error Pages.** By default, MGI error pages, (e.g., the "Wow, I bet you didn't expect to see this..." page) display when errors occur. To a custom error page instead of the default MGI error page, select the radio button beside "Use" and enter the name of the custom error page (from the root of the region). A custom error page can display specific aspects of an error using the mgiGetErrorParameter tag.
- **File Upload Preferences**. In the Region Preferences, you may configure the file upload settings including the maximum file size (in kilobytes), the sub-folder (inside the selected region) where files are saved, the regular expressions that a file name must match to be uploaded, and the IP addresses that are allowed or denied access to upload files. See the MGI Server Admin Features or MGI User Guide for more information about configuring the File Upload preferences.
- **Save the Region Preferences**. Click the "Save" button to save the Region Preferences. Preference changes take effect immediately.
- **Module Access for the Region**. At the bottom of the Region Preferences window the modules available for the selected region are listed. A module disabled in the Domain Management window will not be listed in the Region Preferences window. By default the region has access to all modules. To disable the region's access to a module, click the "Disable" button beside the module. When disabled the module name displays in plain text with a line through the name. To enabled a module, click the "Enable" button beside the module. When enabled the module name displays in bold text.
- **Module Preferences**. To configure the module preferences, click the "Edit..." button beside the module name. Under the Module Preferences window, the

MGI tags in each module are listed. By default the region has access to all tags in each module. To disable access to a specific MGI tag within a module, uncheck the box beside the tag name and click the "Save" button. To enabled an MGI tag within a module, check the box beside the tag name and click the "Save" button or click the "Revert" button to return all settings to default. Additional settings for specific MGI functions such as database search result limits, date and time GMT offsets, and mail server settings may also be entered in the module preferences. If you change the module preferences, click the "Save" button to save the settings. Preference changes take effect immediately.

For Non-DNS Accounts:

a. Add one Domain for all Non-DNS folders. Under " Create new domain" enter a label for the non-dns accounts (e.g., "non-dns") and click the "Create..." button. This name is simply a label for these accounts so that you can recognize them in the Domain Management list.

b. Configure Domain Preferences. Select the non-dns account in the Domain Management list and click the "Edit" button.

- **Virtual Hosts**. In the text box under Domain Preferences, enter all virtual hosts that resolve to the root level of the server. For example, your server's IP address or name (e.g., server.domain.com) may resolve to the root level of the server. Enter the domain or IP address that clients use to access their non-dns web site. Enter each host on one line with a return. Click the "Save" button. Preference changes take effect immediately.
- **Module Access.** At the bottom of the Domain Preferences window the modules available for all non-dns web sites are listed. The module list does not reflect modules that are disabled at the server level (under Module Management in the MGI Server Admin). By default the web sites have access to all modules. To disable access to a module, click the "Disable" button beside the module. When disabled the module name displays in plain text with a line through the name. To enabled a module, click the "Enable" button beside the module. When enabled the module name displays in bold text. You may enable or disable module access for specific non-dns accounts under region preferences.

c. Add a Region for Each Non-DNS Account. Click the "Regions" link beside "Domain Preferences" to view the "Region Management" list. Add a region for each non-dns web site. To add a region, enter the web site's folder name under "Create new region" and click the "Create..." button.

NOTE: If you are using a virtual host application that modifies the URL rather than maintaining separate virtual roots (such as **Welcome in normal mode or HomeDoor** on a Macintosh), you must prepend the region name with the host folder path. For example, the Welcome admin requires a "host folder" for each virtual host. If the host folder for a domain is "/websites/" then the non-dns account region name should be entered as "/websites/accountFolderName/". If you are running a "true" virtual hosting system (such as WebSTAR Virtual Host or Welcome in router mode on a Macintosh or an NT IIS virtual hosting system), there is no need to prepend information to the region name since the URL is not modified.

Default Region. If you are running a shared hosting environment, we recommend that you disable all modules in the Default Region of your Non-DNS domain. This will disallow unauthorized use of MGI on your server and prevent unintended shared databases and other access.

d. Configure Region Preferences. Settings for the default region will apply to any non-dns folder that is not configured as a separate region. To configure a region's preferences, select the region in the list and click the "Edit" button.

- **Remove Unprocessed Tags**. By default MGI tags that are not recognized or that are not processed are removed from pages before they are served. To leave unrecognized or unprocessed tags in pages, check the box beside "Do Not Remove Unprocessed MGI Tags".
- **Embedded Error Checking**. By default MGI does not check for errors in MGI tags that are embedded in HTML. To have MGI check and report errors in embedded tags, check the box beside the "Report Errors in Embedded MGI Tags". **Warning**: MGI may find embedding errors in scripts such as Javascript which are not actually errors and are acceptable coding. To remedy this problem, add spaces to the script near embedded tags (after open brackets "<") or turn this option off.
- **Default Index Pages**. In WebSTAR (Mac) only, you have the option of using the default index pages defined in the web server admin or using the default index pages defined for the region of the specified web site. By default, the region is set to use the "Web Server-Defined Default Pages". To use the default index pages listed for the region, select the radio button beside "Use". Enter each default index page on one line with a return. For example, you may want the "index.html", "index.htm", and "index.mgi" pages to be served by default when a specific index page is not entered. Default pages are checked and served in the order entered.
- **Custom Error Pages.** By default, MGI error pages, (e.g., the "Wow, I bet you didn't expect to see this..." page) display when errors occur. To a custom error page instead of the default MGI error page, select the radio button beside "Use" and enter the name of the custom error page (from the root of the region). A custom error page can display specific aspects of an error using the

mgiGetErrorParameter tag.

- **File Upload Preferences**. In the Region Preferences, you may configure the file upload settings including the maximum file size (in kilobytes), the sub-folder (inside the selected region) where files are saved, the regular expressions that a file name must match to be uploaded, and the IP addresses that are allowed or denied access to upload files. See the MGI Server Admin Features or MGI User Guide for more information about configuring the File Upload preferences.
- **Save the Region Preferences**. Click the "Save" button to save the Region Preferences. Preference changes take effect immediately.
- **Module Access for the Region**. At the bottom of the Region Preferences window the modules available for the selected region are listed. A module disabled in the Domain Management window will not be listed in the Region Preferences window. By default the region has access to all modules. To disable the region's access to a module, click the "Disable" button beside the module. When disabled the module name displays in plain text with a line through the name. To enabled a module, click the "Enable" button beside the module. When enabled the module name displays in bold text.
- **Module Preferences**. To configure the module preferences, click the "Edit..." button beside the module name. Under the Module Preferences window, the MGI tags in each module are listed. By default the region has access to all tags in each module. To disable access to a specific MGI tag within a module, uncheck the box beside the tag name and click the "Save" button. To enabled an MGI tag within a module, check the box beside the tag name and click the "Save" button or click the "Revert" button to return all settings to default. Additional settings for specific MGI functions such as database search result limits, date and time GMT offsets, and mail server settings may also be entered in the module preferences. If you change the module preferences, click the "Save" button to save the settings. Preference changes take effect immediately.

[System Requirements] [Installation] [Domain Configuration] [Server Admin] [Domain Admin] [Licensing]

[MGI Guides Main Menu] [Admin Guide Main Menu]

MGI Server Admin

MGI's web-based server administration allows you to specify server settings, manage MGI modules, manage domains (and regions of domains), and update the server admin username and password.

To access the server admin page, install MGI and access the "MGIServerAdmin.xxx" page in a web browser from **any** virtually hosted domain on the server. Replace the ".xxx" suffix with any suffix that is mapped to be processed by MGI2. For example, if you have the ".mgi" extension mapped to be processed by MGI2 you would access "MGIServerAdmin.mgi" or if you have the ".html" extension mapped to be processed by MGI2, you would access "MGIServerAdmin.html".

The server admin page is password-protected. During the installation of MGI, you are prompted to enter a username and password. Use your specified username and password to access the server admin page. If you did not specify a password during installation, the default password is case-sensitive "MGI".

The server admin includes four sections: Server Settings, Module Management, Domain Management, and Security. To view screen shots and instructions for using these sections of the server admin, click any link below.

Server Settings

Specify MGI event logging and log rotation schedules and update your MGI serial number and authorization.

Module Management

Load or unload MGI modules, enable or disable MGI modules for use, and download module updates through version tracker.

Domain Management

Manage domains including virtual host configurations, enable or disable MGI modules domain by domain, manage regions of domains including custom error pages, file uploading, module preferences, and security.

Security

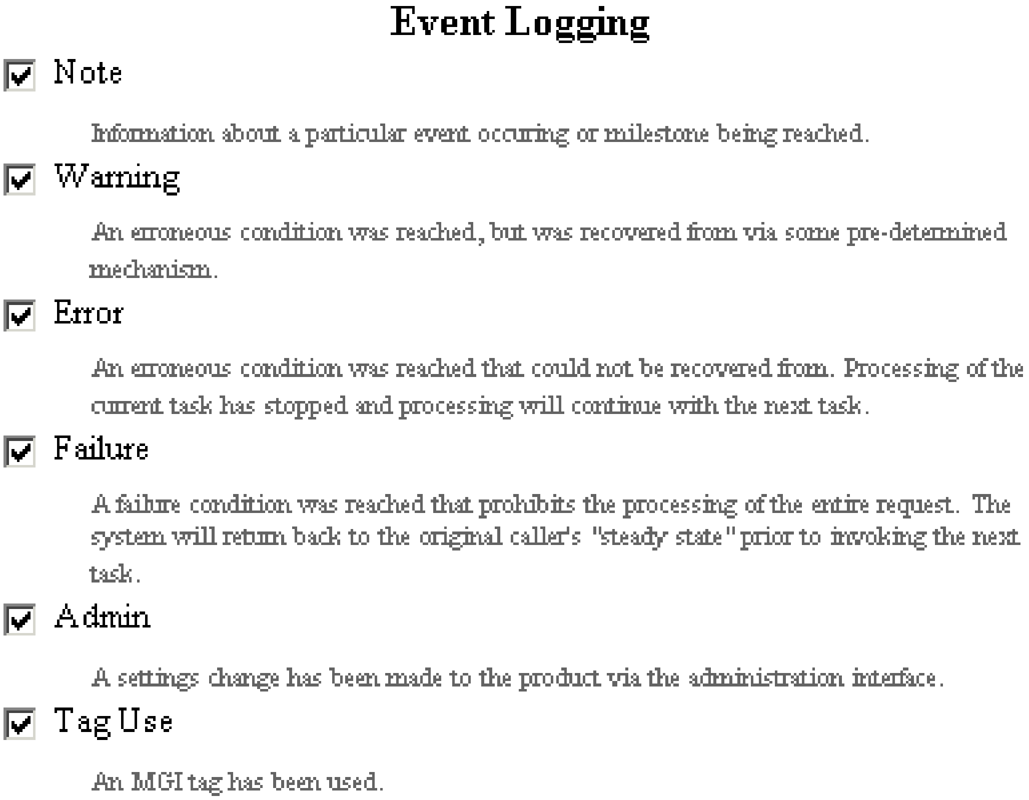
Change the server admin username and password.

MGI Server Admin - Server Settings

MGI server settings allow you to specify MGI event logging and log rotation schedules and update your MGI serial number and authorization.

Event Logging

MGI records six types of events in the MGI log. See descriptions of each event type below. To enable logging for an event, check the box beside the event name. To disable logging for an event, uncheck the box beside the event name. All events are logged in the same file except the Tag Use events which are logged in a separate file. MGI Logs are stored in the MGI Files folder in a folder named "Logs".



Event Logging

☑ Note

Information about a particular event occuring or milestone being reached.

☑ Warning

An erroneous condition was reached, but was recovered from via some pre-determined mechanism.

☑ Error

An erroneous condition was reached that could not be recovered from. Processing of the current task has stopped and processing will continue with the next task.

☑ Failure

A failure condition was reached that prohibits the processing of the entire request. The system will return back to the original caller's "steady state" prior to invoking the next task.

☑ Admin

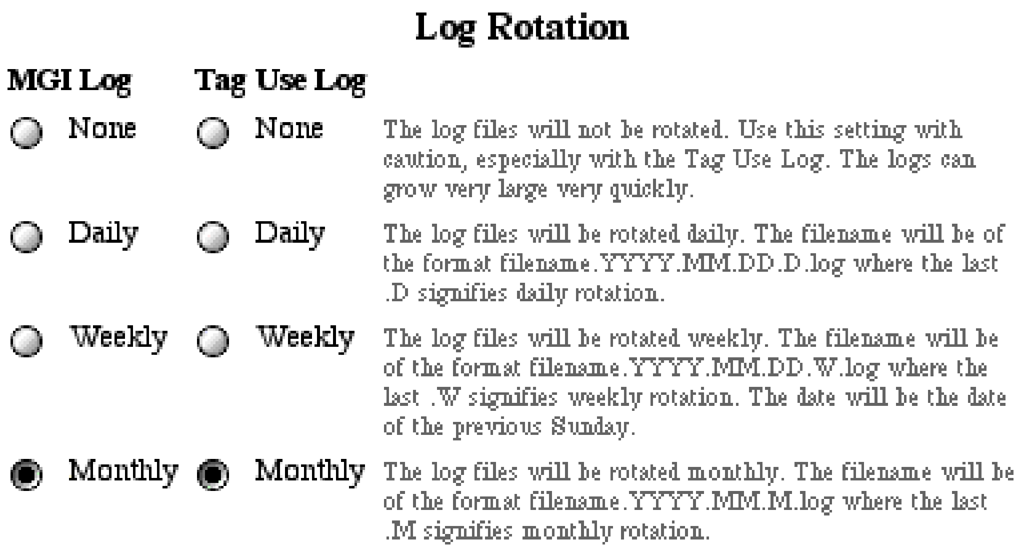
A settings change has been made to the product via the administration interface.

☑ Tag Use

An MGI tag has been used.

Log Rotation

MGI can rotate the MGI Log and Tag Use Log on a daily, weekly, or monthly schedule. To have MGI rotate the MGI Log or Tag Use Log, select the radio button beside the schedule you wish to use for each log. MGI log rotation **does not** affect the web server's log.

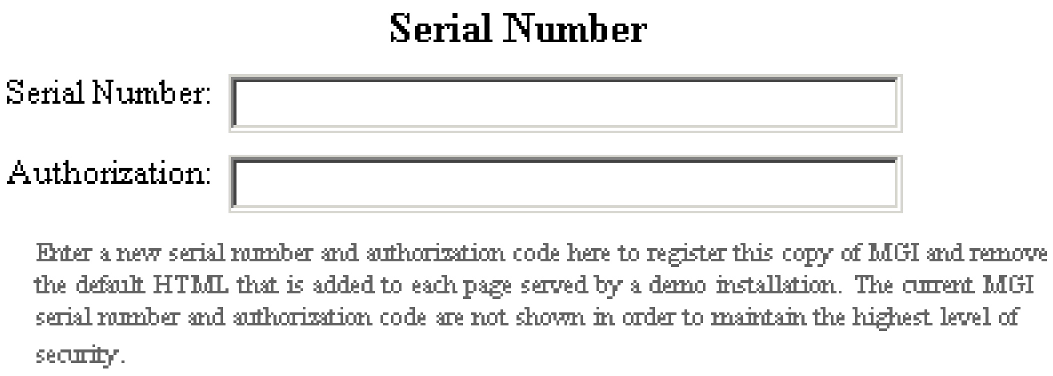


Log Rotation

MGI Log	Tag Use Log	
○ None	○ None	The log files will not be rotated. Use this setting with caution, especially with the Tag Use Log. The logs can grow very large very quickly.
○ Daily	○ Daily	The log files will be rotated daily. The filename will be of the format filename.YYYY.MM.DD.D.log where the last .D signifies daily rotation.
○ Weekly	○ Weekly	The log files will be rotated weekly. The filename will be of the format filename.YYYY.MM.DD.W.log where the last .W signifies weekly rotation. The date will be the date of the previous Sunday.
● Monthly	● Monthly	The log files will be rotated monthly. The filename will be of the format filename.YYYY.MM.M.log where the last .M signifies monthly rotation.

Serial Number

When MGI runs in demo mode, a header and footer are added to each page that is served. To remove the demo header and footer, purchase a license for MGI at PagePlanet Software. When you receive your serial number and authorization code, enter them in the specified fields. You do not need to re-install MGI.



Serial Number

Serial Number:

Authorization:

Enter a new serial number and authorization code here to register this copy of MGI and remove the default HTML that is added to each page served by a demo installation. The current MGI serial number and authorization code are not shown in order to maintain the highest level of security.

[Server Settings] [Module Management] [Domain Management] [Security]

[System Requirements] [Installation] [Domain Configuration] [Server Admin] [Domain Admin] [Licensing]

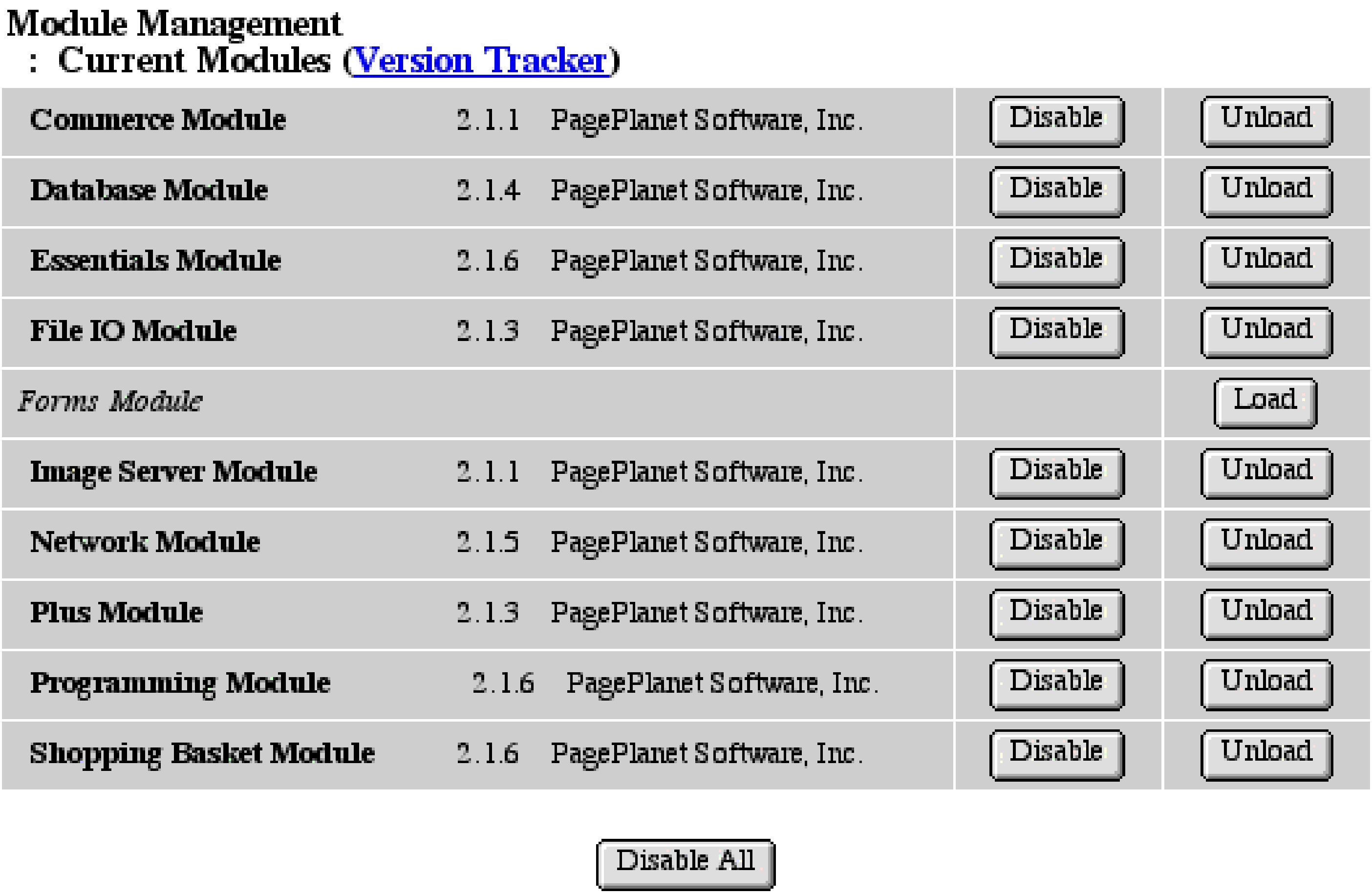
[MGI Guides Main Menu] [Admin Guide Main Menu]

MGI Server Admin - Module Management

MGI module management allows you to load or unload MGI modules, enable or disable MGI modules for use, and download module updates through version tracker.

Current Modules

In the module management section of the server admin, each available MGI module (with the version number and module author) is listed. Modules are sets of MGI tags. To make the module active on the server, click the "Load" button. To make a module inactive on the server, click the "Unload" button. New modules downloaded from Version Tracker must be loaded. If you are replacing an old module with a new module, unload the old module, replace the old module file with the new module file, then load the new module. To make a module's tags available for use on the server, click the "Enable" button. When enabled, the module name is bold text. To prevent a module's tags from being used on the server, click the "Disable" button. When disabled, the module name is plain text. To prevent all MGI tags from being used on the server, click the "Disable All" button.

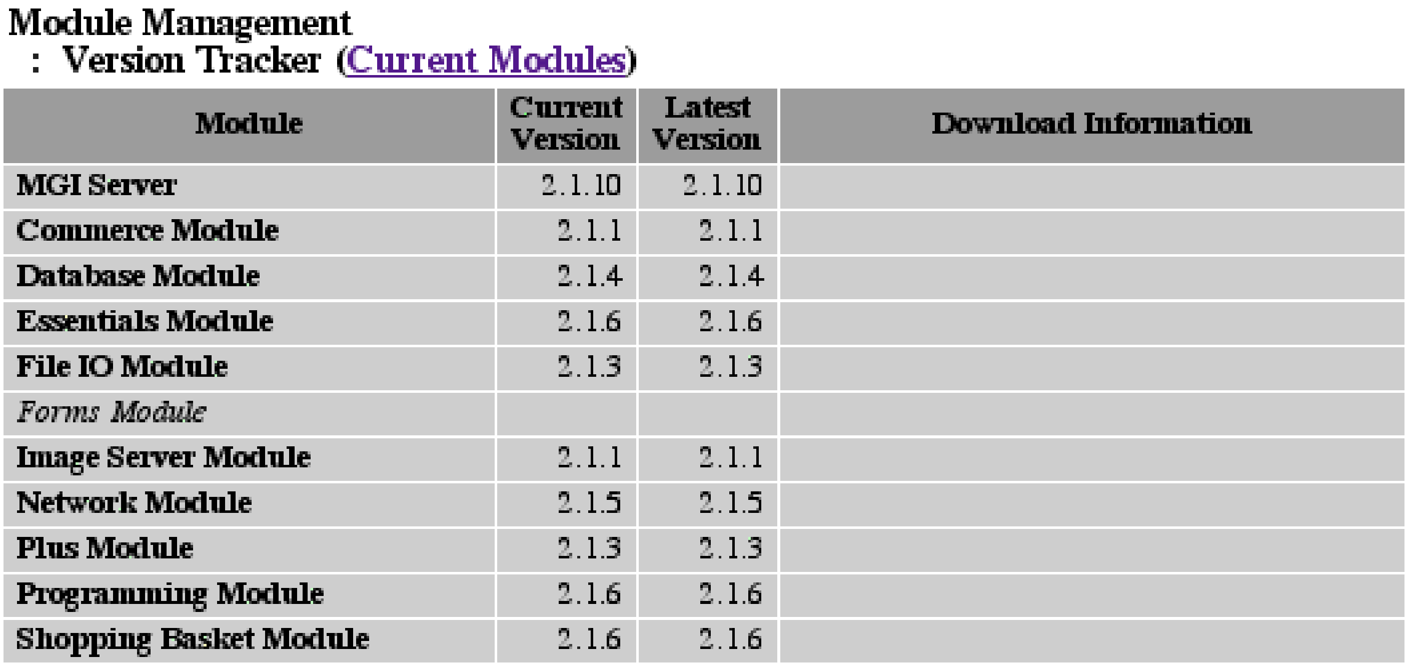


Module Management
: Current Modules (Version Tracker)

Commerce Module	2.1.1	PagePlanet Software, Inc.	Disable	Unload
Database Module	2.1.4	PagePlanet Software, Inc.	Disable	Unload
Essentials Module	2.1.6	PagePlanet Software, Inc.	Disable	Unload
File IO Module	2.1.3	PagePlanet Software, Inc.	Disable	Unload
Forms Module				Load
Image Server Module	2.1.1	PagePlanet Software, Inc.	Disable	Unload
Network Module	2.1.5	PagePlanet Software, Inc.	Disable	Unload
Plus Module	2.1.3	PagePlanet Software, Inc.	Disable	Unload
Programming Module	2.1.6	PagePlanet Software, Inc.	Disable	Unload
Shopping Basket Module	2.1.6	PagePlanet Software, Inc.	Disable	Unload

Version Tracker

Use the version tracker to view and download individual module updates. On the version tracker screen, the name of each module on your server is listed with the current version, the latest version, and a download URL (if needed). MGI modules are enhanced often, but you do not have to reinstall the entire program to install a new module. To install a new module, download the new module, unload the old module, replace the old module file with the new module file, then load the new module.



Module Management
: Version Tracker (Current Modules)

Module	Current Version	Latest Version	Download Information
MGI Server	2.1.10	2.1.10	
Commerce Module	2.1.1	2.1.1	
Database Module	2.1.4	2.1.4	
Essentials Module	2.1.6	2.1.6	
File IO Module	2.1.3	2.1.3	
Forms Module			
Image Server Module	2.1.1	2.1.1	
Network Module	2.1.5	2.1.5	
Plus Module	2.1.3	2.1.3	
Programming Module	2.1.6	2.1.6	
Shopping Basket Module	2.1.6	2.1.6	

[Server Settings] [Module Management] [Domain Management] [Security]

[System Requirements] [Installation] [Domain Configuration] [Server Admin] [Domain Admin] [Licensing]

[MGI Guides Main Menu] [Admin Guide Main Menu]

MGI Server Admin - Domain Management

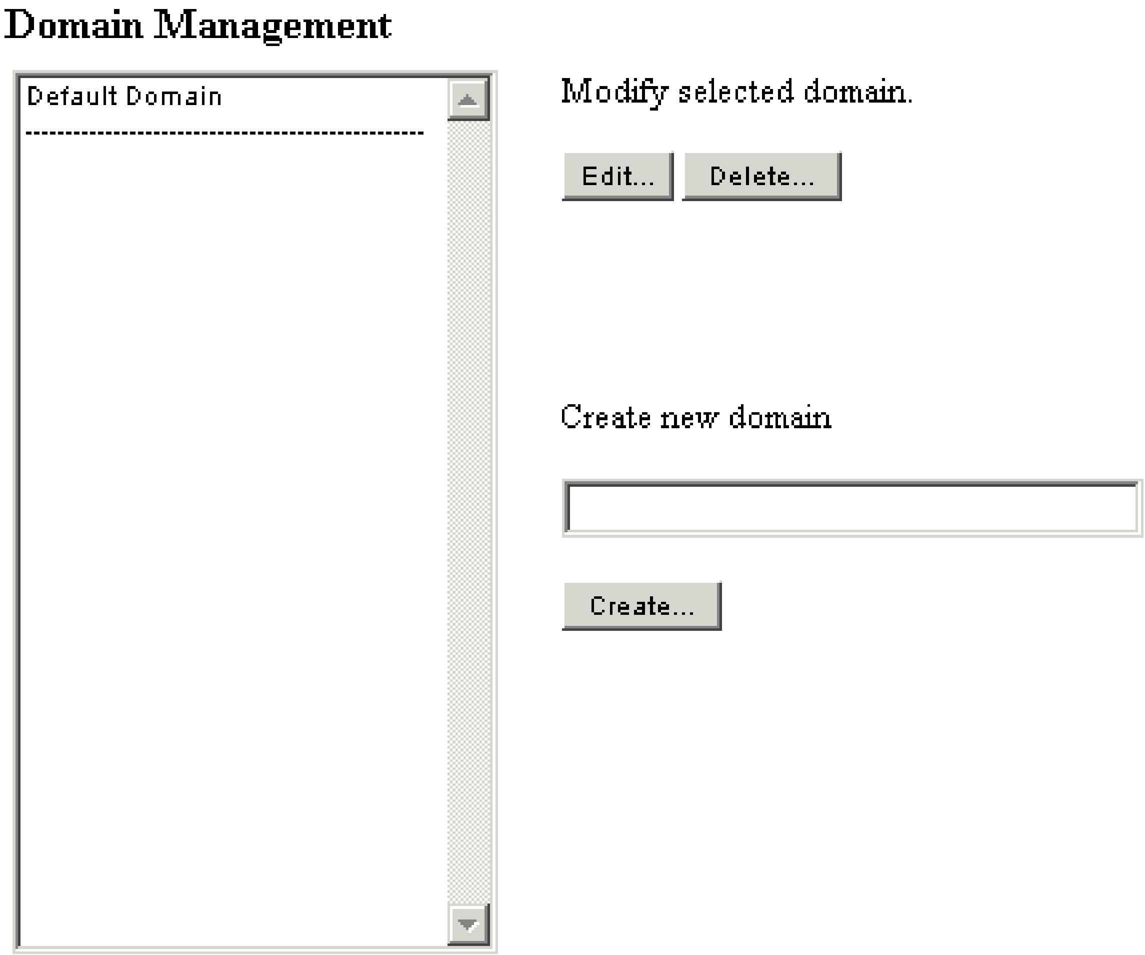
MGI domain management allows you to manage domains including virtual host configurations, enable or disable MGI modules on a doman basis, manage regions of domains including custom error pages, file uploading, module preferences, and security.

Domain Management

In MGI, a domain is defined as one web site. One web site folder (or alias) at the root of WebSTAR or one web site created in IIS.

In the domain management window, a Default Domain appears at the top of the list. The default domain is activated for any web site that is not specifically configured in Domain Management. All web sites that fall under the default domain have shared access to databases and other settings. If you are running a shared hosting environment, we recommend that you disable all modules in the Default Domain (see the Domain Preferences below). This will disallow unauthorized use of MGI on your server and prevent unintended shared databases and other access.

Add a new domain for each web site on your server. To add a new domain, enter a label for the web site (e.g., domain.com) under "Create new domain" and click the "Create..." button. The domain will appear in the domain list.



Domain Management

Default Domain

Modify selected domain.

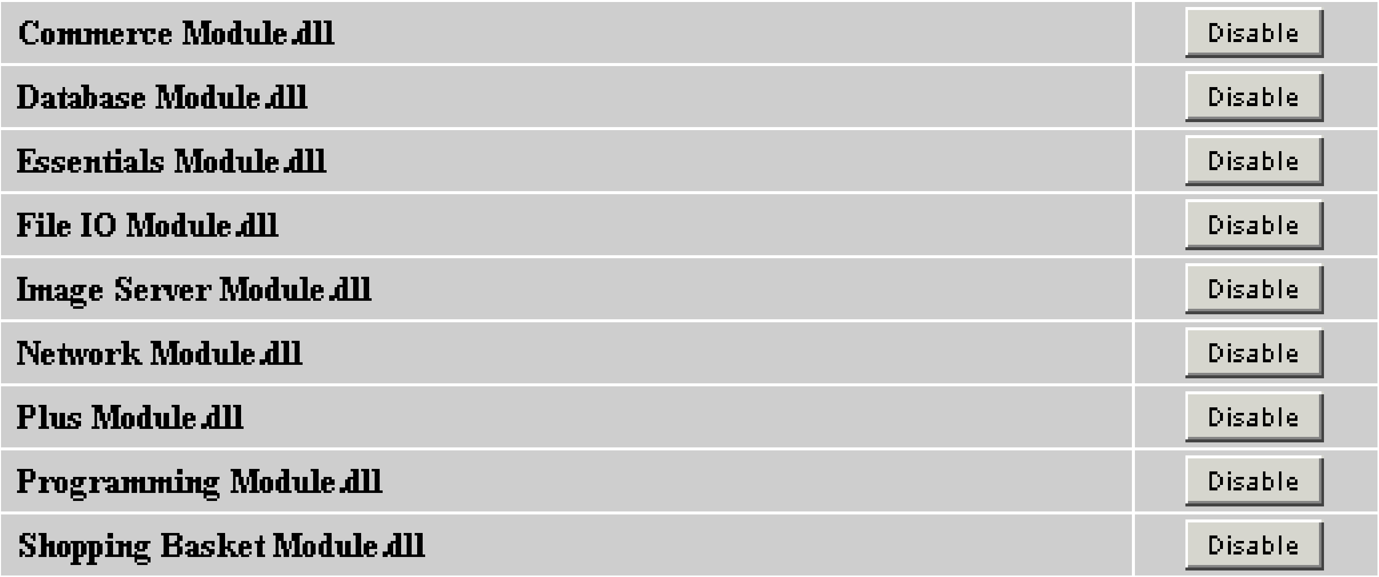
Create new domain

Domain Preferences

Domain preferences allow you to enter virtual host names for the specified domain and enable or disable MGI modules for the specified domain.

In the virtual host name box, enter the name of all virtual domains that point to the web site (one per line) and click the "Save" button. For example, for many domains the virtual hosts "domain.com" and "www.domain.com" should be entered. **Note: Virtual host field mapping must also be set up in the settings of the web server and cannot be specified in MGI alone.** Click the "Return" button to return to the Domain Management window or continue configuring the domain preferences.

At the bottom of the Domain Preferences window, the modules available for the web site are listed. The module list does not reflect modules that are disabled at the server level (under Module Management in the MGI Server Admin). By default the web site has access to all modules. To disable the web site's access to a module, click the "Disable" button beside the module. When disabled the module name displays in plain text with a line through the name. To enabled a module, click the "Enable" button beside the module. When enabled the module name displays in bold text. Changes to modules in the Domain Preferences affects only the specified domain and its regions, but does not affect the server module settings or module settings for other domains.



Commerce Module.dll	Disable
Database Module.dll	Disable
Essentials Module.dll	Disable
File IO Module.dll	Disable
Image Server Module.dll	Disable
Network Module.dll	Disable
Plus Module.dll	Disable
Programming Module.dll	Disable
Shopping Basket Module.dll	Disable

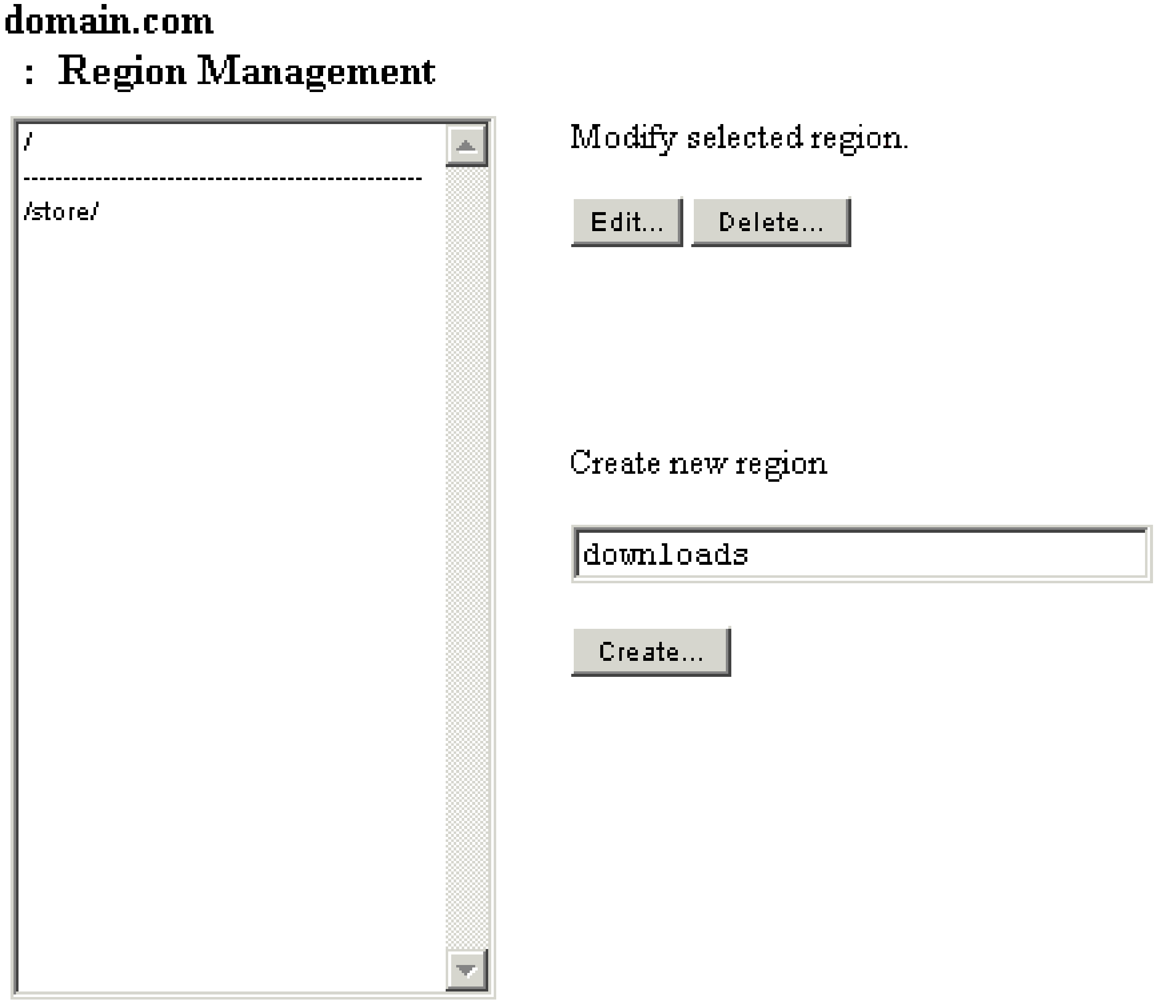
Region Management

Region Management in the Server Admin interface is the same as the region management in the Domain Admin interface. Domain Admin access might be given to those individuals responsible for a domain who do not have server privileges.

In MGI, a region is defined as a subsection of one web site (i.e., a subfolder of the web site). Regions specified in the Region Management list restrict file paths within the region to the root of the region (i.e., a file path cannot walk "up and out" of a defined region) and restrict database access and other settings to the region. Regions can be created to separate access and settings between sub-folders of a web site. For example, if you need to separate database access for different divisions of a company that are sub-folders of a web site, you could create a region for each division. For the majority of web sites, shared access within a domain is preferred and allowed, therefore the default region (/ - the root level) is used.

To add a region, enter the region (sub-folder) path under "Create new region" and click the "Create..." button. The region path is based on your virtual host mapping. A region is any sub-folder under the mapping level of the domain for the region. For example, if your virtual host "domain.com" maps to the folder "/domain/" and you want to create a region for "/domain/store/", then you would enter "store" as the region path. However, if you have the domain "store.domain.com" mapped to the folder "/domain/store/", then the "store" folder is actually the root level ("/") of that domain.

NOTE: If you are using a virtual host application that modifies the URL rather than maintaining separate virtual roots (such as Welcome in normal mode or HomeDoor on a Macintosh), you must prepend the region name with the host folder path. For example, the Welcome admin requires a "host folder" for each virtual host. If the host folder for a domain is "/websites/company/" then the "store" region name should be entered as "/websites/company/store/". If you are running a "true" virtual hosting system (such as WebSTAR Virtual Host or Welcome in router mode on a Macintosh or an NT IIS virtual hosting system), there is no need to prepend information to the region name since the URL is not modified. For example, the "store" region should simply be entered as "/store/".



domain.com
: Region Management

/

/store/

Modify selected region.

Create new region

downloads

Region Preferences

Region preferences in the Server Admin interface are the same as the region preferences in the Domain Admin interface. Domain Admin access might be given to those individuals responsible for a domain who do not have server privileges.

Region preferences allow you to specify the language, the handling of errors, file uploads, the availability of tags in each MGI module, and the global region settings for specific MGI tags.

Unprocessed MGI Tags

If an MGI tag is mis-named or not available to the specified domain, MGI can pass the tag through or remove the tag from the page before it is served. To allow the tag to be passed through, click the checkbox beside "Do Not Remove Unprocessed MGI Tags" and click the "Save" button. To have the tag removed, unclick the checkbox beside "Do Not Remove Unprocessed MGI Tags" and click the "Save" button.

Embedded Error Checking

By default MGI does not check for errors in MGI tags that are embedded in HTML. To have MGI check and report errors in embedded tags, check the box beside the "Report Errors in Embedded MGI Tags". **Warning**: MGI may find embedding errors in scripts such as Javascript which are not actually errors and are acceptable coding. To remedy this problem, add spaces to the script near embedded tags (after open brackets "<") or turn this option off.

Error Pages

When MGI encounters an error, an error page is displayed. The default MGI error page is named **"Wow! I bet you didn't expect to see this..."** followed by the specific error and suggestions for remedy of the error. If there is a mistake in MGI code, the error page will highlight the code being processed in red to facilitate de-bugging. Custom error pages have the option of displaying any or all parts of an error using the mgiGetErrorParameter tag. To use the default MGI error page for the region, click the radio button beside "Use Default MGI Error Page" and click the "Save" button. To use a custom error page in the region, click the radio button beside "Use...", enter the path to the custom error page (relative to the root of the region), and click the "Save" button.

Default Index Pages

In WebSTAR (Mac) only, you have the option of using the default index pages defined in the web server admin or using the default index pages defined for the region of the specified web site. By default, the region is set to use the "Web Server-Defined Default Pages". To use the default index pages listed for the region, select the radio button beside "Use". Enter each default index page on one line with a return. For example, you may want the "index.html", "index.htm", and "index.mgi" pages to be served by default when a specific index page is not entered. Default pages are checked and served in the order entered.

HTTP File Uploads

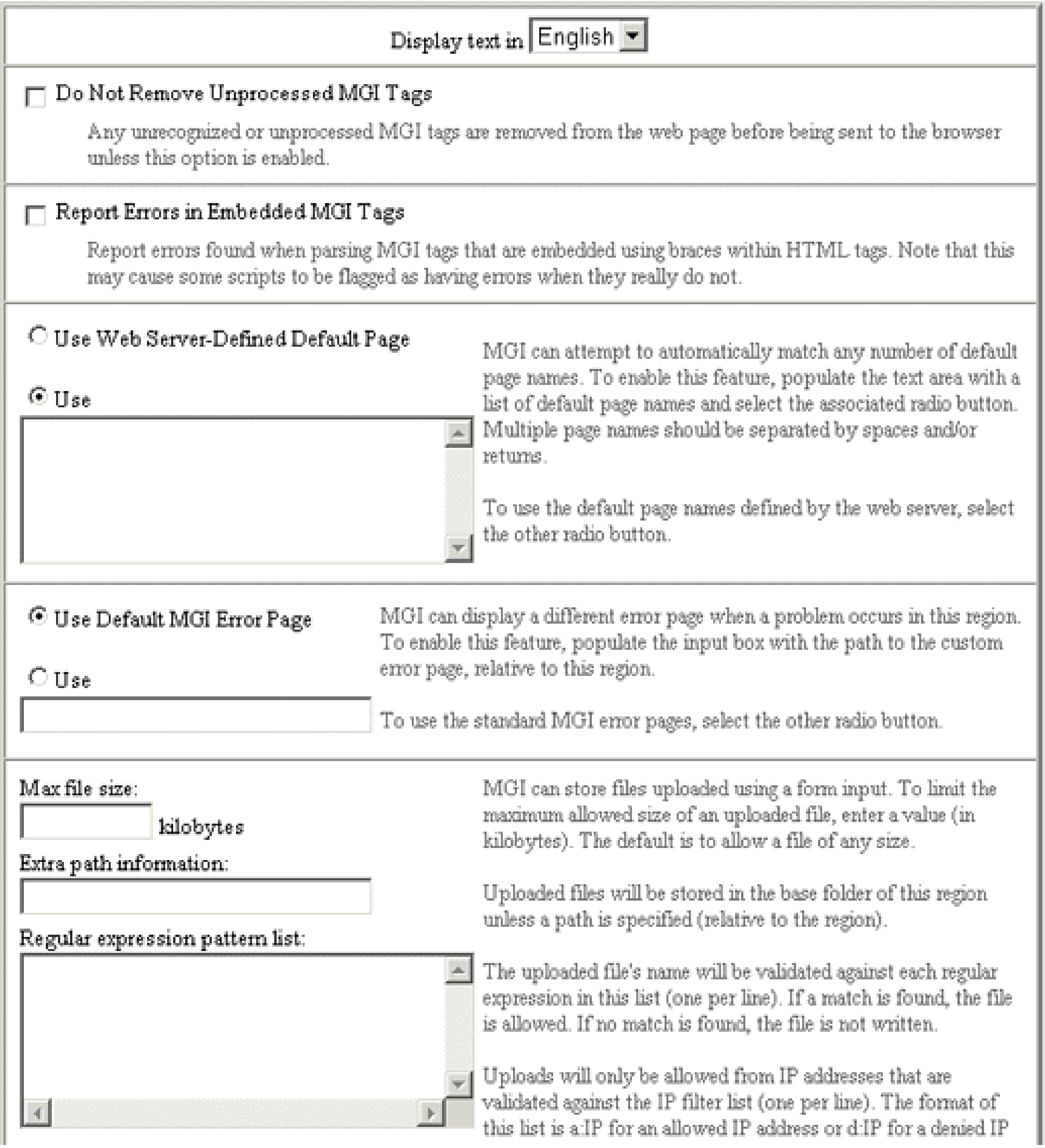
MGI supports file uploads from HTML file input elements. In the Region Preferences, you may set a maximum file size for uploaded files, a specific folder for uploaded files, a validation for file names and allow or deny IP addresses (including wildcards).

To set a maximum file size, enter the maximum file size in kilobytes under "Max file size:" and click the "Save" button.

To specify a folder for uploaded files, enter the path to the folder (relative to the root of the region) under "Extra path information:" and click the "Save" button.

To validate the file name before files are uploaded, enter one or more regular expression patterns under "Regular expression pattern list" and click the "Save" button. See the regular expression symbol list for more information.

For security resons you are required to list at least one IP address for file uploads, although you can use wildcard values. To allow an IP address, enter an "A" followed by a colon and the IP address (e.g., "a:205.160.14.88"). To deny an IP address, enter a "D" followed by a colon and the IP address (e.g., "d:205.160.14.99"). You may use an asterisk (*) for a wildcard value in any position of the IP address. For example, to allow all IP numbers from the 272 Class C block, you would enter "a:272.*.*.*" as a filter. For multiple filters, enter one filter per line. When a file upload request is received, the most restrictive match for the IP address will be processed and if two filters match, a deny filter will override an accept filter. For example, if you wish to allow all IPs from the 205.160.14 address except the IP address "205.160.14.233", then you would enter an allow and deny statement in the IP filter list ("a:205.160.14.*" and "d:205.160.14.233"). To allow all IP addresses, use wildcard values for all positions (i.e., "a:*.*.*.*").



Display text in English

□ Do Not Remove Unprocessed MGI Tags

Any unrecognized or unprocessed MGI tags are removed from the web page before being sent to the browser unless this option is enabled.

□ Report Errors in Embedded MGI Tags

Report errors found when parsing MGI tags that are embedded using braces within HTML tags. Note that this may cause some scripts to be flagged as having errors when they really do not.

○ Use Web Server-Defined Default Page

● Use

MGI can attempt to automatically match any number of default page names. To enable this feature, populate the text area with a list of default page names and select the associated radio button. Multiple page names should be separated by spaces and/or returns.

To use the default page names defined by the web server, select the other radio button.

● Use Default MGI Error Page

○ Use

MGI can display a different error page when a problem occurs in this region. To enable this feature, populate the input box with the path to the custom error page, relative to this region.

To use the standard MGI error pages, select the other radio button.

Max file size:

kilobytes

Extra path information:

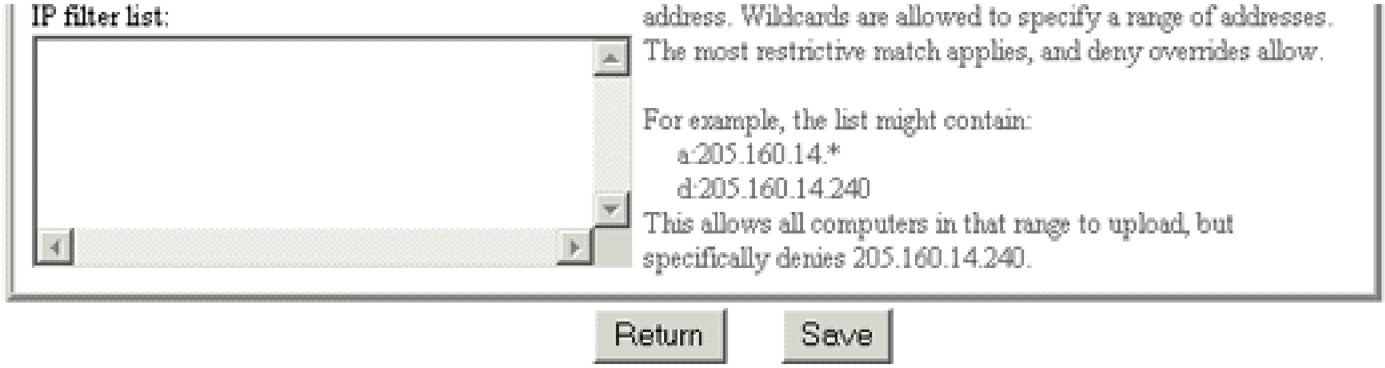
Regular expression pattern list:

MGI can store files uploaded using a form input. To limit the maximum allowed size of an uploaded file, enter a value (in kilobytes). The default is to allow a file of any size.

Uploaded files will be stored in the base folder of this region unless a path is specified (relative to the region).

The uploaded file's name will be validated against each regular expression in this list (one per line). If a match is found, the file is allowed. If no match is found, the file is not written.

Uploads will only be allowed from IP addresses that are validated against the IP filter list (one per line). The format of this list is a:IP for an allowed IP address or d:IP for a denied IP



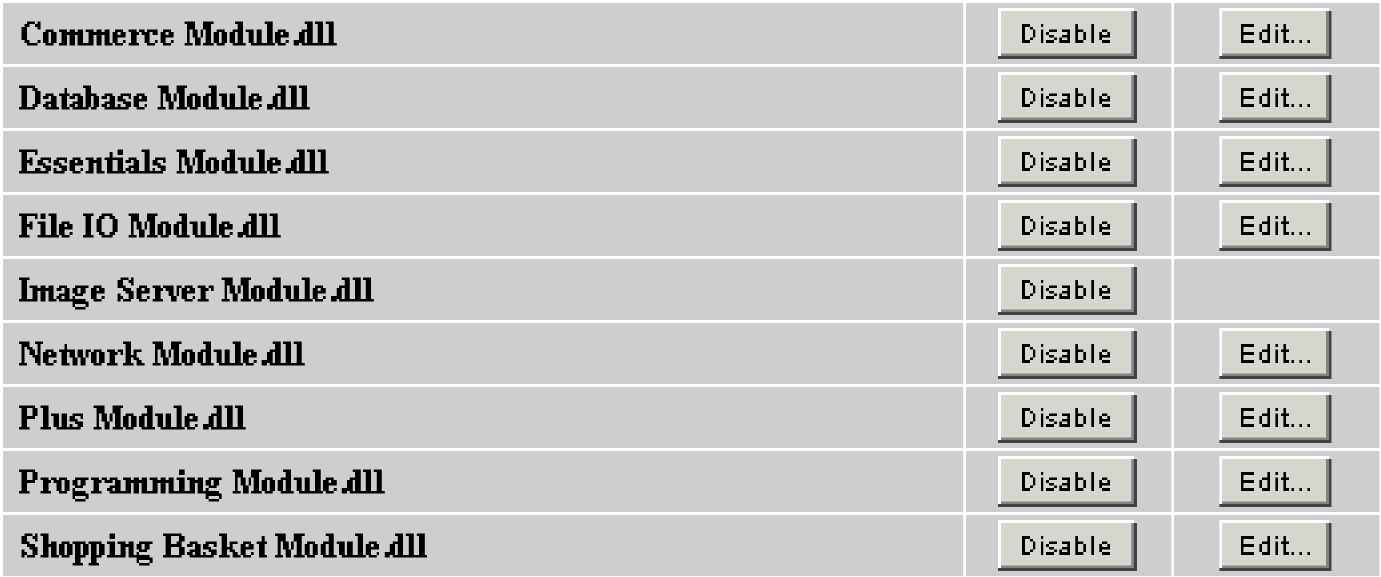
IP filter list:

address. Wildcards are allowed to specify a range of addresses. The most restrictive match applies, and deny overrides allow.

For example, the list might contain:
a:205.160.14.*
d:205.160.14.240
This allows all computers in that range to upload, but specifically denies 205.160.14.240.

Region Module Preferences

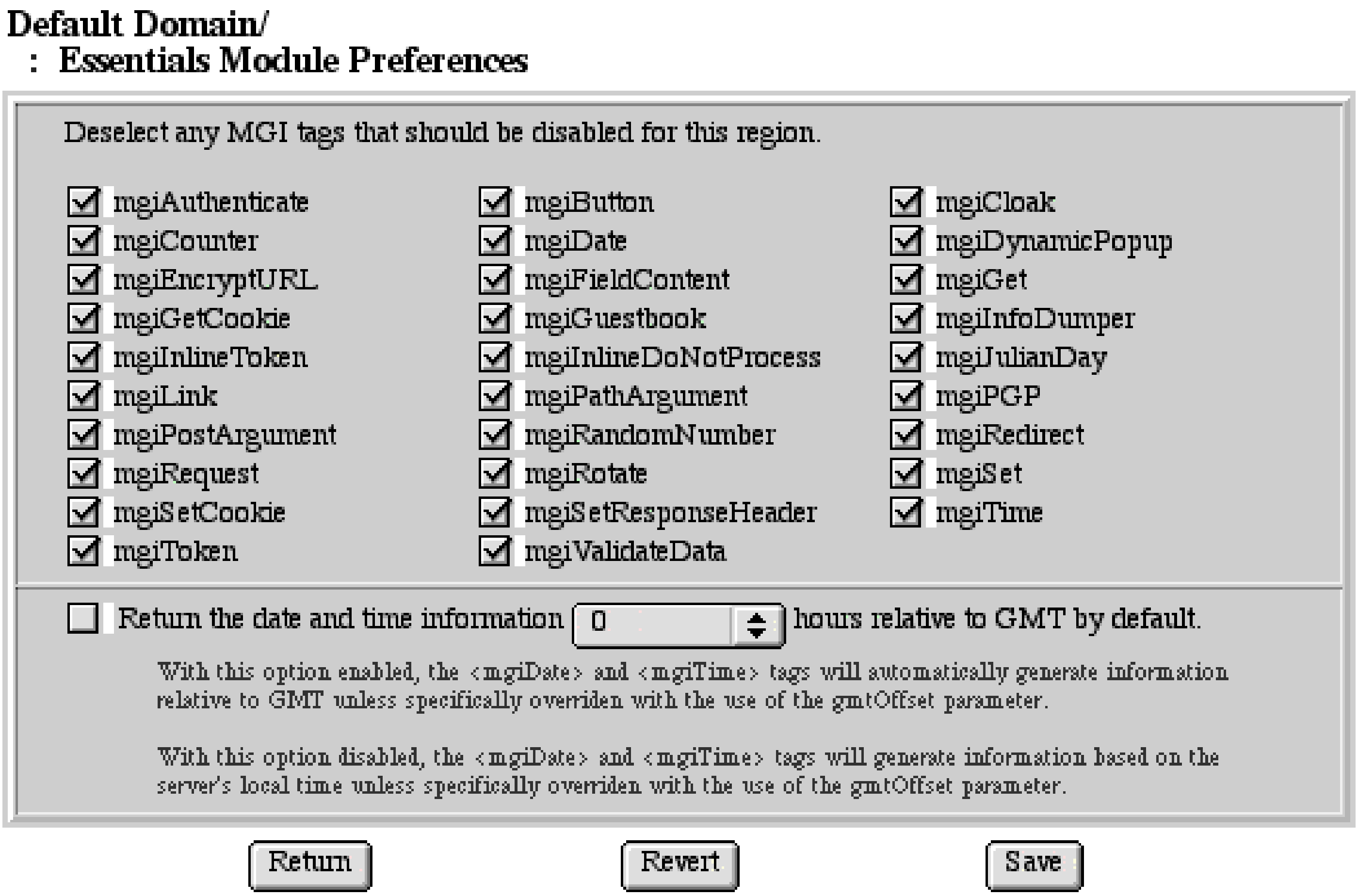
At the bottom of the Region Preferences window the modules available for the selected region are listed. A module disabled in the Domain Management window will not be listed in the Region Preferences window. By default the region has access to all modules. To disable the region's access to a module, click the "Disable" button beside the module. When disabled the module name displays in plain text with a line through the name. To enabled a module, click the "Enable" button beside the module. When enabled the module name displays in bold text. Module management at the region level affects only the specified region and does not affect the server module settings, the domain module settings or module settings for other regions.



Commerce Module.dll	Disable	Edit...
Database Module.dll	Disable	Edit...
Essentials Module.dll	Disable	Edit...
File IO Module.dll	Disable	Edit...
Image Server Module.dll	Disable	
Network Module.dll	Disable	Edit...
Plus Module.dll	Disable	Edit...
Programming Module.dll	Disable	Edit...
Shopping Basket Module.dll	Disable	Edit...

To configure the module preferences, click the "Edit..." button beside the module name. Under the Module Preferences window, the MGI tags in each module are listed. By default the region has access to all tags in each module. To disable access to a specific MGI tag within a module, uncheck the box beside the tag name and click the "Save" button. To enabled an MGI tag within a module, check the box beside the tag name and click the "Save" button or click the "Revert" button to return all settings to default.

Additional settings for specific MGI functions such as database search result limits, date and time GMT offsets, and mail server settings may also be entered in the module preferences. If you change the module preferences, click the "Save" button to save the settings. Preference changes take effect immediately.



Default Domain/
: Essentials Module Preferences

Deselect any MGI tags that should be disabled for this region.

☑ mgiAuthenticate
☑ mgiCounter
☑ mgiEncryptURL
☑ mgiGetCookie
☑ mgiInlineToken
☑ mgiLink
☑ mgiPostArgument
☑ mgiRequest
☑ mgiSetCookie
☑ mgiToken

☑ mgiButton
☑ mgiDate
☑ mgiFieldContent
☑ mgiGuestbook
☑ mgiInlineDoNotProcess
☑ mgiPathArgument
☑ mgiRandomNumber
☑ mgiRotate
☑ mgiSetResponseHeader
☑ mgiValidateData

☑ mgiCloak
☑ mgiDynamicPopup
☑ mgiGet
☑ mgiInfoDumper
☑ mgiJulianDay
☑ mgiPGP
☑ mgiRedirect
☑ mgiSet
☑ mgiTime

□ Return the date and time information 0 hours relative to GMT by default.

With this option enabled, the <mgiDate> and <mgiTime> tags will automatically generate information relative to GMT unless specifically overriden with the use of the gmtOffset parameter.

With this option disabled, the <mgiDate> and <mgiTime> tags will generate information based on the server's local time unless specifically overriden with the use of the gmtOffset parameter.

[Server Settings] [Module Management] [Domain Management] [Security]

[System Requirements] [Installation] [Domain Configuration] [Server Admin] [Domain Admin] [Licensing]

[MGI Guides Main Menu] [Admin Guide Main Menu]

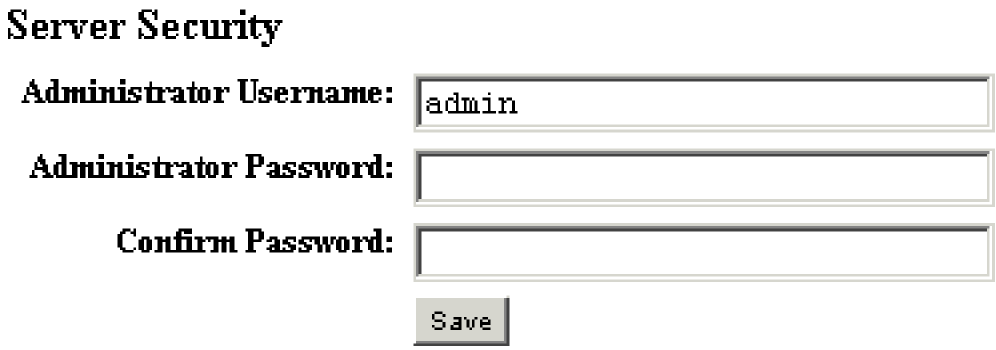
MGI Server Admin - Security

MGI security settings allow you to change the username and password for the server admin.

Server Security

To change the username and password, enter a new username, a new password, and a verification of the password, then click the "Save" button. The username and password are both **case-sensitive**.

All new domains inherit the Server Admin username and password as the Domain Admin username and password. However, changing the Server Admin username and password does not change the Domain Admin username and password of existing domains!



Server Security

Administrator Username: admin

Administrator Password:

Confirm Password:

[Server Settings] [Module Management] [Domain Management] [Security]

[System Requirements] [Installation] [Domain Configuration] [Server Admin] [Domain Admin] [Licensing]

[MGI Guides Main Menu] [Admin Guide Main Menu]

MGI Domain Admin

MGI's web-based domain administration allows you to manage regions within a domain and update the domain admin username and password. Domain Admin access might be given to those individuals responsible for a domain who do not have server privileges.

To access the domain admin page, access the "MGIDomainAdmin.xxx" page in a web browser from one of the domain's virtual host addresses. Replace the ".xxx" suffix with any suffix that is mapped to be processed by MGI2. For example, if you have the ".mgi" extension mapped to be processed by MGI2 you would access "MGIDomainAdmin.mgi" or if you have the ".html" extension mapped to be processed by MGI2, you would access "MGIDomainAdmin.html".

The domain admin page is password-protected. When a domain is created, the Domain Admin username and password are those of the current Server Admin username and password. If the Server Admin username and/or password are changed, those changes do not apply to existing domains.

The domain admin includes two sections: Region Management, and Security. To view screen shots and instructions for using these sections of the domain admin, click any link below.

Region Management

Manage regions and specify the language, the handling of errors, file transfers, the availability of tags in each MGI module, and the global region settings for specific MGI tags.

Security

Change the domain admin username and password.

MGI Domain Admin - Region Management

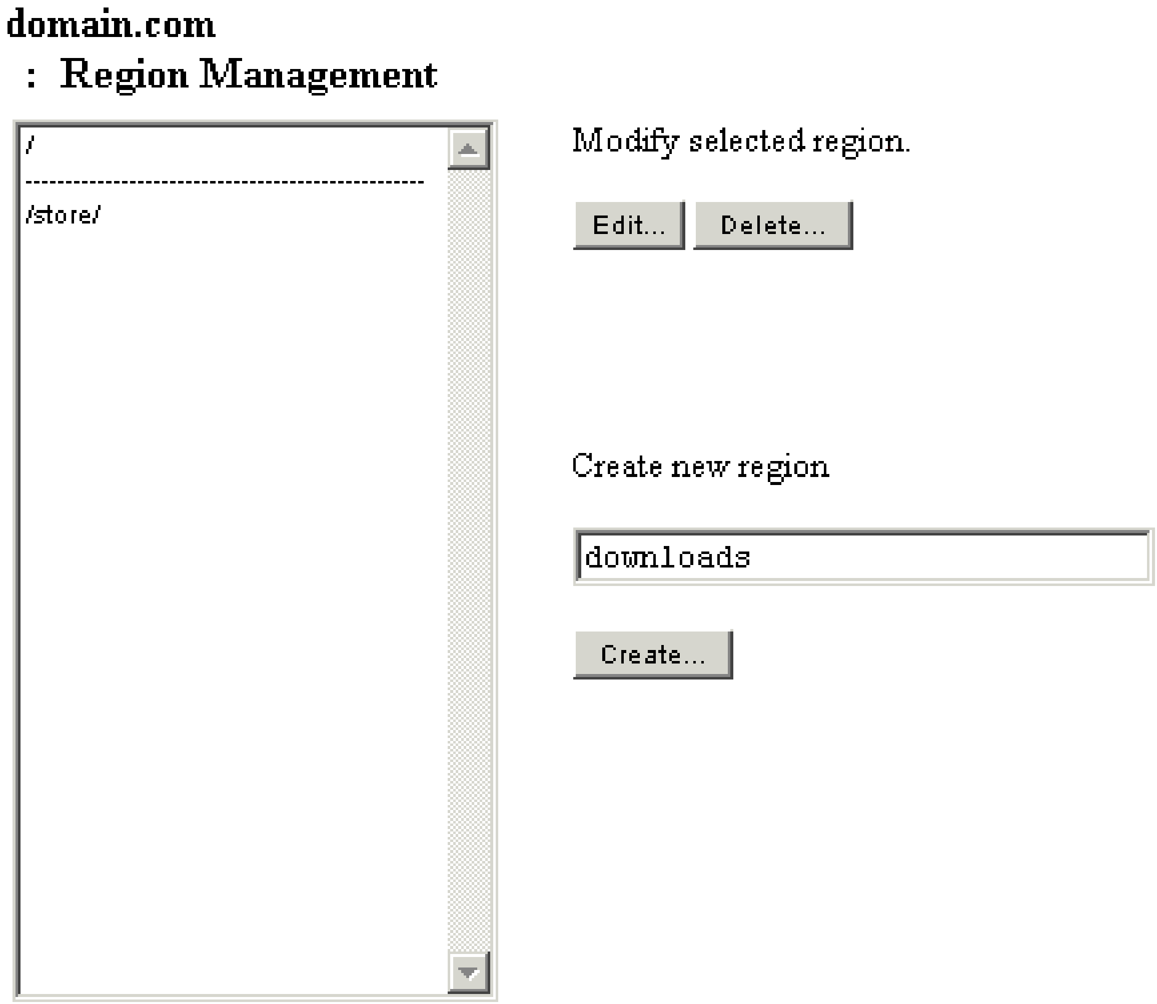
MGI region management allows you to manage regions of a domain and specify the language, the handling of errors, file uploads, the availability of tags in each MGI module, and the global region settings for specific MGI tags.

Region Management

In MGI, a region is defined as a subsection of one web site (i.e., a subfolder of the web site). Regions specified in the Region Management list restrict file paths within the region to the root of the region (i.e., a file path cannot walk "up and out" of a defined region) and restrict database access and other settings to the region. Regions can be created to separate access and settings between sub-folders of a web site. For example, if you need to separate database access for different divisions of a company that are sub-folders of a web site, you could create a region for each division. For the majority of web sites, shared access within a domain is preferred and allowed, therefore the default region (/ - the root level) is used.

To add a region, enter the region (sub-folder) path under "Create new region" and click the "Create..." button. The region path is based on your virtual host mapping. A region is any sub-folder under the mapping level of the domain for the region. For example, if your virtual host "domain.com" maps to the folder "/domain/" and you want to create a region for "/domain/store/", then you would enter "store" as the region path. However, if you have the domain "store.domain.com" mapped to the folder "/domain/store/", then the "store" folder is actually the root level ("/") of that domain.

NOTE: If you are using a virtual host application that modifies the URL rather than maintaining separate virtual roots (such as Welcome in normal mode or HomeDoor on a Macintosh), you must prepend the region name with the host folder path. For example, the Welcome admin requires a "host folder" for each virtual host. If the host folder for a domain is "/websites/company/" then the "store" region name should be entered as "/websites/company/store/". If you are running a "true" virtual hosting system (such as WebSTAR Virtual Host or Welcome in router mode on a Macintosh or an NT IIS virtual hosting system), there is no need to prepend information to the region name since the URL is not modified. For example, the "store" region should simply be entered as "/store/".



domain.com
: Region Management

/

/store/

Modify selected region.

Create new region

downloads

Region Preferences

Region preferences allow you to specify the language, the handling of errors, file uploads, the availability of tags in each MGI module, and the global region settings for specific MGI tags.

Unprocessed MGI Tags

If an MGI tag is mis-named or not available to the specified domain, MGI can pass the tag through or remove the tag from the page before it is served. To allow the tag to be passed through, click the checkbox beside "Do Not Remove Unprocessed MGI Tags" and click the "Save" button. To have the tag removed, unclick the checkbox beside "Do Not Remove Unprocessed MGI Tags" and click the "Save" button.

Embedded Error Checking

By default MGI does not check for errors in MGI tags that are embedded in HTML. To have MGI check and report errors in embedded tags, check the box beside the "Report Errors in Embedded MGI Tags". **Warning**: MGI may find embedding errors in scripts such as Javascript which are not actually errors and are acceptable coding. To remedy this problem, add spaces to the script near embedded tags (after open brackets "<") or turn this option off.

Error Pages

When MGI encounters an error, an error page is displayed. The default MGI error page is named "**Wow! I bet you didn't expect to see this...**" followed by the specific error and suggestions for remedy of the error. If there is a mistake in MGI code, the error page will highlight the code being processed in red to facilitate de-bugging. Custom error pages have the option of displaying any or all parts of an error using the mgiGetErrorParameter tag. To use the default MGI error page for the region, click the radio button beside "Use Default MGI Error Page" and click the "Save" button. To use a custom error page in the region, click the radio button beside "Use...", enter the path to the custom error page (relative to the root of the region), and click the "Save" button.

Default Index Pages

In WebSTAR (Mac) only, you have the option of using the default index pages defined in the web server admin or using the default index pages defined for the region of the specified web site. By default, the region is set to use the "Web Server-Defined Default Pages". To use the default index pages listed for the region, select the radio button beside "Use". Enter each default index page on one line with a return. For example, you may want the "index.html", "index.htm", and "index.mgi" pages to be served by default when a specific index page is not entered. Default pages are checked and served in the order entered.

HTTP File Uploads

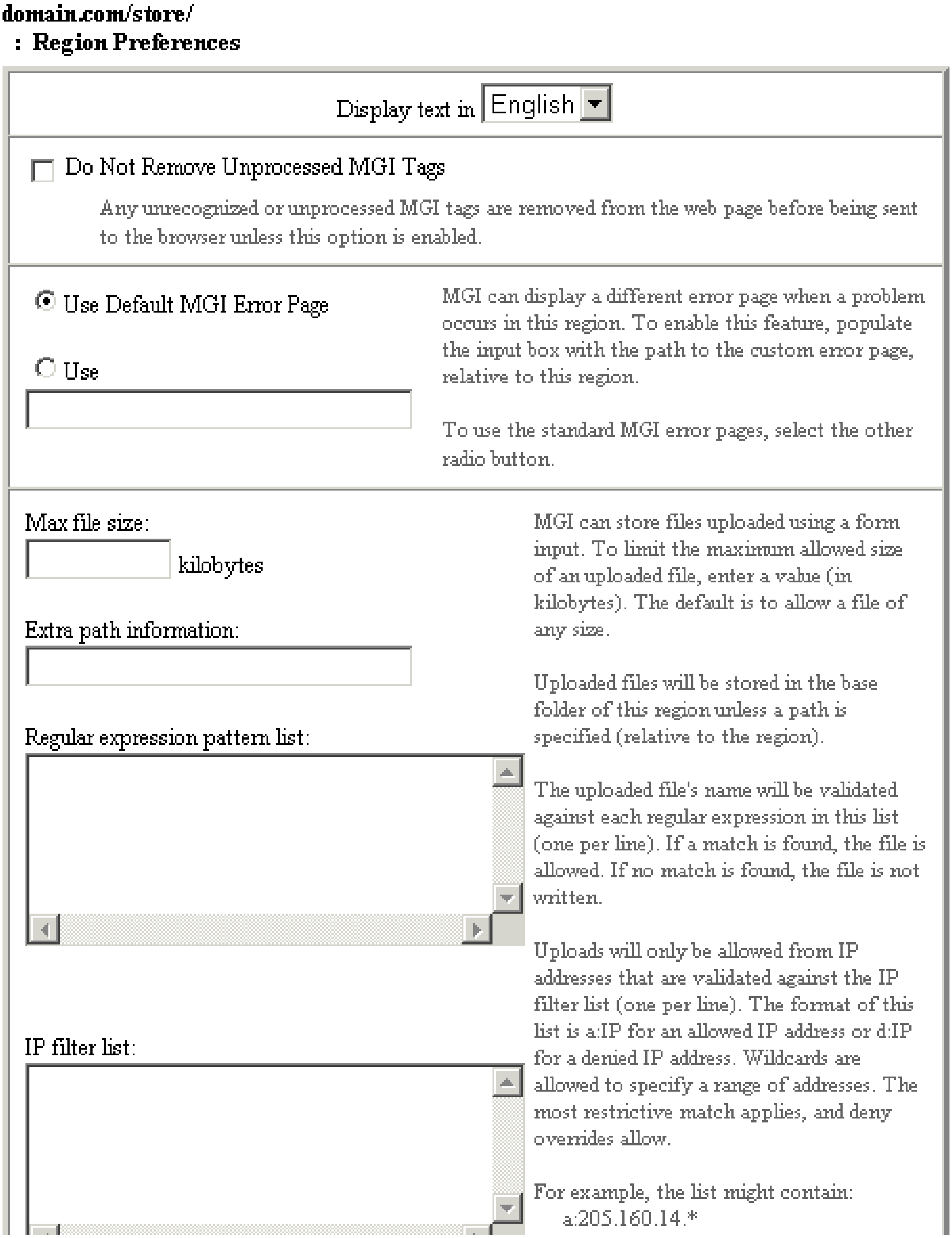
MGI supports file uploads from HTML file input elements. In the Region Preferences, you may set a maximum file size for uploaded files, a specific folder for uploaded files, a validation for file names and allow or deny IP addresses (including wildcards).

To set a maximum file size, enter the maximum file size in kilobytes under "Max file size:" and click the "Save" button.

To specify a folder for uploaded files, enter the path to the folder (relative to the root of the region) under "Extra path information:" and click the "Save" button.

To validate the file name before files are uploaded, enter one or more regular expression patterns under "Regular expression pattern list" and click the "Save" button. See the regular expression symbol list for more information.

For security resons you are required to list at least one IP address for file uploads, although you can use wildcard values. To allow an IP address, enter an "A" followed by a colon and the IP address (e.g., "a:205.160.14.88"). To deny an IP address, enter a "D" followed by a colon and the IP address (e.g., "d:205.160.14.99"). You may use an asterisk (*) for a wildcard value in any position of the IP address. For example, to allow all IP numbers from the 272 Class C block, you would enter "a:272.*.*.*" as a filter. For multiple filters, enter one filter per line. When a file upload request is received, the most restrictive match for the IP address will be processed and if two filters match, a deny filter will override an accept filter. For example, if you wish to allow all IPs from the 205.160.14 address except the IP address "205.160.14.233", then you would enter an allow and deny statement in the IP filter list ("a:205.160.14.*" and "d:205.160.14.233"). To allow all IP addresses, use wildcard values for all positions (i.e., "a:*.*.*.*").



domain.com/store/
: Region Preferences

Display text in: English

□ Do Not Remove Unprocessed MGI Tags

Any unrecognized or unprocessed MGI tags are removed from the web page before being sent to the browser unless this option is enabled.

● Use Default MGI Error Page

○ Use

MGI can display a different error page when a problem occurs in this region. To enable this feature, populate the input box with the path to the custom error page, relative to this region.

To use the standard MGI error pages, select the other radio button.

Max file size: kilobytes

Extra path information:

Regular expression pattern list:

IP filter list:

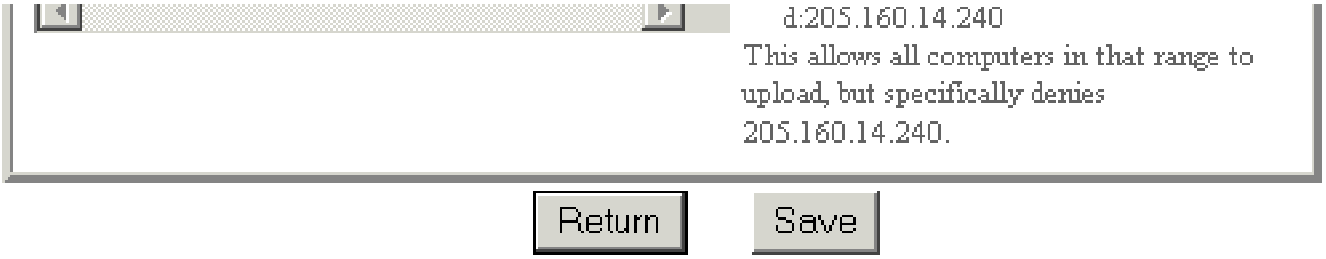
MGI can store files uploaded using a form input. To limit the maximum allowed size of an uploaded file, enter a value (in kilobytes). The default is to allow a file of any size.

Uploaded files will be stored in the base folder of this region unless a path is specified (relative to the region).

The uploaded file's name will be validated against each regular expression in this list (one per line). If a match is found, the file is allowed. If no match is found, the file is not written.

Uploads will only be allowed from IP addresses that are validated against the IP filter list (one per line). The format of this list is a:IP for an allowed IP address or d:IP for a denied IP address. Wildcards are allowed to specify a range of addresses. The most restrictive match applies, and deny overrides allow.

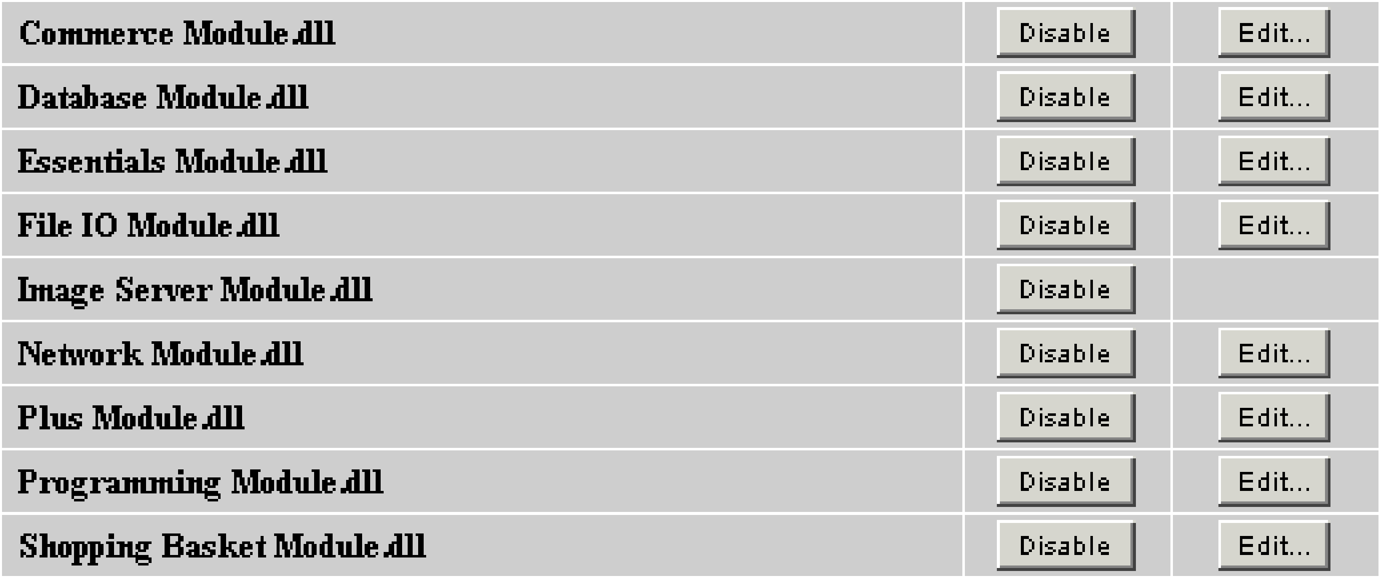
For example, the list might contain:
a:205.160.14.*



d:205.160.14.240
This allows all computers in that range to upload, but specifically denies 205.160.14.240.

Region Module Preferences

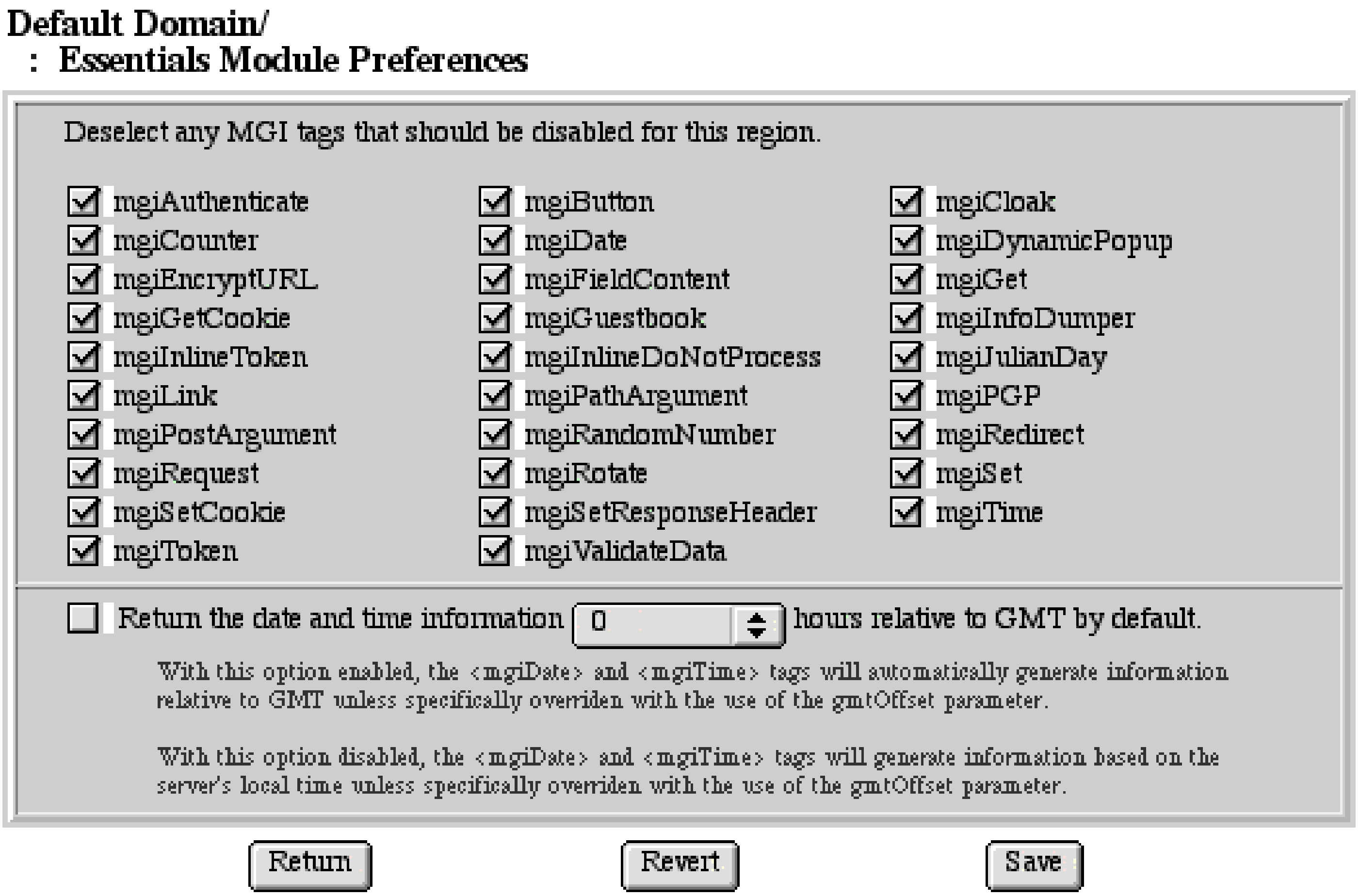
At the bottom of the Region Preferences window the modules available for the selected region are listed. A module disabled in the Domain Management window will not be listed in the Region Preferences window. By default the region has access to all modules. To disable the region's access to a module, click the "Disable" button beside the module. When disabled the module name displays in plain text with a line through the name. To enabled a module, click the "Enable" button beside the module. When enabled the module name displays in bold text. Module management at the region level affects only the specified region and does not affect the server module settings, the domain module settings or module settings for other regions.



Commerce Module.dll	Disable	Edit...
Database Module.dll	Disable	Edit...
Essentials Module.dll	Disable	Edit...
File IO Module.dll	Disable	Edit...
Image Server Module.dll	Disable	
Network Module.dll	Disable	Edit...
Plus Module.dll	Disable	Edit...
Programming Module.dll	Disable	Edit...
Shopping Basket Module.dll	Disable	Edit...

To configure the module preferences, click the "Edit..." button beside the module name. Under the Module Preferences window, the MGI tags in each module are listed. By default the region has access to all tags in each module. To disable access to a specific MGI tag within a module, uncheck the box beside the tag name and click the "Save" button. To enabled an MGI tag within a module, check the box beside the tag name and click the "Save" button or click the "Revert" button to return all settings to default.

Additional settings for specific MGI functions such as database search result limits, date and time GMT offsets, and mail server settings may also be entered in the module preferences. If you change the module preferences, click the "Save" button to save the settings. Preference changes take effect immediately.



Default Domain/
: Essentials Module Preferences

Deselect any MGI tags that should be disabled for this region.

☑ mgiAuthenticate ☑ mgiButton ☑ mgiCloak
☑ mgiCounter ☑ mgiDate ☑ mgiDynamicPopup
☑ mgiEncryptURL ☑ mgiFieldContent ☑ mgiGet
☑ mgiGetCookie ☑ mgiGuestbook ☑ mgiInfoDumper
☑ mgiInlineToken ☑ mgiInlineDoNotProcess ☑ mgiJulianDay
☑ mgiLink ☑ mgiPathArgument ☑ mgiPGP
☑ mgiPostArgument ☑ mgiRandomNumber ☑ mgiRedirect
☑ mgiRequest ☑ mgiRotate ☑ mgiSet
☑ mgiSetCookie ☑ mgiSetResponseHeader ☑ mgiTime
☑ mgiToken ☑ mgiValidateData

□ Return the date and time information 0 hours relative to GMT by default.

With this option enabled, the <mgiDate> and <mgiTime> tags will automatically generate information relative to GMT unless specifically overriden with the use of the gmtOffset parameter.

With this option disabled, the <mgiDate> and <mgiTime> tags will generate information based on the server's local time unless specifically overriden with the use of the gmtOffset parameter.

[Region Management] [Security]

[System Requirements] [Installation] [Domain Configuration] [Server Admin] [Domain Admin] [Licensing]

[MGI Guides Main Menu] [Admin Guide Main Menu]

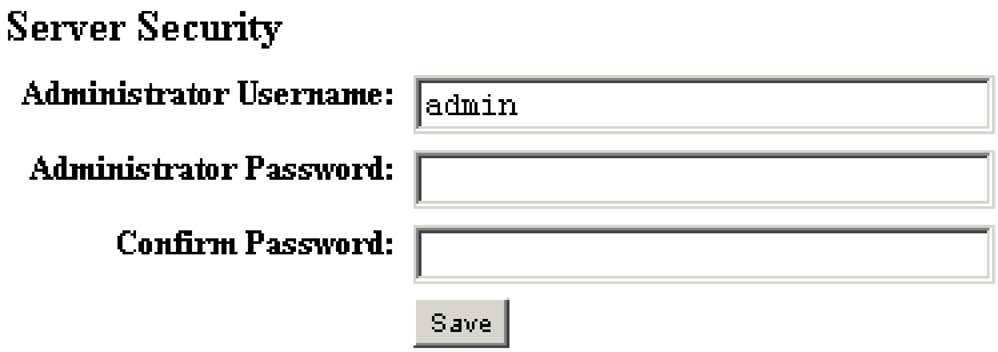
MGI Domain Admin - Security

MGI security settings allow you to change the username and password for the domain admin.

Server Security

To change the username and password, enter a new username, a new password, and a verification of the password, then click the "Save" button. The username and password are both case-sensitive.

All new domains inherit the current Server Admin username and password as the Domain Admin username and password. However, changing the Server Admin username and password does not change the Domain Admin username and password of existing domains!



Administrator Username: admin

Administrator Password:

Confirm Password:

[Region Management] [Security]

[System Requirements] [Installation] [Domain Configuration] [Server Admin] [Domain Admin] [Licensing]

[MGI Guides Main Menu] [Admin Guide Main Menu]

MGI Licensing Agreement

The license agreement contained herein is for informational purposes only. The actual license agreement that you will be required to accept or decline is contained in the downloadable version of MGI.

License Agreement

PLEASE READ THIS DOCUMENT CAREFULLY BEFORE INDICATING YOUR ACCEPTANCE BY CHOOSING THE "ACCEPT" BUTTON. THIS AGREEMENT LICENSES THE ENCLOSED SOFTWARE TO YOU AND CONTAINS WARRANTY AND LIABILITY DISCLAIMERS. BY CONTINUING WITH THE INSTALLATION, YOU ARE CONFIRMING YOUR ACCEPTANCE OF THE SOFTWARE AND AGREEING TO BECOME BOUND BY THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT WISH TO DO SO, CHOOSE THE "REJECT" BUTTON. YOU WILL NOT BE ABLE TO USE THE SOFTWARE.

This Agreement is by and between PagePlanet Software, Inc. (hereinafter "PagePlanet") and you. For good and valuable consideration, the receipt and sufficiency of which are hereby acknowledged, and which includes the mutual promises contained herein, the parties agree as follows:

1. Definitions.

1.1. Modular Gateway Interface (hereinafter "MGI") means the software programs, associated media, documentation (including printed materials and any associated online or electronic documentation), and any multimedia assets (including but not limited to images, graphics, photographs, animations, video, sounds, applets, and behaviors) included in or available as part of the enclosed package or downloadable executable, as well as all related updates supplied by PagePlanet. From time to time PagePlanet may release new versions of MGI which contain improvements. MGI includes both the original version and the new version(s) of MGI.

2. License.

2.1. MGI is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. MGI is licensed and not sold. This Agreement grants you a limited, non-exclusive license to:

2.1.1. Use MGI on a single computer.

2.1.2. Make one copy of MGI in machine-readable form solely for backup purposes. You must reproduce on any such copy all the copyright notices and any other proprietary legends on the original copy of MGI.

2.1.3 Run one copy of MGI on a secure server for each copy of MGI licensed and running on a non-secure server.

2.1.4. Certain software products are licensed with additional rights as set forth in the Supplementary Rights Addendum that may be included in MGI.

2.2. In the event of an emergency, the holder of a license of MGI is permitted to install and use a second or subsequent copy of MGI apart from the legal use explicitely described in paragraph 2.1.3 with the understanding that the license for the second or subsequent copy of MGI will be purchased from PagePlanet Software, Inc. no later than three business days after the installation.

2.3. Subject to Section 2.2, your license will automatically terminate upon any transfer of MGI. Upon transfer, you must deliver MGI, including any copies and related documentation, to the transferee. The transferee must accept these Terms and Conditions as a condition precedent to a valid transfer.

3. Supplementary Licenses. Certain rights are not granted under this Agreement, but may be available under a separate agreement.

3.1. Multi-User License. You must enter into a Multi-User License if you wish to make copies of MGI for use with additional CPUs owned by you. The Multi-User License allows you and other individuals in your organization to use MGI in perpetuity or for annual renewable periods, whether loaded on separate computers or a network server, as well as on any home or portable computers that any of you may possess, but only so long as each of you uses MGI on just one computer at any time and your organization has paid us our then-current multi-user license fee for the total number of individuals within your organization who will use MGI.

4. Restrictions.

4.1. You may not make or distribute copies of MGI, or electronically transfer MGI from one computer to another or over a network. You may not decompile, reverse engineer, disassemble, or otherwise reduce MGI to a human-perceivable form. You may not modify, rent, resell for profit, distribute, or create derivative works based upon MGI or any part thereof. MGI Output may contain multimedia assets available within MGI (including but not limited to images, graphics, photographs, animations, video, sounds, applets, and behaviors) but these assets may not be distributed separately. You will not export, directly or indirectly, MGI to any country prohibited by the United States Export Administration Act and the regulations thereunder.

4.2. PagePlanet may provide you with support services related to MGI ("Support Services"). Use of Support Services is governed by the MGI polices and programs described in the user manual, in on line documentation, or other PagePlanet-provided materials. Any supplemental software code provided to you as part of the Support Services shall be considered part of MGI and subject to the terms and conditions of this Agreement. With respect to technical information you provide to PagePlanet as part of the Support Services, PagePlanet may use such information for its business purposes, including for product support and development.

4.3. Without prejudice to any other rights, PagePlanet may terminate this Agreement without prior notice if you fail to comply with its terms and conditions. In such event, you must destroy all copies of MGI and all of its component parts.

5. Ownership. The foregoing license gives you limited rights to use MGI. Although you own the disk on which MGI is recorded, if any, you do not become the owner of, and PagePlanet retains title to, MGI (including but not limited to images, graphics, photographs, animations, video, sounds, applets, and behaviors incorporated into MGI), and all copies thereof. All rights not specifically granted in this Agreement, including Federal and international trademarks and copyrights, are reserved by PagePlanet.

6. Limited Warranties.

6.1. PagePlanet warrants that, for a period of ninety (90) days from the date of delivery (as evidenced by a copy of your receipt): (i) when used with a recommended hardware configuration, MGI will perform in substantial conformance with the documentation supplied; and (ii) that the media on which MGI is furnished will be free from defects in materials and workmanship under normal use. EXCEPT AS SET FORTH IN THE FOREGOING LIMITED WARRANTY, PAGEPLANET DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IF APPLICABLE LAW IMPLIES ANY WARRANTIES WITH RESPECT TO THE PAGEPLANET PRODUCT, ALL SUCH WARRANTIES ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF DELIVERY. No oral or written information or advice given by PagePlanet, its dealers, distributors, agents, or employees shall create a warranty or in any way increase the scope of this warranty.

6.2. SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER LEGAL RIGHTS WHICH VARY FROM STATE TO STATE.

7. Exclusive Remedy. Your exclusive remedy under Section 6 is to return MGI to the place you acquired it along with a copy of your receipt and a description of the problem. PagePlanet will use reasonable commercial efforts to supply you with a replacement copy of MGI that substantially conforms to the documentation, provide a replacement for the defective media, or refund your purchase price for MGI, at its option. PagePlanet shall have no responsibility with respect to MGI that has been altered in any way, if the media has been damaged by accident, abuse, or misapplication, or if the nonconformance arises out of use of MGI in conjunction with software not supplied by PagePlanet.

8. Limitations of Damages.

8.1. PAGEPLANET SHALL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS, LOSS OF PROFITS, OR THE LIKE), WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), PRODUCT

LIABILITY OR OTHERWISE, EVEN IF PAGEPLANET OR ITS REPRESENTATIVES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND EVEN IF A REMEDY SET FORTH HEREIN IS FOUND TO HAVE FAILED OF ITS ESSENTIAL PURPOSE.

8.2. PagePlanet's total liability for actual damages for any cause whatsoever caused by or alleged to have been caused by MGI will be limited to the amount paid by you for MGI

8.3. SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OF CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

9. Basis of Bargain. The limited warranty, exclusive remedies, and limited liability set forth above are fundamental elements of the basis of the bargain between PagePlanet and you. PagePlanet would not be able to provide MGI on an economic basis without such limitations.

10. General.

10.1. This Agreement shall be governed by the internal laws of the State of North Carolina. This Agreement contains the complete agreement between the parties with respect to the subject matter hereof, and supersedes all prior or contemporaneous agreements or understandings, whether oral or written. All questions concerning this Agreement shall be directed to: PagePlanet Software, Inc., 3252 Octavia Street, Raleigh, North Carolina 27606, Attention: Valerie Crisp.

10.2. Any legal action or proceeding relating to this Agreement shall be instituted in any state or federal court in Wake County, North Carolina. PagePlanet and Customer agree to submit to the jurisdiction of, and agree that venue is proper in, the aforesaid courts in any such legal action or proceeding. If at any time or times hereafter PagePlanet employs counsel to pursue collection, to intervene, to pursue enforcement (by any lawful means) of the terms of this Agreement, or to file a petition, complaint, answer, motion, injunction, or other pleading in any suit or proceeding relating to this Agreement, then in such event, all of the reasonable attorneys' fees costs of collection shall be an additional liability of Customer, payable on demand of PagePlanet.

10.3. If you are a unit or agency of the Government, or acquiring MGI with government funds, the software and documentation are provided subject to PagePlanet's standard commercial license; provided, however, that any contracts with non-defense agencies subject to the FAR, the Government shall have the rights set forth in subparagraph (c) of FAR 52.227-19, "Commercial Computer Software-Restricted Rights," as applicable.

11. Separate FairCom License Agreement for Embedded FairCom Products.

FairComÆ Server License Agreement

CAREFULLY READ THE TERMS OF THIS FairCom Server License Agreement (hereinafter called "Agreement") BEFORE OPENING THE ACCOMPANYING FAIRCOM MEDIA PACKAGE. OPENING THE FAIRCOM MEDIA PACKAGE INDICATES YOUR

CONCLUSIVE ACCEPTANCE OF THESE TERMS. IF YOU DO NOT AGREE WITH ANY TERM, CONDITION OR LIMITATION OF THIS AGREEMENT, PROMPTLY RETURN THE UNOPENED MEDIA PACKAGE AND ALL ITEMS ACCOMPANYING THE PACKAGE TO YOUR SUPPLIER. This is a legally binding and enforceable Agreement between you (which is you personally if you are acting on your own behalf or the entity on whose behalf you are acting) and FairCom Corporation, hereinafter called "FairCom".

1. Grant of License. Subject to all the terms, conditions and limitations set forth herein, FairCom grants you the non-exclusive right to use the FairCom Server (comprised of the accompanying FairCom Server software programs, utilities, and documentation, hereinafter collectively called the "Software"), on a single server computer with as many concurrent client computers as the Software is configured to support. You may terminate this Agreement at any time. In the event you violate any term of this Agreement, FairCom may, at its sole discretion, terminate this Agreement. Upon termination of this Agreement, you agree to continue to maintain the Software confidential, immediately destroy all copies of the Software (whether in whole or in part, whether or not modified, whether in source, object or binary executable format) and execute a software destruction affidavit, provided by FairCom upon request, indicating that you have complied with this section.

2. Proprietary Information. You understand and agree that the Software is and remains the confidential property of FairCom and is protected by United States copyright laws and international treaty provisions. You are hereby informed that the Software contains certain "Proprietary Information" which is solely owned by FairCom. "Proprietary Information" means the Software received by you from FairCom or from any third party whether or not under obligation with FairCom to maintain such information as confidential. You agree not to distribute, remove, disclose to any third party, copy or make summaries of the Proprietary Information or utilize said Proprietary Information for any purpose, except as specifically granted herein.

3. Title. The Software is licensed, not sold to you. This license does not transfer to you any ownership interest in the Software or any FairCom trademark or registered trademark, but only a limited right to use the Software strictly in accordance with the terms, conditions and limitations of this Agreement. This license does not permit you or any other person to:

a) disclose or transfer the Software to any person or entity, except as provided for in Section 6 herein; or
b) translate the Software to another computer language; or
c) modify, disassemble, decompile, or reverse engineer the Software in any manner whatsoever; or
d) modify the copyright and/or statements of confidentiality embedded in the Software.

4. Backup Copies. You are permitted to make a maximum of two copies of the Software for backup purposes only. You may not copy the Software for any purpose other than that specified herein. Media which contains the Software must display the following notice: This computer software is the confidential and proprietary property of FairComÆ Corporation. Any unauthorized use, reproduction or transfer of this computer software is strictly

computer software" as that term is defined in Section 27.401 of the Federal Acquisition Regulations and is "commercial computer software" as that term is defined in Section 227.401(1) of the Department of Defense Federal Acquisition Regulation Supplement ("DFARS"); and

b) the Government agrees that if the Software is supplied to the Department of Defense, the Government is acquiring no more than the minimum restricted rights in the Software as the term "restricted rights" is defined in Subparagraph (a)(17) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, and the Government agrees that the Software is marked as follows: RESTRICTED RIGHTS LEGEND. Use, duplication or disclosure by the Government is subject to restrictions as set forth in Subparagraphs (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227.7013. FairComÆ Corporation 2100 Forum Blvd., Suite C, Columbia, Missouri 65203-5456.

c) the Government agrees that if the Software is supplied to any unit or agency of the Government other than the Department of Defense, the Government's rights in the Software shall be no more than those rights set forth in Subparagraphs (c)(1) and (c)(2) of the Commercial Computer Software - Restricted Rights, at 48CFR52.227-19, and the Government agrees that the Software is marked as follows: This is an unpublished work and is subject to limited distribution and restricted disclosure only. All Rights Reserved.

11. Export Restrictions. You may not export or reexport the Software or any file created with the Software except as authorized by United States law and the laws of the jurisdiction in which the Software was obtained. In particular, but without limitation, the Software or any file created with the Software may not be exported or reexported into (or to a national or resident of) Cuba, Iran, Iraq, Libya, North Korea, Sudan, Syria or any other U.S. embargoed country or to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce's Table of Denial Orders. Pursuant to Article 6 of the United Nations Convention on Contracts for the International Sale of Goods (UN Convention), the parties agree that the UN Convention shall not apply to this Agreement.

12. Governing Law. This Agreement is deemed to have been executed and entered into in Boone County, Missouri and shall be construed in accordance with the laws of the State of Missouri. Any dispute concerning or arising out of this Agreement shall be commenced and prosecuted in the appropriate forum or court located in Boone County, Missouri and you hereby consent to personal jurisdiction in such forum or court.

13. Access. You agree and authorize FairCom, or its authorized representative, access to any location where the Software is being used in order to verify that your use of the Software complies with all of the terms, conditions and limitations set forth in this Agreement.

14. Severability. Should any provision of this Agreement be declared void or unenforceable by any court of competent jurisdiction, such declaration shall have no effect on the remaining terms of this Agreement.

15. Waiver. No failure to exercise, and no delay in exercising, on the part of either party, any privilege, power or right herein will operate as a waiver thereof, nor will any single or partial

prohibited. Copyright 1984-2000, FairCom Corporation. This is an unpublished work and is subject to limited distribution and restricted disclosure only. All Rights Reserved.

5. Assignment of License. You may not rent or lease the Software, but you may permanently transfer all of your rights under this Agreement, provided that you retain no copies of the Software, transfer all of the Software, and the recipient provides to FairCom written acceptance of all of the terms of this Agreement prior to the transfer. If the Software is an upgrade, any transfer must include all prior versions.

6. Trademarks. "FairCom" and FairCom's circular disc logo are registered trademarks of FairCom in the United States and other countries. No right or ownership interest to such trademarks is granted to you herein. You hereby agree that you will not use these trademarks, without the express written consent of FairCom.

7. Disclaimer of Warranties. The entire risk as to the quality and performance of the Software is with you. Should the Software prove defective, you assume the entire cost of all necessary servicing, repair and correction. FairCom does not warrant that the technology contained in the Software will meet your requirements or that the operation of the Software will be uninterrupted or error free. THE SOFTWARE IS LICENSED "AS IS", AND FAIRCOM DISCLAIMS ALL OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OR MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

8. Limitation of Liability and Remedies. FairCom's cumulative and total liability to you or any other party for any claim, demand or action arising from or related to this Agreement or your use of the Software (whether in contract, warranty, tort (including negligence), product liability, patent or copyright infringement or any other theory whatsoever), including any damages from computer viruses, shall not exceed the license fee paid to FairCom for the use of the Software even if you paid no fee. In no event shall FairCom be liable for any indirect, incidental, consequential or special damages, exemplary damages, lost profits, or data loss even if FairCom is advised of the possibility of such damages in advance and not withstanding the failure of the essential purpose of any remedy. Any cause of action brought by you under this agreement, whether in contract, tort or otherwise, shall be commenced no later than one (1) year after such right of action accrues and may not be extended for any reason. This limitation of liability and risks is reflected in the price of the software license.

9. Indemnification. You shall defend, indemnify and hold FairCom harmless from any and all claims, damages, losses, liabilities, costs and expenses (including fees of lawyers and other professionals) arising out of or in connection with your use of the Software, whether direct or indirect, including, but not qualified to data loss, business interruption and computer viruses. You shall immediately notify FairCom of any such claim.

10. U.S. Government Restricted Rights. If you are acting on behalf of any unit or agency of the government of the United States of America, hereinafter called the "Government", the following provisions apply:

a) the Government acknowledges FairCom's representation that the Software is "restricted exercise of any right or power herein preclude further exercise of any right herein.

16. Entire Agreement. This Agreement constitutes the entire understanding between the parties hereto with respect to the subject matter hereof and supersedes any and all prior or contemporaneous oral or written agreements, understandings, representations or communications between the parties.

17. Survival of Obligations. All of your obligations and responsibilities under this Agreement shall survive its termination for any reason. 000822

THIS DOCUMENT IS A LEGALLY BINDING CONTRACT. BY CLICKING THE "YES" BUTTON, YOU SPECIFICALLY ASSENT TO AND ACCEPT ALL OF THE TERMS AND CONDITIONS OF THE SOFTWARE LICENSE AGREEMENT AND SHALL BE DEEMED AS THE LICENSEE UNDER THE TERMS OF THIS AGREEMENT. MGI IS MADE AVAILABLE FOR DOWNLOADING SOLELY FOR USE BY CUSTOMERS OF PAGEPLANET SOFTWARE, INC. ANY REPRODUCTION OR REDISTRIBUTION OF MGI NOT IN ACCORDANCE WITH THESE TERMS OF USE IS EXPRESSLY PROHIBITED BY LAW. YOU MAY NOT DOWNLOAD OR INSTALL ANY SOFTWARE FROM THIS SERVER UNLESS YOU AGREE TO THESE TERMS OF USE IN THEIR ENTIRETY.

[System Requirements] [Installation] [Domain Configuration] [Server Admin] [Domain Admin] [Licensing]

[MGI Guides Main Menu] [Admin Guide Main Menu]

MGI User Guide

This MGI User Guide includes:

- **Understanding MGI --** a basic overview of MGI.
- **Using MGI --** a step-by-step tutorial for typical uses of MGI in your web site.
- **Referencing MGI --** term definitions and a technical description of each MGI tag.

Understanding MGI

In the Understanding MGI section, you will understand how MGI processes the pages of your web site, how MGI handles errors in your web site, how MGI uses the MGI Data folder, and you will understand the basic structure of regular MGI tags and embedded MGI tags. Read the Understanding MGI Section first if you are a new MGI user.

Using MGI

In the Using MGI section, you will learn how to use MGI in your web site by following step-by-step instructions for typical MGI functions such as displaying banner ads, processing forms to email, searching databases, and shopping online. You will begin programming MGI immediately without memorizing every option of an MGI tag. Read the Using MGI section to get started using MGI or to review a typical MGI function.

Referencing MGI

In the Referencing MGI section, you will find a complete list of definitions and a technical reference for each MGI tag. Each MGI tag was written for a specific purpose, but you can choose tag options to customize the tag's function and you can combine tags to create new functions. Read the Referencing MGI section to customize the function of MGI for your web site.

[MGI Guides Main Menu] [User Guide Main Menu]

Understanding MGI

In the Understanding MGI section, you will understand how MGI processes the pages of your web site, how MGI handles errors in your web site, how MGI uses the MGI Data folder, and you will understand the basic structure of regular MGI tags and embedded MGI tags. Read the Understanding MGI Section first if you are a new MGI user.

Processing MGI Tags

> How the MGI tags in your web pages are processed by the Web Server and the MGI Plug-In.

Handling MGI Errors

> How MGI notifies you of errors through pre-formatted MGI error pages and custom error pages.

Writing MGI Tags

> The proper syntax for writing MGI tags.

Using MGI Tags

> Rules for using MGI tags in the pages of your web site.

Embedding MGI Tags

> How to embed one MGI tag within another MGI tag.

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Processing MGI Tags

MGI processes files wiith extensions specified by the server's suffix mapping settings. Any file extension can be mapped and processed by MGI. The ".mgi" extension is commonly set as a suffix for MGI processing.

Processing means that MGI scans the HTML code of the page looking for MGI tags. Tags are parsed (read by the MGI program) in a specific order -- from the top to the bottom of a page and from the inside out (i.e. the body of a tag is parsed before the rest of the tag, and embedded tags are parsed before the tags in which they reside). If an MGI tag is encountered during the scan, MGI parses the tag, replacing it with the appropriate information **before** the page is served (e.g. an mgiCounter tag is parsed and replaced with a string of numbers).

Because MGI performs its functions before the page is served, MGI tags will not be visible to the visitor viewing the HTML source code of a page served by MGI. **To avoid re-entering MGI tags, it is advisable to work with new, original HTML code only, rather than code downloaded via a browser's "Save As" command.**

[Processing Tags] [Errors] [Writing Tags] [Using Tags] [Embedding Tags]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Handling MGI Errors

If MGI encounters an error while parsing a page, an error page displays. The type of error page that displays depends on the region preferences setting for the region in which the error occurred.

If the region preferences are set to "Use Default MGI Error Page", then a pre-formatted MGI error page is served when an error is encountered. Each error page is named **"Wow! I bet you didn't expect to see this..."** followed by the specific error and suggestions for remedy of the error. If there is a mistake in MGI code, the error page will highlight the code being processed in red to facilitate de-bugging.

If the region preferences are set to use a custom error page, then the custom error page displays when an error is encountered. Custom error pages have the option of displaying any or all parts of an error using the mgiGetErrorParameter tag.

[Processing Tags] [Errors] [Writing Tags] [Using Tags] [Embedding Tags]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Writing MGI Tags

MGI incorporates powerful back-end processing with simple HTML-style tags. No matter how complex those back-end processes are, the tags stay simple and easy to use. To facilitate ease of use, there is a common syntax for all MGI tags.

MGI Tag Syntax

Each MGI tag has three distinct components:

1. Opening and Closing Characters:

> Each MGI tag opens with the less than ("<") character and closes with the greater-than (">") character, just like regular HTML tags.

2. Tag Name:

> The first component of a tag within the opening and closing characters is the name of the tag. That name always starts with "mgi". For instance, the counter tag is <mgiCounter>, the shopping basket tag is <mgiShoppingBasket>, and so forth.

3. Parameters:

> Some tags have additional components known as **parameters**. Parameters ("arguments" for those versed in programming) allow you to specify the behavior of an MGI tag. There are two types of parameters: **required** and **optional**. Required parameters must always be specified and included with the associated MGI tag.
>
> You can choose to include or not include optional parameters - they are optional after all! Most optional parameters do, however, have a default behavior that determines how a tag will function if the optional parameter is not specified. For example, if an optional parameter can cause an MGI tag to do one of four things (let's call them 1, 2, 3, and 4), then typically option "1" will be the default behavior. If you do not specify the optional parameter, it will automatically behave as if you had specified option "1". To change the default behavior, you must include the optional parameter and its designated value with the MGI tag. You can also positively indicate that you want the default behavior by including the optional tag and specifying the default behavior.

Some MGI tags consist of two tags, a **beginning tag** and an **ending tag**. A beginning tag follows the syntax described above with required and optional parameters. An ending tag always consists of an opening character, a forward slash (/), the name of the MGI beginning tag, and the closing character. The space between a beginning and ending tag is known as the **body** and can usually consist of text , HTML and/or other MGI tags. For example, this mgiSendMail tag includes a beginning tag with three required parameters, an optional parameter, a body, and an ending tag. In the following example, text and mgiPostArgument tags are included in the body of the mgiSendMail tag.

```
<mgiSendMail to="info@domain.com" from="webmaster@domain.com"
subject="Info Request" mailserver="mail.domain.com">

Info Request
************

    Name: <mgiPostArgument name="Name">
 Address: <mgiPostArgument name="Street Address">
    City: <mgiPostArgument name="City">
   State: <mgiPostArgument name="State">
     Zip: <mgiPostArgument name="Zip Code">
   Phone: <mgiPostArgument name="Phone">
   Email: <mgiPostArgument name="Email Address">

Comments: <mgiPostArgument name="Comments">

</mgiSendMail>
```

[Processing Tags] [Errors] [Writing Tags] [Using Tags] [Embedding Tags]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Using MGI Tags

The proper format of an MGI tag is:

```
<mgiNameOfTag RequiredParameter="value of parameter"
RequiredParameter2="value" OptionalParameter1="value"
OptionalParameter2="value">
```

Spacing

Note that there is no space between the name of the tag and the opening character and that there is also no space before the closing character (which may occur directly after the tag name or occur after the last parameter, depending on the tag). There is one space between the name of the tag and the first parameter and one space between each parameter. Do not string different parameters together. Do not use tabs. Do not use periods. Keep it simple -- just use single spaces. Within the value of a parameter (enclosed in quotation marks), you may use spaces. For example, there are spaces in the name parameter of this mgiButton tag:

```
<mgiButton name="Submit Order">
```

Parameter Format

Each parameter has a specific syntax. A parameter begins with the name of the parameter, followed by an equal (=) sign, followed by the value of the specific parameter within quotation marks. The use of the word "value" in the examples above is just an example and does not indicate an actual option for each parameter. A list of options for all parameters (required and optional) are included in the tutorial for each tag.

Some MGI tags have no required or optional parameters at all. In those cases, all you need is the name of the tag itself enclosed within opening (<) and closing (>) characters. Other tags may have more than one required parameter. In that case, just separate each parameter with a single space. Some tags will have many optional parameters. Again, just separate each parameter that you want to include with a single space.

Most required and optional parameters can also be written in any order. The only thing that has to be specifically placed is the actual name of the MGI tag -- it must go first, immediately following the opening character of the tag.

Case-Sensitivity

MGI tags and parameter names are **not** case-sensitive -- we use both upper- and lower-case letters in this guide to make the tags easier to read.

File Paths

The value of some MGI parameters is an absolute or relative file path. For absolute file paths, you must provide the full URL to the page (e.g., "http://www.domain.com/folder/page.mgi").

For relative file paths, you must only provide the name of the page (e.g., "page.mgi") or the path to the page in an associated directory (e.g., "folder/page.mgi"). Using MGI, relative paths may reference files below the current directory or above the current directory to the **root of the region** (e.g., "../page.mgi").

Nesting MGI Tags and HTML Elements

Many MGI tags and HTML elements require a begining and ending tag. When nesting an MGI tag or HTML element within another MGI tag or HTML element, use proper nesting such that the beginning and ending tags match (when required) and the order of beginning and ending tags is correct. Failing to match and order tags and elements can lead to unreliable results.

Incorrect Order:

```
<font size="+2"><b>Shopping Basket</font></b>
```

Correct Order:

```
<font size="+2"><b>Shopping Basket</b></font>
```

Incorrect Order:

```
<mgiIf lhs={mgiPostArgument name="email"}
relationship="isNotEmpty">
<p>Thank you for your application.
<mgiElse>
</p>
<p>The email address is missing.
</mgiIf>
</p>
```

Correct Order:

```
<mgiIf lhs={mgiPostArgument name="email"}
relationship="isNotEmpty">
<p>Thank you for your application.</p>
<mgiElse>
<p>The email address is missing.</p>
</mgiIf>
```

This Nesting Example is also in the correct order since ending HTML paragraph elements are optional:

```
<mgiIf lhs={mgiPostArgument name="email"}
relationship="isNotEmpty">
<p>Thank you for your application.
<mgiElse>
<p>The email address is missing.
</mgiIf>
```

Non-Matching Beginning and Ending Tags:

```
<mgiSet name="ID">
  <mgiPostArgument name="name">
<mgiSet name="Type">
  <mgiPostArgument name="custtype">
</mgiSet>
```

Matching Beginning and Ending Tags:

```
<mgiSet name="ID">
  <mgiPostArgument name="name">
</mgiSet>

<mgiSet name="Type">
  <mgiPostArgument name="custtype">
</mgiSet>
```

Coding MGI Tags in HTML Files

MGI tags (like other HTML tags) should be coded using a text editor with raw HTML text files. Even if you use a graphic HTML editor to create web site pages, you should use the raw HTML option or a separate text editor to add MGI tags. For example, MGI tags should be coded in the "HTML Source" view of PageMill and surrounded by NoEdit tags to insure that PageMill does not make automatic corrections or changes to the tags:

```
<!--NOEDIT--><mgicounter name="homepage"><!--/NOEDIT-->
```

If MGI tags are added in the graphic view of an HTML editor, the opening and closing tag characters and the quotations marks are often converted from text to HTML character entities (e.g. "&quot;" represents a quotation mark in HTML). MGI cannot be parsed correctly if the tag components are converted to such character entities.

[Processing Tags] [Errors] [Writing Tags] [Using Tags] [Embedding Tags]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Embedding MGI Tags

MGI also supports embedded tags. That is, you can use one MGI tag within the parameter value of another MGI tag or within the parameter value of an HTML tag. The embedded MGI tag replaces the entire value of the MGI or HTML parameter including the quotation marks:

```
<mgiNameOfTag Parameter={mgiEmbeddedTag Parameter="Value"}>
```

The syntax of embedded tags is:

1. Opening and Closing Braces

An embedded MGI tag is enclosed by braces (the squiggly ones) as in { }. An embedded MGI tag **does not** begin with an opening less-than character (<) and **does not** end with a closing greater-than (>) character.

2. Tag Name

The name of embedded MGI tags does not change.

3. Parameters

You may include an embedded tag's required and optional parameters. The syntax of parameters does not change in the embedded tag.

Embedded tags are often functional and useful. For example, you might replace the "from" parameter of the mgiSendMail tag with a visitor's email address so that you can reply directly to the visitor after their form submission is converted to an email:

```
<mgiSendMail mailserver="mail.domain.com" to="info@domain.com"
from={mgiPostArgument name="Email"
defaultValue="webmaster@domain.com"}
subject="Info Request">

Information Request Text and MGI Tags

</mgiSendMail>
```

Embedding Multiple Tags

Only one MGI tag can be embedded within the parameter of another MGI tag. An **extended embedded function** can be achieved using variables. By first setting information in the value of a page or site variable, you can include any combination of text, tags and HTML in the value of a parameter. For example, you may want to include several components in the subject of an mgiSendMail tag. First set the information in the value of a page variable using the mgiSet tag:

```
<mgiSet name="Email Subject">
<mgiFieldContent name="Product"><mgiFieldContent name="Style">
```

```
Request
</mgiSet>
```

Next, embed the value of the page variable in the subject parameter of the mgiSendMail tag using an mgiGet tag:

```
<mgiSendMail mailserver="mail.domain.com" to="info@domain.com"
from="webmaster@domain.com" subject={mgiGet name="Email Subject"}>
Information Request Text
</mgiSendMail>
```

See the mgiSet and mgiGet tag descriptions for more information.

Tags With Bodies

A tag with a body cannot be embedded within another MGI tag or an HTML tag. To embed the value of a tag with a body in another MGI tag or an HTML tag, use the variable method described under Embedding Multiple Tags above.

[Processing Tags] [Errors] [Writing Tags] [Using Tags] [Embedding Tags]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Using MGI

In the Using MGI section, you will learn how to use MGI in your web site by following step-by-step instructions for typical MGI functions such as displaying banner ads, processing forms to email, searching databases, and shopping online. You will begin using MGI immediately without memorizing every option of an MGI tag. Read the Using MGI section to get started using MGI or to review a typical MGI function.

The functions below are organized loosely by complexity with the simple functions at the top of the list and more complex functions at the bottom of the list. If you are a new MGI user, we suggest that you start with the simple functions and try more complex functions as you gain experience with using MGI. However, depending on the amount of time you can spend learning MGI and your goals for using MGI, you may want to start with the function that is most useful to you. Most examples do not assume that you have previous knowledge of MGI.

Displaying Page Counters

Learn how to display text and graphic counters on your web site. Learn advanced counter techniques including counter administration, custom counter graphics and hidden counters.

Processing Form Submissions

Learn how to send an email that contains information from a form submission. Learn advanced form processing techniques including verifying information in form fields, sending email attachments and sending encrypted emails.

Collecting Guestbook Entries

Learn how to collect, format and display visitors' comments and information in a guestbook. Learn advanced guestbook techniques including database-driven guestbooks.

Displaying Banner Ads

Learn how to schedule and display banner ads using a web-based interface. Learn advanced banner ad functions such as displaying client statistics.

HTTP File Upload

Learn how to upload files via HTTP file upload.

Password Protecting Pages (Authentication)

Learn how to password protect pages with a single username and password or via IP number. Learn advanced authentication techniques including using multiple usernames and passwords with user groups, importing and exporting authentication data, and form administration of usernames and passwords (adding, deleting, and verifying usernames, changing passwords, and sending "forgotten" passwords).

Administering Polls and Surveys

Learn how to collect and display survey information. Learn advanced poll techniques including custom results.

Administering Online Quizzes

Learn how to create, grade and manage online quizzes. Learn advanced quiz techniques including custom quiz grading.

Embedding MGI Tags

Learn how to embed one MGI tag within another MGI tag. Learn advanced embedding techniques including multiple embedded tags using variables.

Using Variables

Learn how to set and display temporary page variables and database-driven site variables.

Using If, Then, Else Comparisons

Learn how to make simple inline and regular if, then, else comparisons. Learn advanced conditional comparison techniques such as nested if statements.

Creating and Populating Databases

Learn how to create, modify, and delete databases, fields, and records. Learn advanced functions such as importing and exporting and advanced string searches.

Searching Databases and Displaying Results

Learn how to search a database and display a list of results that match the search criteria.

Shopping Online

Learn how to integrate a shopping basket system with hard-coded products and products displayed from a database search. Learn advanced online shopping techniques including tax and shipping rules, customizing the shopping basket display, encrypting orders, implementing inventory control, and authorizing credit cards via Accesspoint.

Using ODBC Databases

Learn guidelines for ODBC set up, tags that are affected by ODBC databases, and other isues that are specific to ODBC databases.

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Displaying Page Counters

In the **Beginner Tutorial** section, learn how to display text and graphic counters on your web site. In the **Advanced Tutorial** section, learn advanced counter techniques including counter administration, custom counter graphics and hidden counters. In the **Reference** section, view a complete technical reference for each tag used for counters. In the **Downloads** section, download example code for each beginner and advanced tutorial. In the **FAQs** section, read answers to frequently asked questions.

Beginner Counter Tutorials

- B1. Displaying Text Counters
- B2. Displaying Graphic Counters

Advanced Counter Tutorials

- A1. Counter Administration (Adding, Setting and Deleting Counter Values)
- A2. Displaying Graphic Counters with Custom Fonts
- A3. Counting Unique Visitors

Counter MGI Tag Reference

- mgiCounter

Counter Downloads

- B1. Download text counter examples (B1TextCount.zip - 1 KB)
- B2. Download graphic counter examples (B2GraphCount.zip - 1 KB)
- A1. Download counter admin page example (A1CounterAdmin.zip - 1 KB)
- A2. Download custom graphic counter examples (A2CustomGC.zip - 21 KB)
- A3. Download a unique IP address counter examples (A3Unique.zip - 1 KB)

Frequently Asked Counter Questions

- Q1. Why doesn't my counter display?
- Q2. Why does the mgiCounter tag show up on my page in a browser?
- Q3. Why doesn't my counter increment?

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Displaying Text Counters

Introduction

Counters tabulate the number of visitor's to a page. However, counters cannot determine unique visitors to a page (i.e, one visitor may visit the page multiple times).

Text counter values appear as text numbers. Graphic counter values appear as number graphics in a specified font.

In this example, a text counter is added to a web page.

MGI Tags

- mgiCounter

Steps

1. Open a page in a text editor .
2. Insert the mgiCounter tag.
3. Save the page.
4. FTP the page to the web server running MGI.
5. View the page in a web browser.

Step 1: Open a page in a text editor.

Open a web page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiCounter tag.

Enter the mgiCounter tag and name parameter where you want the counter value to appear. In the name parameter, enter the unique counter name. Counters are stored in an internal MGI database and values from multiple counters with the same name are added to the same database record.

```
<p>You are visitor number <mgiCounter name="index">.
```

Step 3: Save the page.

Save the changes you have made to the page.

Step 4: FTP the page to the web server running MGI.

Upload the page from your local computer to the web server using an FTP program.

Step 5: View the page in a web browser.

View the page in a web browser. The counter value displays "1" for the first visitor, "2" for the second visitor, etc. Counter values continue to increment until the counter value is reset using the counter admin.

[Return to the Displaying Page Counters Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Displaying Graphic Counters

Introduction

Counters tabulate the number of visitor's to a page. However, counters cannot determine unique visitors to a page (i.e, one visitor may visit the page multiple times).

Text counter values appear as text numbers. Graphic counter values appear as number graphics in a specified font.

In this example, a graphic counter is added to a web page.

MGI Tags

- mgiCounter

Steps

1. Open a page in a text editor .
2. Insert the mgiCounter tag.
3. Save the page.
4. FTP the page to the web server running MGI.
5. View the page in a web browser.

Step 1: Open a page in a text editor.

Open a web page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiCounter tag.

Enter the mgiCounter tag, name parameter, type parameter and font parameter where you want the counter value to appear. In the name parameter, enter the unique counter name. Counters are stored in an internal MGI database and values from multiple counters with the same name are added to the same database record. In the type parameter, enter "Graphic". In the font parameter, enter the font name you wish to display. The default font is "odometer". Additional built-in fonts include "DigitBlueGlow", "Greeny", "RedDigital", "SmallBlue", and "SaintFrancis". View font graphics.

```
<p>You are visitor number <mgiCounter name="about"
type="Graphic" font="Odometer">.
```

Step 3: Save the page.

Save the changes you have made to the page.

Step 4: FTP the page to the web server running MGI.

Upload the page from your local computer to the web server using an FTP program.

Step 5: View the page in a web browser.

View the page in a web browser. The counter value displays a "1" graphic for the first visitor, a "2" graphic for the second visitor, etc. Counter values continue to increment until the counter value is reset using the counter admin.

[Return to the Displaying Page Counters Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Counter Administration (Adding, Setting, and Deleting Counter Values)

Introduction

The admin mode of the mgiCounter tag displays a web-based administration screen for the internal MGI Counter Database. Although new counters are added automatically to the database by the mgiCounter tag, you may add new counters via the admin screen. In addition, you may set the value of a counter, reset the value of a counter to zero, or delete a counter from the database.

The following tutorial describes how to create and use a counter administration page.

MGI Tags

- mgiCounter

Steps

1. Create a counter administration page in a text editor.
2. Insert the mgiCounter tag in admin mode.
3. Save the counter administration page.
4. FTP the counter administration page to the web server running MGI.
5. View the counter administration page in a web browser.
6. How to add a new counter.
7. How to set the value of a counter.
8. How to reset the value of a counter.
9. How to delete a counter.

Step 1: Create a counter administration page in a text editor.

Create a new page in a text editing program to display the web-based counter administration interface.

Step 2: Insert the mgiCounter tag in admin mode.

On the counter administration page, enter the mgiCounter tag and mode parameter. In the mode parameter, enter "admin".

```
<mgiCounter mode="Admin">
```

Step 3: Save the counter administration page.

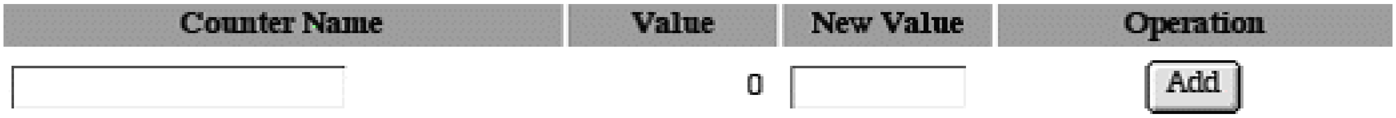
Save the counter administration page and name it "counteradmin.mgi".

Step 4: FTP the counter administration page to the web server running MGI.

Upload the counter administration page from your local computer to the web server using an FTP program.

Step 5: View the counter administration page in a web browser.

View the counter administration page in a web browser. The first screen of the web-based administration interface displays.

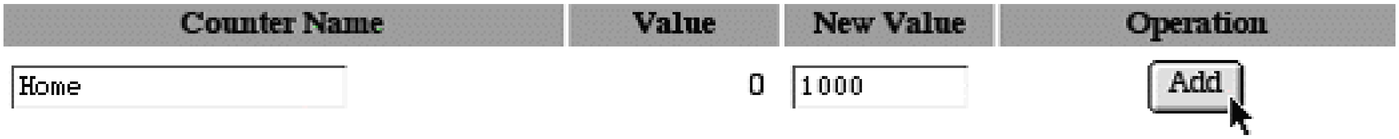


Counter Name	Value	New Value	Operation
	0		Add

Step 6: How to add a new counter.

To add a new counter, enter a counter name in the first field under the column labeled "Counter Name". If you wish to set the counter to a specific value, enter the counter value in the first field under the column labeled "New Value". If you do not enter a new value for the counter, it will begin at zero. To create the counter, click the "Add" button and the counter appears alphabetically in the list.

In this example, a new counter named "Home" is created and set at the value "1000".

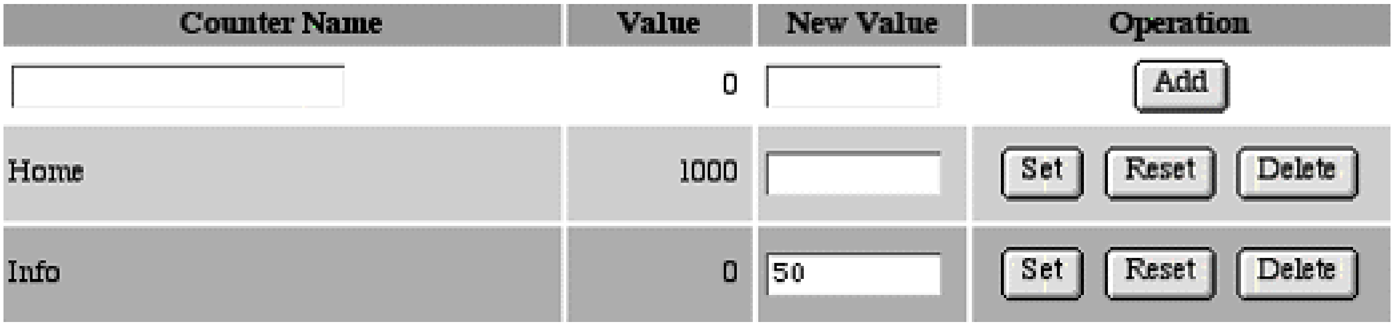


Counter Name	Value	New Value	Operation
Home	0	1000	Add

Step 7: How to set the value of a counter.

Once a counter is created manually using the counter administration interface or automatically by the mgiCounter tag, the counter value may be set to any number up to 9,999,999,999. To set a counter value, enter the new value in the field to the right of the counter name under the "New Value" column and click the "Set" button. The new counter displays in the "Value" column and is the basis for all subsequent counts.

In this example, the "Info" counter is set to "50".



Counter Name	Value	New Value	Operation
	0		Add
Home	1000		Set Reset Delete
Info	0	50	Set Reset Delete

Step 8: How to reset the value of a counter.

The reset feature of the counter administration interface will set any counter value to zero. To reset a counter value to zero, click the "Reset" button beside the counter name.

You may also reset a counter value by entering "0" in the "New Value" field and clicking the "Set" button.

Step 9: How to delete a counter.

The counter administration interface allows you to delete old counters from the counter database, however the counter must also be deleted from your code or it will be recreated by the mgiCounter tag.

To delete a counter from the database, click the "Delete" button beside the counter name.

[Return to the Displaying Page Counters Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Displaying Graphic Counters with Custom Fonts

Introduction

Counters tabulate the number of visitor's to a page. However, counters cannot determine unique visitors to a page (i.e, one visitor may visit the page multiple times).

Text counter values appear as text numbers. Graphic counter values appear as number graphics in a specified font. MGI has six internal fonts that may suits your needs, however if they do not you can use custom number graphics for the graphic counter values. If you don't want to create your own graphics, there are tons of web sites that offer free sets of counter graphics in every style that you can imagine.

In this example, a graphic counter with a custom font is added to a web page.

MGI Tags

- mgiCounter

Steps

1. Open a page in a text editor .
2. Insert the mgiCounter tag.
3. Save the page.
4. Prepare the counter graphics.
5. FTP the page and custom graphics to the web server running MGI.
6. View the page in a web browser.

Step 1: Open a page in a text editor.

Open a web page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiCounter tag.

Enter the mgiCounter tag, name parameter, type parameter and fontFolderLocation parameter where you want the counter value to appear. In the name parameter, enter the unique counter name. Counters are stored in an internal MGI database and values from multiple counters with the same name are added to the same database record. In the type parameter, enter "Graphic". In the fontFolderLocation parameter, enter the relative path to the folder that contains the custom graphics to display. In this example, the counter graphics are stored in a folder named "CounterImages".

```
<mgiCounter name="index" type="Graphic"
fontFolderLocation="CounterImages">
```

Step 3: Save the page.

Save the changes you have made to the page.

Step 4: Prepare the counter graphics.

The custom counter graphics must be saved in GIF format and named with the following syntax:

- 0.gif
- 1.gif
- 2.gif
- 3.gif
- 4.gif
- 5.gif
- 6.gif
- 7.gif
- 8.gif
- 9.gif

Step 5: FTP the page and custom graphics to the web server running MGI.

Upload the page and custom graphics from your local computer to the web server using an FTP program.

Step 6: View the page in a web browser.

View the page in a web browser. The counter value displays in your custom font.

[Return to the Displaying Page Counters Menu]

Counting Unique Visitors

Introduction

By default, a counter increments for every access to a page. If the current counter value is 547, that number could represent 547 unique visitors, one person accessing the page 547 times or some combination of the two. To increase your likelihood of counting unique visitors, set the "ignoreClientIPAddress" parameter in your mgiCounter tag to "No". The mgiCounter tag will not ignore the current visitor's IP address and will compare the current visitor's IP address to the previous visitor's IP address. If the two IP addresses match, the counter will not increment. If the two IP addresses differ, the counter will increment. Notice that the "igonoreClientIPAddress" parameter of mgiCounter does not prevent the count of repeat visitors that do not occur sequentially.

In this brief example, the ignoreClientIPAddress parameter is used on a software download page.

MGI Tags

- mgiCounter

Steps

1. Open the software download page in a text editor.
2. Insert the mgiCounter tag, name parameter and ignoreClientIPAddress parameter.
3. Save the software download page.
4. FTP the software download page to the web server running MGI.
5. View the software download page in a web browser.

Step 1: Open the software download page in a text editor.

Open the software download page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiCounter tag, name parameter and ignoreClientIPAddress parameter.

Insert your cursor in the HTML of the software download page page where you want the counter to display. Enter the mgiCounter tag, name parameter and ignoreClientIPAddress parameter. In the name parameter, enter a unique counter name. In the ignoreClientIPAddress parameter, enter "No".

```
<mgiCounter name="download" ignoreClientIPAddress="No">
```

Step 3: Save the software download page.

Save the changes you have made to the software download page.

Step 4: FTP the software download page to the web server running MGI.

Upload the software download page from your local computer to the web server using an FTP program.

Step 5: View the software download page in a web browser.

View the software download page in a web browser. The counter increments and displays on the page. If you quickly reload the software download page, the counter will not increment and will display the same value.

[Return to the Displaying Page Counters Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Frequently Asked Counter Questions

Q1. Why doesn't my counter display?

A1. Your page is not being processed by MGI. Make sure that you are viewing the page in a browser after it has been processed by a server running MGI. Also make sure that the extension of your page name (e.g., .mgi, .html, .htm, etc.) is set to be processed by MGI.

A2. The "display" parameter is included in the mgiCounter tag and is set to "no". The "display" parameter is used to hide a counter value. If the "display" parameter is included and is set to "no", the counter will not display even when it is processed by MGI and even though it is counting.

Q2. Why does the mgiCounter tag show up on my page in a browser?

A1. The mgiCounter tag was entered in the graphic view of a graphic HTML editor. MGI tags should always be entered using a text editor or the HTML view of a graphic editor. If MGI tags are entered in the graphic view of a graphic editor, the opening and closing tag characters and the quotations marks are often converted from text to HTML character entities (e.g. "&quot;" represents a quotation mark in HTML). MGI cannot be parsed correctly if the tag components are converted to such character entities.

Q3. Why doesn't my counter increment?

A1. You are viewing a cached copy of the page. Most browsers cache copies of the pages that you view by default. That is, browsers keep a copy of the page on your local machine so that the page loads quicker the next time you view it. However, since the page that you are viewing on subsequent "visits" is from your local machine rather than a new copy served from MGI, you will see the same counter value each time you view the page. To avoid this problem you can change the cache settings on your browser. To view the newest copy of any page, you can often force reload the page by holding down the Apple key (Mac) or Control Key (PC) while you click the "Reload" or "Refresh" button on your browser.

[Return to the Displaying Page Counters Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

Processing Forms Submissions

In the **Beginner Tutorial** section, learn how to send an email that contains information from a form submission. In the **Advanced Tutorial** section, learn advanced form processing techniques including verifying information in form fields, sending email attachments and sending encrypted emails. In the **Reference** section, view a complete technical reference for each tag used for form submissions. In the **Downloads** section, download example code for full beginner and advanced tutorials. In the **FAQs** section, read answers to frequently asked questions.

Beginner Form Submission Tutorials

- B1. Processing Forms to Email

Advanced Form Submission Tutorials

- A1. Verifying Form Information
- A2. Encrypting Emails
- A3. Sending Email Attachments

Form Submission MGI Tag Reference

- mgiPGP
- mgiSendMail
- mgiPostArgument
- mgiValidateData

Form Submission Downloads

- B1. Download a form to email example (B1FormEmail.zip - 2 KB)
- A1. Download a form verification example (A1Verify.zip - 3 KB)

Frequently Asked Form Submission Questions

- Q1. I never received a form submission. What happened to it?

- Q2. Can I attach a file from my local computer to an email?

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Processing Forms to Email

Introduction

Forms allow your web site visitors to interact with you. To receive the visitor's information, use the mgiSendMail tag to translate the form submission into a formatted email.

In this example, a store is collecting comments about their products.

MGI Tags

- mgiSendMail
- mgiPostArgument

Steps

1. Create an information request form.
2. Create a form processing page and open it in a text editor.
3. Insert the mgiSendMail tag.
4. Save the form processing page.
5. FTP the information request form and form processing page to the web server running MGI.
6. View the information request form in a web browser and submit the request.

Step 1: Create an information request form.

Create an information request form with form elements. In this example, the information request form consists of text fields for the visitor's Name, Company, Address, City, State, Zip, Country, Phone Number, and Email Address, and a text box for Comments. Name each form element uniquely. Enclose all form elements with HTML <FORM> tags and post the form to the form processing page (infoprocess.mgi).

This is an example information request form.

```
<form action="infoprocess.mgi" method="post">

<h2>Information Request Form</h2>

<P><CENTER>

<TABLE BORDER="0" CELLSPACING="0" CELLPADDING="3">
```

Step 2: Create a form processing page and open it in a text editor.

Create a page named "infoprocess.mgi" to format the form information and send it to the designated address via email. Open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Insert the mgiSendMail tag.

Enter the beginning mgiSendMail tag, to parameter, from parameter, mailServer parameter, subject parameter, and ending mgiSendMail tag. In the "to" parameter enter the recipient's email address. In the "from" parameter, enter the sender's email address. In the "mailServer" parameter, enter the address of the outgoing SMTP server. In the "subject" parameter, enter the subject of the email.

In the body of the mgiSendMail tags, enter text labels and an mgiPostArgument tag for each form element. In the name parameter of mgiFieldContent, enter the name of the form element to display.

This is an example form processing page.

```
<h2><center>Information Request Submission
</center></h2>

<p><center>Thank you for your request.
A representative will contact you shortly.
</center></p>

<mgiSendMail to="info@domain.com"
from="webmaster@domain.com"
mailserver="mail.domain.com"
subject="Info Request">
Information Request
*******************

   Name: <mgiPostArgument name="name">
Company: <mgiPostArgument name="company">
Address: <mgiPostArgument name="address">
   City: <mgiPostArgument name="city">
  State: <mgiPostArgument name="state">
    Zip: <mgiPostArgument name="zip">
Country: <mgiPostArgument name="country">
  Phone: <mgiPostArgument name="phone">
  Email: <mgiPostArgument name="email">

Comments:
```

```
<mgiPostArgument name="comments">
</mgiSendMail>
```

Step 4: Save the form processing page.

Save the changes you have made to the form processing page.

Step 5: FTP the information request form and form processing page to the web server running MGI.

Upload the information request form and form processing page from your local computer to the web server using an FTP program.

Step 6: View the information request form in a web browser and submit the request.

View the information request form in a web browser. Complete and submit the information request form. An email with the following format is sent to the specified recipient

```
Information Request
*******************

    Name: Allan Para
 Company: Para, Inc.
 Address: 1234 Main Ave.
    City: Denver
   State: CO
     Zip: 33489
 Country: US
   Phone: 848-293-4712
   Email: allan@parainc.com

Comments:
I am interested in your product, but does it
come in different sizes?
```

[Return to the Forms Submission Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

Verifying Form Information

Introduction

Validating form information saves you time in processing a form requested. If you require specific information or information in a specifc format, use the mgiValidateData tag to check the for the existence of the data or the format of the data and return an error message or redirect visitors to a separate error page when data is invalid.

In this example, a doctor's office is accepting online registrations and validates specific data including required fields, email addresses, and appointment dates.

MGI Tags

- mgiSendMail
- mgiPostArgument
- mgiValidateData

Steps

1. Create a medical registration form.
2. Create a registration processing page and open it in a text editor.
3. Validate the form information and send a formatted email.
4. Save the registration processing page.
5. FTP the registration form and processing page to the web server running MGI.
6. View the registration form in a web browser and request an appointment.

Step 1: Create a medical registration form.

Create a medical registration form with form elements. In this example, the form has six sections including Contact Information, Medical History, Insurance Information, Emergency Contact Information, Reason for Visit and Appointment Date.

Name each form element uniquely. Enclose all form elements with HTML <FORM> tags and post the form to the registration processing page (register.mgi).

This is code for an example medical registration form.

```
<FORM ACTION="register.mgi" METHOD="POST">
<H2><CENTER>Medical Appointment Registration</CENTER></H2>
```

```
<P>Please complete all information as adequately as possible.
This will save you time during your initial visit to our clinic.
Required fields are marked in <B>bold</B>.</P>
<P><CENTER><TABLE BORDER="0" CELLSPACING="2" CELLPADDING="3"
WIDTH="500">
  <TR>
    <TD COLSPAN="2" BGCOLOR="#66cccc"><B><FONT SIZE="+1">Patient
      Information</FONT></B></TD>
  </TR>
  <TR>
    <TD WIDTH="50%" BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Name:
    </FONT></B></TD>
    <TD WIDTH="50%" BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Address,
      City, State, Zip Code:</FONT></B></TD>
  </TR>
  <TR>
    <TD WIDTH="50%" VALIGN="TOP"><INPUT NAME="Patient Name"
      TYPE="text" SIZE="30"></TD>
    <TD WIDTH="50%" VALIGN="TOP"><INPUT NAME="Patient Address"
      TYPE="text" SIZE="30"><BR>
      <INPUT NAME="Patient City" TYPE="text" SIZE="15">
      <INPUT NAME="Patient State" TYPE="text" SIZE="2">
      <INPUT NAME="Patient Zip" TYPE="text" SIZE="5"></TD>
  </TR>
  <TR>
    <TD WIDTH="50%" BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Phone
      Number:</FONT></B></TD>
    <TD WIDTH="50%" BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Email
      Address:</FONT></B></TD>
  </TR>
  <TR>
    <TD WIDTH="50%"><INPUT NAME="Patient Phone" TYPE="text"
      SIZE="30"></TD>
    <TD WIDTH="50%"><INPUT NAME="Patient Email" TYPE="text"
      SIZE="30"></TD>
  </TR>
  <TR>
    <TD COLSPAN="2" BGCOLOR="#66cccc"><B><FONT SIZE="+1">Medical
      History</FONT></B></TD>
  </TR>
  <TR>
    <TD WIDTH="50%" BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Please
      list all current medications and dosages:</FONT></B></TD>
    <TD WIDTH="50%" BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Please
      describe any previous medical conditions including operations:
      </FONT></B></TD>
  </TR>
  <TR>
    <TD WIDTH="50%"><TEXTAREA NAME="Medications" ROWS="5"
      COLS="27"></TEXTAREA></TD>
    <TD WIDTH="50%">&nbsp;<TEXTAREA NAME="Medical Conditions"
      ROWS="5" COLS="27"></TEXTAREA></TD>
  </TR>
  <TR>
    <TD COLSPAN="2" BGCOLOR="#66cccc"><B><FONT SIZE="+1">Insurance
    Information</FONT></B></TD>
</TR>
<TR>
  <TD WIDTH="50%" BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Insurance
    Provider:</FONT></B></TD>
  <TD WIDTH="50%" BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Group
    Number:</FONT></B></TD>
</TR>
<TR>
  <TD WIDTH="50%"><INPUT NAME="Insurance Provider"
    TYPE="text" SIZE="30"></TD>
  <TD WIDTH="50%"><INPUT NAME="Group Number" TYPE="text"
    SIZE="30"></TD>
</TR>
<TR>
  <TD WIDTH="50%" BGCOLOR="#eeeeee"><FONT SIZE="-1">Claims
    Telephone Number:</FONT></TD>
  <TD WIDTH="50%">&nbsp;</TD>
</TR>
<TR>
  <TD WIDTH="50%"><INPUT NAME="Claims Phone" TYPE="text"
    SIZE="30"></TD>
  <TD WIDTH="50%">&nbsp;</TD>
</TR>
<TR>
  <TD COLSPAN="2" BGCOLOR="#66cccc"><B><FONT SIZE="+1">Emergency
    Contact Information</FONT></B></TD>
</TR>
<TR>
  <TD WIDTH="50%" BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Name:</FONT>
  </B></TD>
  <TD WIDTH="50%" BGCOLOR="#eeeeee"><FONT SIZE="-1">Address,
    City, State, Zip Code:</FONT></TD>
</TR>
<TR>
  <TD VALIGN="TOP"><INPUT NAME="Contact Name" TYPE="text"
    SIZE="30"></TD>
  <TD VALIGN="TOP"><INPUT NAME="Contact Address" TYPE="text"
    SIZE="30"><BR>
    <INPUT NAME="Contact City" TYPE="text" SIZE="15">
    <INPUT NAME="Contact State" TYPE="text" SIZE="2">
    <INPUT NAME="Contact Zip" TYPE="text" SIZE="5"></TD>
</TR>
<TR>
  <TD BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Phone Number:
  </FONT></B></TD>
  <TD BGCOLOR="#eeeeee"><FONT SIZE="-1">Alternate Phone
    Number:</FONT></TD>
</TR>
<TR>
  <TD><INPUT NAME="Contact Phone" TYPE="text" SIZE="30"></TD>
  <TD><INPUT NAME="Contact Alt Phone" TYPE="text"
    SIZE="30"></TD>
</TR>
<TR>
```

```
  <TD COLSPAN="2" BGCOLOR="#66cccc"><B><FONT SIZE="+1">Reason for
    Visit</FONT></B></TD>
</TR>
<TR>
  <TD COLSPAN="2" BGCOLOR="#eeeeee"><FONT SIZE="-1">Please describe
    the reason for your current visit to our clinic:</FONT></TD>
</TR>
<TR>
  <TD COLSPAN="2">
<TEXTAREA NAME="Reason for Visit" ROWS="5" COLS="60">
  </TEXTAREA></TD>
</TR>
<TR>
  <TD COLSPAN="2" BGCOLOR="#66cccc"><B><FONT SIZE="+1">Appointment
    Date</FONT></B></TD>
</TR>
<TR>
  <TD COLSPAN="2" BGCOLOR="#eeeeee"><B><FONT SIZE="-1">Select your
    preferred appointment date. The date you choose cannot be more
    than 30 days after the current date (<mgiDate>).</FONT></B></TD>
</TR>
<TR>
  <TD COLSPAN="2">
    <P><CENTER><SELECT NAME="Month">
    <OPTION SELECTED VALUE="">Month
    <OPTION VALUE="01">January
    <OPTION VALUE="02">February
    <OPTION VALUE="03">March
    <OPTION VALUE="04">April
    <OPTION VALUE="05">May
    <OPTION VALUE="06">June
    <OPTION VALUE="07">July
    <OPTION VALUE="08">August
    <OPTION VALUE="09">September
    <OPTION VALUE="10">October
    <OPTION VALUE="11">November
    <OPTION VALUE="12">December
    </SELECT> <SELECT NAME="Day">
    <OPTION SELECTED VALUE="">Day
    <OPTION>01
    <OPTION>02
    <OPTION>03
    <OPTION>04
    <OPTION>05
    <OPTION>06
    <OPTION>07
    <OPTION>08
    <OPTION>09
    <OPTION>10
    <OPTION>11
    <OPTION>12
    <OPTION>13
    <OPTION>14
    <OPTION>15
    <OPTION>16
```

```
        <OPTION>17
        <OPTION>18
        <OPTION>19
        <OPTION>20
        <OPTION>21
        <OPTION>22
        <OPTION>23
        <OPTION>24
        <OPTION>25
        <OPTION>26
        <OPTION>27
        <OPTION>28
        <OPTION>29
        <OPTION>30
        <OPTION>31
        </SELECT> <SELECT NAME="Year">
        <OPTION SELECTED VALUE="">Year
        <OPTION>2001
        <OPTION>2002
        </SELECT></CENTER></TD>
    </TR>
  </TABLE></CENTER></P>
  <P><CENTER><input type="submit" value="Submit Registration"></CENTER>
```

Step 2: Create a registration processing page and open it in a text editor.

Create a page named "register.mgi" to format the registration information and send it to the designated address via email. Open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Validate the form information and send a formatted email.

Use the mgiValidateData tag to verify the existence of required form information, verify the format of the required email address, and verify the upper range of the selected appointment date. Since the default behavior of the mgiValidateData tag is to display the data when it is verified, the formatted body of the email is built in a page variable. A conditional comparison is then performed with the contents of the page variable to determine if errors are present. If errors are present, an error message is displayed. If errors are not present, a receipt message is displayed and the formatted email is sent.

To verify the existence of data, enter the mgiValidateData tag and data parameter. Embed an mgiPostArgument tag and name parameter in the data parameter. The post argument name is the name of the form information to validate.

To verify the format of a required email address, enter the mgiValidateData tag, data parameter and format parameter. Embed an mgiPostArgument tag and name parameter in the data parameter. The post argument name is the name of the form field that contains the email address to validate. In the format parameter, enter "emailAddress".

To validate dates you must first format the date in the default mm-dd-yyyy format or with another specified format. Enter the mgiValidateData tag, data parameter, type parameter, and upperSpan parameter. Embed an mgiPostArgument tag and name parameter in the data parameter. The post argument name is the name of the page variable that contains the properly formatted date information. In the type parameter, enter "Date". In the upperSpan parameter, enter the maximum number of days (from the current datte) that the date value is valid (e.g., "30").

For information that is not required, enter an mgiPostArgument tag and name parameter to display the data from the specified form field.

Use an mgiIf tag to check for "invalid data" errors produced by the mgiValidateData tag. If errors exist, display an error message. If errors do not exist, send the formatted email using the mgiSendMail tag. In the "to" parameter of mgiSendMail, enter the recipient's email address. In the "from" parameter, enter the sender's email address. In this example, the patient's email address is embedded as the sender of the email. In the "subject" parameter, enter the subject of the email. In the "mailServer" parameter, enter the address of the outgoing SMTP server. The formatted email is displayed in the body of the mgiSendMail tags with an mgiGet tag.

This is an example registration processing page.

```
<H2><CENTER>Medical Appointment Submission</CENTER></H2>

<mgiSet name="FormattedDate">
<mgiPostArgument name="Month">-<mgiPostArgument name="Day">-
<mgiPostArgument name="Year">
</mgiSet>

<mgiSet name="Validation">
Patient Information
-------------------
     Name: <mgiValidateData
data={mgiPostArgument name="Patient Name"}>
  Address: <mgiValidateData
data={mgiPostArgument name="Patient Address"}>
           <mgiValidateData
data={mgiPostArgument name="Patient City"}>
           <mgiValidateData
data={mgiPostArgument name="Patient State"}>
           <mgiValidateData
data={mgiPostArgument name="Patient Zip"}>
    Phone: <mgiValidateData
data={mgiPostArgument name="Patient Phone"}>
    Email: <mgiValidateData
data={mgiPostArgument name="Patient Email"}
           format="emailAddress">
```

```
Medical History
---------------
Medications:
<mgiValidateData data={mgiPostArgument name="Medications"}>

Conditions:
<mgiValidateData data={mgiPostArgument name="Medical Conditions"}>

Insurance Information
---------------------
  Provider: <mgiValidateData
data={mgiPostArgument name="Insurance Provider"}>
     Group: <mgiValidateData
data={mgiPostArgument name="Group Number"}>
    Claims: <mgiPostArgument name="Claims Phone">

Emergency Contact
-----------------
      Name: <mgiValidateData
data={mgiPostArgument name="Contact Name"}>
   Address: <mgiPostArgument name="Contact Address">
            <mgiPostArgument name="Contact City">
            <mgiPostArgument name="Contact State">
            <mgiPostArgument name="Contact Zip">
     Phone: <mgiValidateData
data={mgiPostArgument name="Contact Phone"}>
 Alt Phone: <mgiPostArgument name="Contact Alt Phone">


Reason for Visit
----------------
<mgiPostArgument name="Reason for Visit">

Requested Appointment Date: <mgiValidateData type="date"
data={mgiGet name="FormattedDate"} upperSpan="30">
</mgiSet>

<mgiIf lhs={mgiGet name="Validation"} relationship="contains"
rhs="invalid data">

<p><b>An error has occurred processing your appointment request.</b>
Please use the "Back" button on your browser and verify the
information you have entered.  Make sure all required fields
have been completed, all email addresses are formatted
properly and that your selected appointment date is not more
than 30 days from today's date.

<mgiElse>

<P>Your registration and appointment information has been received.
Please bring copies of any previous medical records during your
visit. Your appointment time will be verified by phone.
```

```
<mgiSendMail to="registration@domain.com"
from={mgiPostArgument name="Patient Email"}
subject="Medical Registration" mailserver="mail.domain.com">
<mgiGet name="Validation">
</mgiSendMail>

</mgiIf>
```

Step 4: Save the registration processing page.

Save the changes you have made to the registration processing page.

Step 5: FTP the registration form and processing page to the web server running MGI.

Upload the registration form and processing page from your local computer to the web server using an FTP program.

Step 6: View the registration form in a web browser and request an appointment.

View the registration form in a web browser. Complete and submit the form. An email with the following format is sent to the specified recipient.

```
Patient Information
-------------------
     Name: Charles Washington
  Address: 123 Main Ave
           Durham NC 21546
    Phone: 123-456-7894
    Email: charles@dbulls.net

Medical History
---------------
Medications:
Allegra - 50 MG

Conditions:
Allergies
Diabetes

Insurance Information
---------------------
  Provider: Blue Cross/Blue Shield
     Group: F8993H232121
    Claims: 704-256-8945
```

```
Emergency Contact
-----------------
      Name: Helen Washington
   Address: 123 Main Ave
            Durham NC 21546
     Phone: 123-456-7894
 Alt Phone: 856-456-7845


Reason for Visit
----------------
Yearly physical and checkup.

Requested Appointment Date: 06-03-2001
```

[Return to the Forms Submission Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Encrypting Emails

Introduction

Encryption protects the contents of an email while the email is in transit from the web server to the email recipient. MGI performs encryption with a "passphase". MGI encrypts the email with the "passphrase" (from the passPhrase parameter or from a passPhrase located in a text file) as a key for the encryption. The recipient decrypts the email with the same "passphrase" using a PGP utility.

Free PGP utilities for personal use can be found at http://web.mit.edu/network/pgp.html. Commercial PGP utilities can be found at http://www.pgp.com.

This example illustrates the encryption of a payment email.

MGI Tags

- mgiPGP
- mgiSendMail

Steps

1. Create a payment form and payment processing page.
2. Open the payment processing page in a text editor.
3. Insert the mgiPGP tag.
4. Save the payment processing page.
5. FTP the payment form and payment processing page to the web server running MGI.
6. Install a PGP utility on the recipient's computer.
7. Complete and submit the payment form.

Step 1: Create a payment form and payment processing page.

Create a payment form and payment processing page. On the payment processing page, enter an mgiSendMail tag to format and send the payment email. For instructions regarding the construction of a form and email to process the form, please review the Processing Forms to Email tutorial.

Step 2: Open the payment processing page in a text editor.

Open the payment processing page in a text editing program that allows you to modify the HTML and code of the page.

Step 3: Insert the mgiPGP tag.

The mgiPGP tag encrypts the contents of the email, therefore the beginning and ending mgiPGP tags should enclose the content of the email. Insert a beginning mgiPGP tag and passPhrase parameter after the beginning mgiSendMail tag. In the passPhrase parameter, enter the case-sensitve passphrase to encrypt the email. Insert an ending mgiPGP tag before the ending mgiSendMail tag.

Note: if you choose to use a passphrase from a text file (via the fileLocation parameter), protect that text file with a WebSTAR realm. The server administrator can set the realm for a specific file name.

The following code is an example email from a payment processing form.

```
<mgiSendMail to="accounting@domain.com"
from="webmaster@domain.com"
subject="Payment" mailserver="mail.domain.com">

<mgiPGP passPhrase="encryptWithPhrase">
     Acct: <mgiPostArgument name="Account">
     Type: <mgiPostArgument name="CreditCardType">
   Number: <mgiPostArgument name="CreditCardNumber">
Exp Month: <mgiPostArgument name="ExpireMonth">
 Exp Year: <mgiPostArgument name="ExpireYear">
</mgiPGP>

</mgiSendMail>
```

Step 4: Save the payment processing page.

Save the changes you have made to the payment processing page.

Step 5: FTP the payment form and payment processing page to the web server running MGI.

Upload the payment form and payment processing page from your local computer to the web server using an FTP program.

Step 6: Install a PGP utility on the recipient's computer.

In order to read an ecrypted email the email recipient must install a PGP utility on their computer to decrypt the email.

Step 7: Complete and submit the payment form.

Access the payment form in a browser. Complete and submit the form. Upon submission, an encrypted email is sent and the payment processing page is displayed. When the email is received, decrypt it with the passphrase specified in the mgiPGP tag.

[Return to the Forms Submission Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Sending Email Attachments

Introduction

The mgiSendMail tag can attach files from the server to an email. The mgiSendMail tag cannot directly attach files from your local computer to an email, however you could upload the file as part of a form submission via MGI's HTTP upload function and dynamically embed the file name as the attachment.

You may attach multiple files to an email by including multiple "attachmentData" or "attachmentFileLocation" parameters in the mgiSendMail tag.

This example illustrates attaching a product distributor list to a customer request.

MGI Tags

- mgiSendMail

Steps

1. Create a request form and processing page.
2. Open the processing page in a text editor.
3. Enter the attachmentFileLocation parameter in the mgiSendMail tag.
4. Save the processing page.
5. FTP the request form and processing page to the web server running MGI.
6. Complete and submit the request form.

Step 1: Create a request form and processing page.

Create a customer request form and processing page. On the processing page, enter an mgiSendMail tag to format and send the request email. For instructions regarding the construction of a form and email to process the form, please review the Processing Forms to Email tutorial.

Step 2: Open the processing page in a text editor.

Open the processing page in a text editing program that allows you to modify the HTML and code of the page.

Step 3: Enter the attachmentFileLocation parameter in the mgiSendMail tag.

To attach a file to an email, enter the attachmentFileLocation parameter in the mgiSendMail tag. In the attachmentFileLocation parameter, enter the relative path to the file to attach. Multiple files may be attached by including multiple attachmentFileLocation parameters in one mgiSendMail tag. All attachments are Base64 encoded.

```
<mgiSendMail to={mgiPostArgument name="email"}
from="sales@domain.com"
subject="Distributor Request"
mailserver="mail.domain.com"
attachmentFileLocation="Docs/distributor.doc">
Thank you for requesting our distributor list.
The distributor list is attached to this email.
If you have any questions, please contact us.
</mgiSendMail>
```

Step 4: Save the processing page.

Save the changes you have made to the processing page.

Step 5: FTP the request form and processing page to the web server running MGI.

Upload the request form and processing page from your local computer to the web server using an FTP program.

Step 6: Complete and submit the request form.

Access the request form in a browser. Complete and submit the form. Upon submission, an email with the "distributor.doc" attachment is sent.

[Return to the Forms Submission Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Frequently Asked Forms Processing Questions

Q1. I never received a form submission. What happened to it?

A1. The email failed and is in the Failed Mail database. If an email cannot be delivered to the specified outgoing SMTP server for any reason (including incorrect email addresses, relay restrictions, or a server failure), all of the email information is stored in the Failed Mail database. The Failed Mail database can be accessed with the web-based Admin mode of the mgiSendMail tag and you may choose to edit and resend the email or delete the email.

One of the most common reasons for a failed email is relay restrictions. Due to the abudance of unsolicited emails ("spam"), email servers are protected with filters that allow and deny email based on the sender and recipient addresses. On many servers, either the sender or recipient email address must be hosted by that email server. Emails that are not sent to an address hosted by the server or received from an address hosted by the server are assumed to be "spam" and are bounced. Check with your email server administrator to make sure your emails are hosted by the server and are properly configured.

A2. The email was bounced or lost by the recipient server. Email isn't perfect, as you undoubtedly know. Many problems can occur with an email after MGI has successfully delivered the email to the email server including "bounces". If an email is returned by the recipient server (because the user is unknown, for example), the email will be returned to the sender's address and will not be returned to or stored by MGI.

Q2. Can I attach a file from my local computer to an email?

A1. Not directly. Attachments that are sent with mgiSendMail must reside on the web server, therefore you cannot directly attach a file from your local computer to an email. However, you can upload a file via HTTP file upload to your region and attach that file to an email. The HTTP file upload could occur along with the regular form submission.

[Return to the Forms Submission Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

HTTP File Upload

In the **Beginner Tutorial** section, learn how to upload files via HTTP file upload. In the **Reference** section, view a complete technical reference for each tag used for file upload. In the **Downloads** section, download example code for each beginner tutorial. In the **FAQs** section, read answers to frequently asked questions.

Beginner HTTP File Upload Tutorials

- B1. HTTP File Upload

HTTP File Upload MGI Tag Reference

- None - HTTP File Upload is configured in the Domain Admin.

HTTP File Upload Downloads

- B1. Download example file upload pages (B1FileUpload.zip - 28 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

HTTP File Upload

Introduction

Files can be transferred from your local computer to a web server via serveral means. One is standard FTP (File Transfer Protocol). FTP requires an FTP application on your local computer and an FTP server on the machine you are transferring files to. To accomplish the same task you can use HTTP file upload. Rather than transferring files from your local machine to a server using an FTP application, you transfer files using your browser and MGI places those files in a designated location on the web server.

HTTP file uploads are preferable for clients who do not know how to use FTP, anonymous file uploads, group projects, etc. Using MGI's HTTP file upload, you may specify which files to accept and deny all others to keep your web site secure.

Access to HTTP file upload and a specified upload location must be configured in the Domain Admin on a region basis.

In this example, configure a region to upload files to a folder on the server named "images".

MGI Tags

- None - HTTP File Upload is configured in the Domain Admin.

Steps

1. Configure the region for file upload.
2. Create a file upload form.
3. Create an upload results page and open it in a text editor.
4. Insert the mgiPostArgument tag.
5. Save the upload results page.
6. FTP the file upload form and upload results page to the web server running MGI.
7. View the file upload form.

Step 1: Configure the region for file upload.

Access the domain admin page for the domain where files will be uploaded.

To access the domain admin page, access the "MGIDomainAdmin.xxx" page in a web browser from one of the domain's virtual host addresses. Replace the ".xxx" suffix with any suffix that is mapped to be processed by MGI2. For example, if you have the

".mgi" extension mapped to be processed by MGI2 you would access "MGIDomainAdmin.mgi" or if you have the ".html" extension mapped to be processed by MGI2, you would access "MGIDomainAdmin.html".

The domain admin page is password-protected. When a domain is created, the domain admin username and password are those of the current server admin and password.

Click the "Region Management" link. Select the region to configure for file upload. Highlight the forward slash "/" for the default region or the region name in the region list and click the "Edit" button.

In the file upload section, enter the maximum file size, path information, and regular expressions for authorized files.

By default, a file of any size can be uploaded. To limit the size of uploaded files, enter the maximum file size in kilobytes. Remember, a megabyte is 1024 kilobytes!

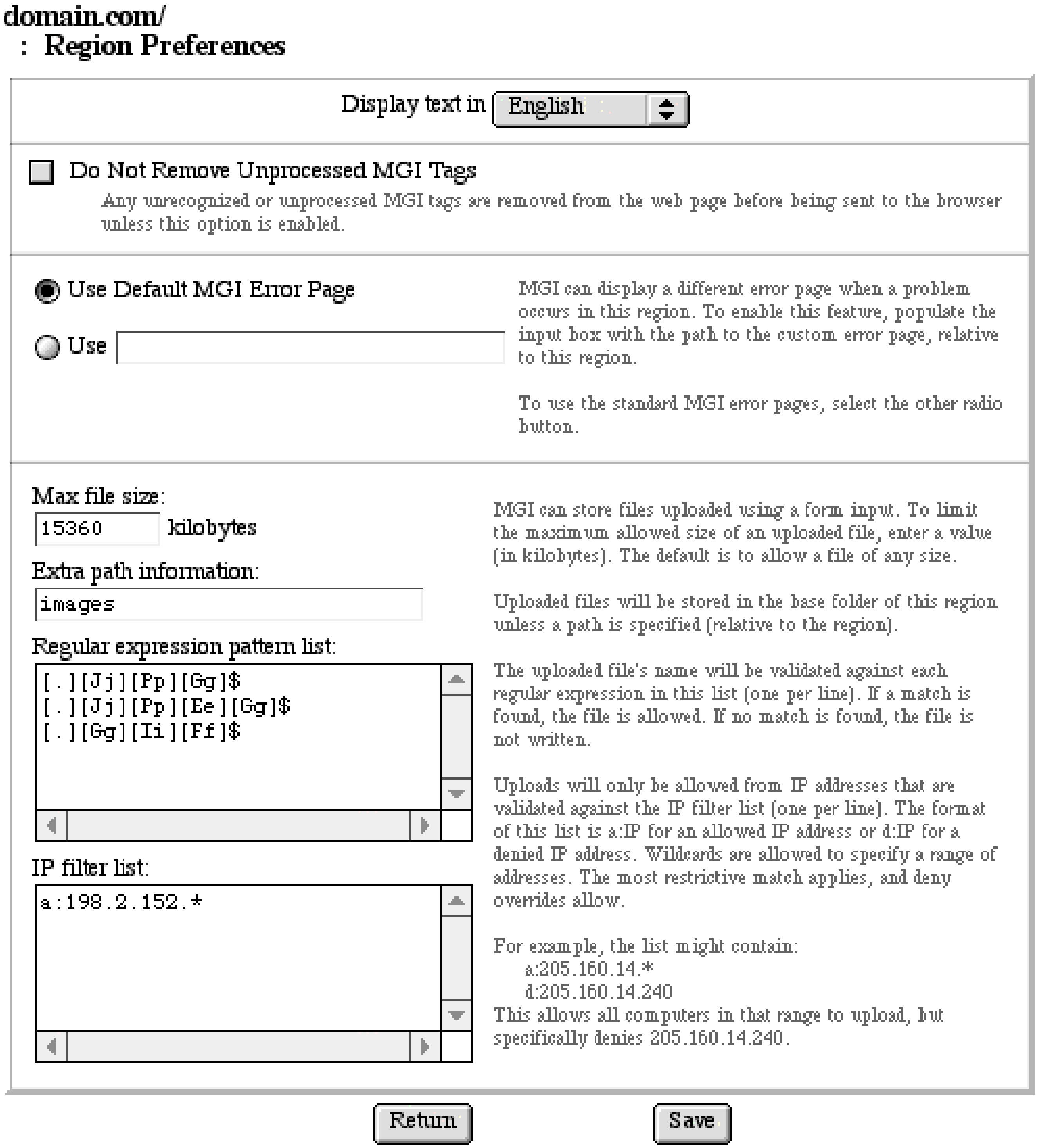
By default, files are uploaded to the root of the region. To specify a different folder, enter the path to the folder **relative to the selected region**. The folder path must be below the level of the selected region. If a folder does not exist, MGI will create it automatically. For this example, files will be saved in the "images" folder.

By default, all files that you try to upload are denied. Enter regular expressions that match authorized file names. The uploaded file name will be checked against the regular expressions until a match is found or until all expressions have been checked. Files that are not authorized for upload will be deleted and will not be written to the server. To allow any file to be uploaded, enter a single period (.) in the regular expression box. See the regular expression reference for more information and examples. For this example, only files ending with .jpg, .jpeg, or .gif will uploaded to the images folder.

By default, all IP addresses that attempt to upload files are denied access. To allow or deny an IP address, enter an IP filter in the "IP filter list". To allow an IP address, enter an "A" followed by a colon and the allowed IP address (e.g., "a:198.2.152.111"). To deny an IP address, enter a "D" followed by a colon and the denied IP address (e.g., "d:198.3.145.200"). You may use an asterisk (*) for a wildcard value in any position of the IP address. A wildcard value allows you to match of range of IP addresses. For example, to allow all IP addresses from the 198 Class C block you would enter "a:198.*.*.*" in the IP filter list. For multiple filters, enter one filter per line. When a file upload request is received, the most restrictive match for the IP address will be processed and if two filters match, a deny filter will override an accept filter. For example, if you wish to allow all IPs from the 198.3.145 address except the IP address

"198.3.145.201", then you would enter an allow and deny statement in the IP filter list ("a:198.3.145.*" and "d:198.3.145.201"). To allow all IP addresses, use wildcard values for all positions (i.e., "*.*.*.*").

Click the "Save" button to save the region preferences including the file upload preferences.



domain.com/
: Region Preferences

Display text in English

□ Do Not Remove Unprocessed MGI Tags
Any unrecognized or unprocessed MGI tags are removed from the web page before being sent to the browser unless this option is enabled.

● Use Default MGI Error Page

○ Use

MGI can display a different error page when a problem occurs in this region. To enable this feature, populate the input box with the path to the custom error page, relative to this region.

To use the standard MGI error pages, select the other radio button.

Max file size:
15360 kilobytes

Extra path information:
images

Regular expression pattern list:

```
[.][Jj][Pp][Gg]$
[.][Jj][Pp][Ee][Gg]$
[.][Gg][Ii][Ff]$
```

IP filter list:

```
a:198.2.152.*
```

MGI can store files uploaded using a form input. To limit the maximum allowed size of an uploaded file, enter a value (in kilobytes). The default is to allow a file of any size.

Uploaded files will be stored in the base folder of this region unless a path is specified (relative to the region).

The uploaded file's name will be validated against each regular expression in this list (one per line). If a match is found, the file is allowed. If no match is found, the file is not written.

Uploads will only be allowed from IP addresses that are validated against the IP filter list (one per line). The format of this list is a:IP for an allowed IP address or d:IP for a denied IP address. Wildcards are allowed to specify a range of addresses. The most restrictive match applies, and deny overrides allow.

For example, the list might contain:
a:205.160.14.*
d:205.160.14.240
This allows all computers in that range to upload, but specifically denies 205.160.14.240.

Step 2: Create a file upload form.

Create a form named "upload.mgi" to select and enter the file to upload. The file upload form field may be combined with any other type of form fields, however the form method must be post and the form encoding type must be "multipart/form-data".

Enter an HTML input tag, type parameter and name parameter. In the type parameter, enter "file". In the name parameter, enter the name of the post argument for the file upload. The post argument can be checked on the results page to determine if the file had to be renamed or to determine if there was an error uploading the file. A file input creates a text field with a "Browse" button.

Enter a submit button with the value "Submit Image".

Enclose all form elements with HTML <FORM> tags. The form method must be "Post" (posted to the "results.mgi" page in this example) and the form encoding type (ENCTYPE) must be "multipart/form-data".

The following example is an image upload page

```
<FORM ACTION="results.mgi" METHOD="POST"
ENCTYPE="multipart/form-data">

<CENTER>

<H3>Image Upload</H3>
<P>Enter the image file to upload or click
the "Browse" button to locate and select the
image file from your hard drive.</P>

<input type="file" name="imageFile">
<mgiButton value="Submit Image">

</CENTER>

</FORM>
```

The form in this example would display the following.

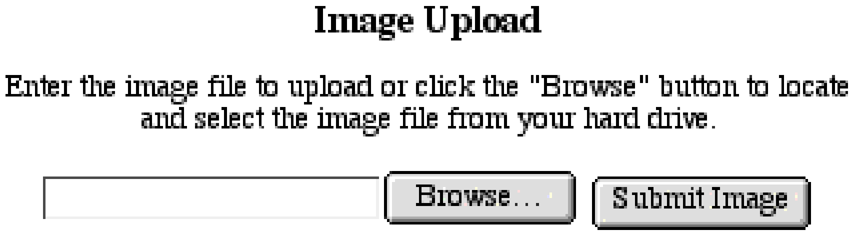


Image Upload

Enter the image file to upload or click the "Browse" button to locate and select the image file from your hard drive.

Step 3: Create an upload results page and open it in a text editor.

Create a page named "results.mgi" to display the results of the file upload. Open the upload results page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 4: Insert the mgiPostArgument tag.

When a file is successfully uploaded, the final file name can be displayed with the mgiPostArgument tag. If an upload file name is a duplicate of an existing file name, the new file name is appended with a number before the final period (i.e., before the file extension).

If an error occurs while uploading a file, the post argument will be empty (i.e., not exist). An error may occur if the file exceeds the maximum file size or does not match any of the specified regular expressions.

Check the post argument with a conditional and display the results if the post argument is not empty.

In this example, the post argument is named "imageFile".

```
<mgiIf lhs={mgiPostArgument name="imageFile"}
relationship="isNotEmpty">

<P>Your file has been uploaded.  The file name for your
upload is <mgiPostArgument name="imageFile">.

<mgiElse>

<p>An error has occurred during your file upload.  Please
make sure that the file is an image that ends with .jpg,
.jpeg or .gif.  The file size should not exceed 15 MB.
</mgiIf>
```

Step 5: Save the upload results page.

Save the changes you have made to the upload results page.

Step 6: FTP the file upload form and upload results page to the web server running MGI.

Upload the file upload form and upload results page from your local computer to the web server using an FTP program (at least until you implement the HTTP upload :).

Step 7: View the file upload form.

View the file upload form in a web browser. Enter the file name to upload or use the "browse" button to locate and select the file on your hard drive. Click the "Submit Image" button.

[Return to the HTTP File Upload Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Collecting Guestbook Entries

In the **Beginner Tutorial** section, learn how to collect, format and display visitors' comments and information in a guestbook. In the **Advanced Tutorial** section, learn advanced guestbook techniques including database-driven guestbooks. In the **Reference** section, view a complete technical reference for each tag used for guestbooks. In the **Downloads** section, download example code for full beginner and advanced tutorials. In the **FAQs** section, read answers to frequently asked questions.

Beginner Guestbook Tutorials

- B1. Displaying Guestbook Entries from a File

Advanced Guestbook Tutorials

- A1. Database-Driven Guestbook

Guestbook MGI Tag Reference

- mgiGuestbook
- mgiGuestbookDB

Guestbook Downloads

- B1. Download an example guestbook from a file (B1Guestbook.zip - 4 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Displaying Guestbook Entries from a File

Introduction

Guestbooks allow you to collect and format comments from web site visitors for display on your web site. The basic method of collecting guestbook entries is for visitors to complete and submit a form with their contact information and comments. When the form is processed, the guestbook entry is formatted and written to the specified location in a file (newer entries are written above older entries). Guestbook entries are then read by viewing the display file.

In this example, a store is collecting comments about their products.

MGI Tags

- mgiGuestbook

Steps

1. Create a form to collect guestbook information.
2. Create a guestbook processing page and open it in a text editor.
3. Insert the mgiGuestbook tag.
4. Save the guestbook processing page.
5. Create a page to display guestbook entries and open it in a text editor.
6. Insert the Guestbook Insertion Marker.
7. Save the guestbook display page.
8. FTP the guestbook form, guestbook processing page and guestbook display page to the web server running MGI.
9. View the guestbook form in a web browser and submit an entry.

Step 1: Create a form to collect guestbook information.

Create a guestbook form with form elements. In this example, the guestbook form consists of text fields for the visitor's Name, Location and Comments. Name each form element uniquely. Enclose all form elements with HTML <FORM> tags and post the form to the guestbook processing page (processgb.mgi).

This is an example guestbook form.

```
<form action="processgb.mgi" method="post">

<center>
```

```
<h2>Sign Our Guestbook</h2>

<p><table cellpadding="3" cellspacing="0">

<tr><td>Name:</td>
<td><input type="text" name="Name" size="30">
</td></tr>

<tr><td>Location (City, State):</td>
<td><input type="text" name="Location" size="30">
</td></tr>

<tr><td>Comments:</td>
<td><textarea name="Comments" rows="5" cols="30">
</textarea></td></tr>

<tr><td>&nbsp;</td>
<td><input type="submit" value="Post Comments">
</td></tr>

</table>
</center>

</form>
```

Step 2: Create a guestbook processing page and open it in a text editor.

Create a page named "processgb.mgi" to format each guestbook entry and write the entry to the guestbook display page. Open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Insert the mgiGuestbook tag.

Enter the beginning mgiGuestbook tag, fileLocation parameter, and ending mgiGuestbook tag. In the fileLocation parameter, enter the name of the guestbook display page (displaygb.mgi). In the body of the mgiGuestbook tags, enter text, HTML, and MGI tags to format each guestbook entry.

This is an example guestbook processing page.

```
<h2>Thank You</h2>

<p>Your comments are appreciated.
<a href="displaygb.mgi">Click here to
read all comments.</a>
```

```
<mgiGuestbook fileLocation="displaygb.mgi">
<table width="400" cellpadding="3" cellspacing="0">
<tr>
<td bgcolor="#99cc99"><b><mgiPostArgument name="Name"> of
<mgiPostArgument name="Location"></b></td>
</tr>
<tr>
<td><font color="#660099">
<mgiPostArgument name="Comments"></font></td>
</tr>
</table>
</mgiGuestbook>
```

Step 4: Save the guestbook processing page.

Save the changes you have made to the guestbook processing page.

Step 5: Create a page to display guestbook entries and open it in a text editor.

Create a named "displaygb.mgi" to display guestbook entries and open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 6: Insert the Guestbook Insertion Marker.

On the guestbook display page, insert the Guestbook Insertion Marker. The Guestbook Insertion Marker is written in the style of an HTML comment that designates where guestbook entries are displayed.

This is an example guestbook display page.

```
<center>

<h2>This is what our fine customers had to say:</h2>

<!-- Guestbook Insertion Marker -->

</center>
```

Step 7: Save the guestbook display page.

Save the changes you have made to the guestbook display page.

Step 8: FTP the guestbook form, guestbook processing page and guestbook display page to the web server running MGI.

Upload the guestbook form, guestbook processing page and guestbook display page from your local computer to the web server using an FTP program.

Step 9: View the guestbook form in a web browser and submit an entry.

View the guestbook form in a web browser. Complete the guestbook form and submit an entry. View the guestbook display page. Your entry will appear on the guestbook display page:

Sandra Clemmons of Maryland

Your products are the best. Keep up the good work. I am looking forward to the new line that is due out in March.

Pete Wilson of Atlanta, Georgia

Just wanted to say that my last order was a hit at my party. That is why today I am ordering some more. Also could you please tell me when will be the latest date I can order for xmas holidays... Have yourself a great day!

Each new guestbook entry is added to the display page at the location of the guestbook insertion marker. Newer guestbook entries appear before older guestbook entries.

[Return to the Guestbooks Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Database-Driven Guestbook

Introduction

Guestbooks allow you to collect and format comments from web site visitors for display on your web site. Database-driven guestbook entries are collected via a form, added to the guestbook database, then displayed from the guestbook database. You may format your guestbook entries with any text and HTML that you like.

The following is an example of a simple comment guestbook.

MGI Tags

- mgiGuestbookDB

Steps

1. Create a form to collect guestbook information.
2. Create a guestbook processing page and open it in a text editor.
3. Insert the mgiGuestbookDB tag in submit mode.
4. Save the guestbook processing page.
5. Create a guestbook display page and open it in a text editor.
6. Insert the mgiGuestbookDB tag in display mode.
7. FTP the form and guestbook pages to the web server running MGI.
8. View the guestbook form and submit an entry.

Step 1: Create a form to collect guestbook information.

Create a guestbook form with form elements. In this example, the guestbook form consists of text fields for the visitor's Name, Location and Comments. Name each form element uniquely. Enclose all form elements with HTML <FORM> tags and post the form to the guestbook page (guestbook.mgi).

This is an example guestbook form.

```
<form action="guestbook.mgi" method="post">

<center>
<h2>Sign Our Guestbook</h2>
```

```
<p><table cellpadding="3" cellspacing="0">

<tr><td>Name:</td>
<td><input type="text" name="Name" size="30">
</td></tr>

<tr><td>Location (City, State):</td>
<td><input type="text" name="Location" size="30">
</td></tr>

<tr><td>Comments:</td>
<td><textarea name="Comments" rows="5" cols="30">
</textarea></td></tr>

<tr><td>&nbsp;</td>
<td><input type="submit" value="Post Comments">
</td></tr>

</table>
</center>

</form>
```

Step 2: Create a guestbook processing page and open it in a text editor.

Create a page named "process.mgi" to submit the entry to the guestbook database. Open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Insert the mgiGuestbookDB tag in submit mode.

Enter the beginning mgiGuestbook tag, mode parameter, name parameter and ending mgiGuestbookDB tag. In the mode parameter, enter "submit". In the name parameter, enter the case-sensitive name of the guestbook.

In the body of the mgiGuestbookDB tags, enter text, HTML, and mgiPostArgument tags for form information.

In this example the visitor is immediately redirected to the guestbook display page via the mgiRedirect tag.

The following is an example guestbook processing page.

```
<mgiGuestbookDB name="Comments"
mode="submit" resultsPerPage="10"
page={mgiGet name="PageToDisplay"}
resultVariableName="GBResults">

<table width="400" cellpadding="3" cellspacing="0">
<tr>
<td bgcolor="#99cc99"><b><mgiPostArgument name="Name"> of
<mgiPostArgument name="Location"></b></td>
</tr>
<tr>
<td><font color="#660099">
<mgiPostArgument name="Comments"></font></td>
</tr>
</table>

</mgiGuestbookDB>

<mgiRedirect url="guestbook.mgi">
```

Step 4: Save the guestbook processing page.

Save the changes you have made to the guestbook processing page.

Step 5: Create a guestbook display page and open it in a text editor.

Create a page named "guestbook.mgi" to display entries from the guestbook database. Open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 6: Insert the mgiGuestbookDB tag in display mode.

Enter the beginning mgiGuestbook tag, mode parameter, name parameter, resultsPerPage parameter, page parameter, resultVariableName parameter, and ending mgiGuestbookDB tag.

In the mode parameter, enter "display". In the name parameter, enter the case-sensitive name of the guestbook. In the resultsPerPage parameter, enter the number of guestbook results to display per page. If the resultsPerPage parameter is not included, 25 results will display by default.

If it is possible for the number of guestbook entries to exceed the value listed in the resultsPerPage parameter, you must construct dynamic "Previous" and "Next" buttons to allow visitors to browse through different sets of guestbook entries. Below the mgiGuestbookDB tags, enter a hidden input for the value of the previous page of entries and the value of the next page of entries using the result variables (ResultVariableName_PrevPage and ResultVariableName_NextPage). Below the hidden inputs, display a "Previous" button if you are not currently displaying the first set of search results and display a "Next" button if there is more than one page of results available.

Above the mgiGuesbookDB tags, perform a conditional comparison to determine which set of entries should be displayed. If the visitor selected the "Previous" button during a search, then display the previous set of entries. If the visitor selected the "Next" button during a search, then display the next set of entries. Otherwise, display the first set of results. Set the page value in a variable and embed that variable in the page parameter of the mgiGuestbookDB tag.

In order for the "Previous" and "Next" buttons to function, enter HTML <FORM> tags. The action of the form tag should be the name of the guestbook page (guestbook.mgi) and the method should be "post".

The following is an example guestbook display page.

```
<mgiComment>
Determine which page of entries to display
</mgiComment>

<mgiSet name="PageToDisplay" defaultValue="1">
  <mgiInlineIf lhs={mgiPostArgument name="ResultsSet"}
  relationship="equals" rhs="Prev"
  then={mgiPostArgument name="PrevPage"}>

  <mgiInlineIf lhs={mgiPostArgument name="ResultsSet"}
  relationship="equals" rhs="Next"
  then={mgiPostArgument name="NextPage"}>
</mgiSet>

<mgiComment>
Submit and Display Entries
</mgiComment>

<mgiGuestbookDB name="Comments"
mode="display" resultsPerPage="10"
page={mgiGet name="PageToDisplay"}
resultVariableName="GBResults">
</mgiGuestbookDB>
```

```
<form action="guestbook.mgi" method="post">

<mgiComment>
Display dynamic Previous and Next Buttons
</mgiComment>

<input type="hidden" name="PrevPage"
value={mgiGet name="GBResults_PrevPage"}>

<input type="hidden" name="NextPage"
value={mgiGet name="GBResults_NextPage"}>

<mgiIf lhs={mgiGet name="GBResults_Page"}
relationship="greaterThan" rhs="1">
<input type="submit" name="ResultsSet" value="Prev">
</mgiIf>

<mgiIf lhs={mgiGet name="GBResults_NextPage"}
relationship="greaterThan" rhs="0">
<input type="submit" name="ResultsSet" value="Next">
</mgiIf>

</center>

</form>
```

Step 7: Save the guestbook display page.

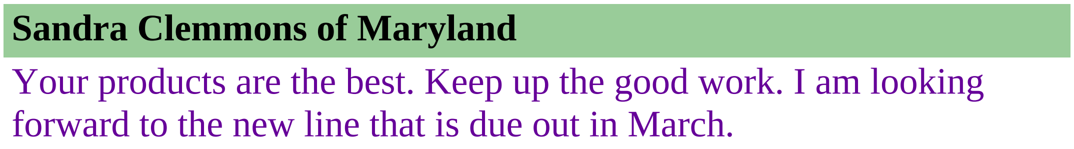
Save the changes you have made to the guestbook display page.

Step 8: FTP the form and guestbook pages to the web server running MGI.

Upload the form and guestbook pages from your local computer to the web server using an FTP program.

Step 9: View the guestbook form and submit an entry.

View the guestbook form in a web browser. Complete the guestbook form and submit an entry. On the guestbook processing page your entry is submitted to the guestbook database and you are redirected to the guestbook display page. Newer guestbook entries appear before older guestbook entries on the display page..



Sandra Clemmons of Maryland

Your products are the best. Keep up the good work. I am looking forward to the new line that is due out in March.

Pete Wilson of Atlanta, Georgia

Just wanted to say that my last order was a hit at my party. That is why today I am ordering some more. Also could you please tell me when will be the latest date I can order for xmas holidays... Have yourself a great day!

[Return to the Guestbooks Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Displaying Banner Ads

In the **Beginner Tutorial** section, learn how to schedule and display banner ads using a web-based interface. In the **Advanced Tutorial** section, learn advanced banner ad functions such as displaying client statistics. In the **Reference** section, view a complete technical reference for each tag used for banner ads. In the **Downloads** section, download example code for full beginner and advanced tutorials. In the **FAQs** section, read answers to frequently asked questions.

Beginner Banner Ad Tutorials

- B1. Scheduling and Displaying Banner Ads

Advanced Banner Ad Tutorials

- A1. Displaying Client Statistics

Banner Ad MGI Tag Reference

- mgiBannerAd

Banner Ad Downloads

- B1. Download example banner ad administration and display pages (B1Banner.zip - 28 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Scheduling and Displaying Banner Ads

Introduction

Banner ads are text and/or graphics advertisements for services, products, or web sites. Banner ads are usually sold by a block of impressions (hits) or click-throughs during a specific period of time. For example, if your web site receives 10,000 hits per month, you might sell 10 banner ad clients 1,000 impressions each on a monthly contract. The mgiBannerAd tag allows you to sell create and display groups of banner ads for different locations on a web page, for example.

In this example, 2 banner ads are added to the top of a home page.

MGI Tags

- mgiBannerAd

Steps

1. Create a banner ad administration page and open it in a text editor.
2. Insert the mgiBannerAd tag in Admin mode.
3. Save the banner ad administration page.
4. FTP the banner ad administration page to the web server running MGI.
5. View the banner ad administration page.
6. Add new banner ads.
7. Open the home page in a text editor.
8. Insert the mgiBannerAd tag in Display mode.
9. Save the home page.
10. FTP the home page and the banner ad images to the web server running MGI.

11. View the home page in a web browser.

Step 1: Create a banner ad administration page and open it in a text editor.

Create a page named "banneradmin.mgi" to display the web-based banner ad interface. Open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiBannerAd tag in Admin mode.

Insert your cursor after the beginning <BODY> element and enter the mgiBannerAd tag and mode parameter. In the mode parameter, enter "Admin":

```
<mgiBannerAd mode="Admin">
```

Step 3: Save the banner ad administration page.

Save the changes you have made to the banner ad administration page.

Step 4: FTP the banner ad administration page to the web server running MGI.

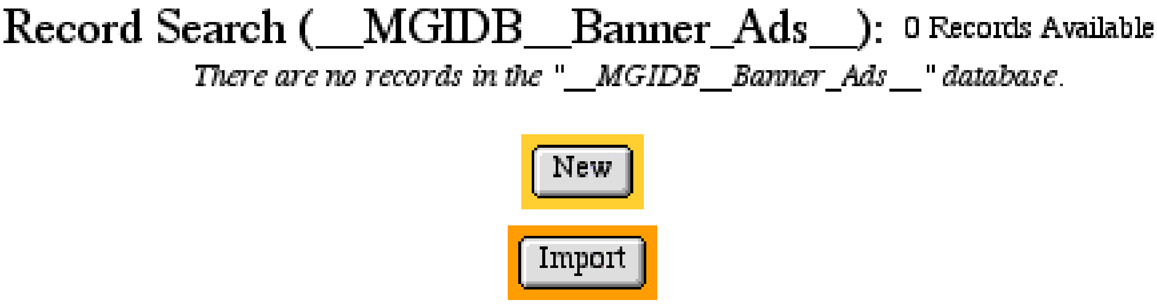
Upload the banner ad administration page from your local computer to the web server using an FTP program.

Step 5: View the banner ad administration page.

View the banneradmin.html page in a web browser. The first screen of the web-based, administration interface is displayed.

Step 6: Add new banner ads.

The first screen of the banner ad administration allows you to add or import banner ads (or search for banner ads if they already exist).

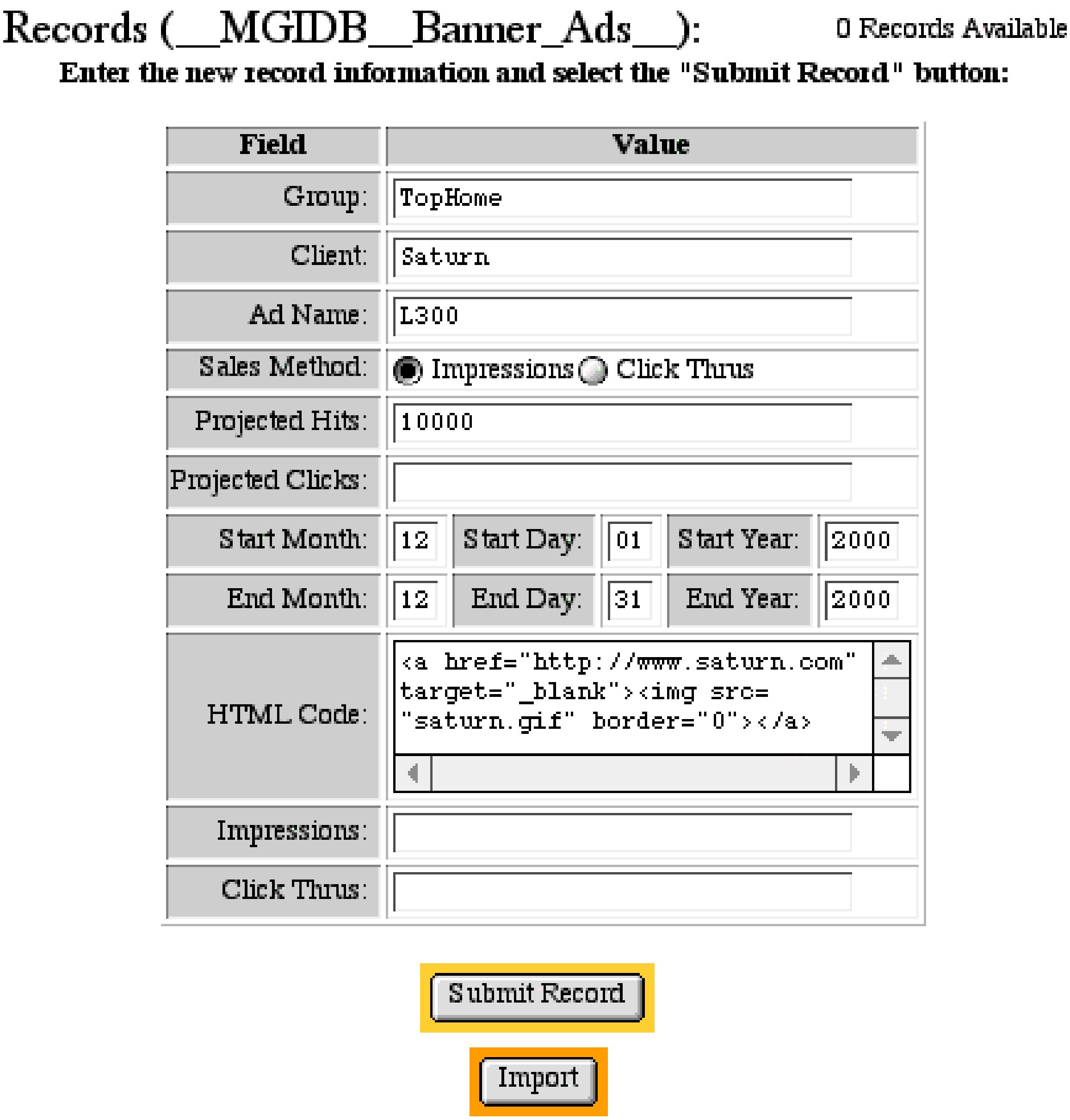


Record Search (__MGIDB__Banner_Ads__): 0 Records Available

There are no records in the "__MGIDB__Banner_Ads__" database.

To add a new banner ad, click the "New" button. On the new banner ad screen, enter information about the banner ad in the following fields.

- **Group.** The name of the group that the banner ad rotates with. A group may be a location on a page or any other organization of banner ads.
- **Client**: The name of the company or individual purchasing the banner ad. The client name is for your record keeping purposes and will be used when displaying client statistics.
- **Ad Name**: The name of the banner ad. This does not have to be the name of the image file. The ad name is for your record keeping and banner differentiation purposes only.
- **Sales Method**. Select "Impressions" to sell the banner ad by a block of hits (impressions). Select "Click Thrus" to sell the banner ad by the number of click-throughs from your site to the banner ad link.
- **Projected Hits**: If you select the"Impressions" sales method, enter the projected (sold) hits for the banner ad over the time period specified in the start and end date. Before selling banner ad space, you should get traffic reports for your web site to help you estimate the total hits that are available for sale.
- **Projected Clicks**. If you selected the "Click Thrus" sales method, enter the projected (sold) clicks for the banner ad over the time period specified in the start and end date.
- **Start Date**: The month, day and 4-digit year that the banner ad should be placed in rotation.
- **End Date**: The month, day and 4-digit year that the banner ad should be taken out of rotation.
- **HTML Code**: The HTML code including text, image sources and links that will display for the banner ad.
- **Impressions**: The total number of times the ad has been displayed on a page. The impressions are calculated automatically by mgiBannerAd and should not be entered for new banner ads.
- **Click Thrus**: The total number of times a user has clicked the banner ad and linked to the URL specified in the HTML Code. The click thrus are calculated automatically by mgiBannerAd and should not be enetered for new banner ads.


Records (__MGIDB__Banner_Ads__): 0 Records Available

Enter the new record information and select the "Submit Record" button:

Field	Value
Group:	TopHome
Client:	Saturn
Ad Name:	L300
Sales Method:	● Impressions ○ Click Thrus
Projected Hits:	10000
Projected Clicks:	
Start Month:	12 Start Day: 01 Start Year: 2000
End Month:	12 End Day: 31 End Year: 2000
HTML Code:	`<a href="http://www.saturn.com" target="_blank"><img src= "saturn.gif" border="0"></a>`
Impressions:	
Click Thrus:	

After submitting a new banner ad, the message "Record successfully added" displays and allows you to enter additional banner ads at the new banner screen. For this example, enter 2 banner ads with the following information:

Saturn Banner Ad:

Group: Top Home

Client: Saturn

Ad Name: L300

Sales Method: Impressions

Projected Hits: 10000

Start Date: 12/01/2000

End Date: 12/31/2000

HTML Code: <a href="http://www.saturn.com" target="_blank"><img

src="saturn.gif" border="0"></a>

Lexus Banner Ad:

Group: Top Home
Client: Lexus
Ad Name: AmEx
Sales Method: Impressions
Projected Hits: 10000
Start Date: 12/01/2000
End Date: 12/31/2000
HTML Code: <a href="http://www.lexus.com" target="_blank"><img src="lexus.gif" border="0"></a>

To view banner ads in the database, click the "First" button and use the ">>" (Next) and "<<" (Previous) buttons to browse through the records.

To locate specific banner ad records, click the "Search" button and enter search criteria into the "Group", "Client" and/or "Ad Name" fields (other fields cannot be searched) and click the "Search Now" button. Use an asterisk * for wildcard and partial searches. Select the radio button under the "Order" column to order search results by the selected field. Check the box under the "Rev" column beside the field you have selected in the "Order" column to reverse the order of search results. If no ordering is selected, search results are displayed in the ordered they were entered into the database. If a field is ordered, search results are ordered in ascending order (A to Z, smallest to largest) by default. If a field is ordered and reversed, search results are ordered in descending order (Z to A, largest to smallest). When results are displayed, select the radio button beside any search result and click the "View" button to view the full record.

To update a record, browse to view the record or locate and view the record via a search. While viewing the record, make changes in any field and click the "Save" button to save the changes.

When viewing a record, click the "Delete" button to delete that record. Click the "Delete All" button on any screen to delete all records in the database.

Close the banner ad administration page when all banner ads are entered correctly.

Step 7: Open the home page in a text editor.

Open the home page in a text editing program that allows you to view and modify the

HTML and code of the page.

Step 8: Insert the mgiBannerAd tag in Display mode.

Insert your cursor where you want banner ads to appear and enter the mgiBannerAd tag, group parameter and rotation parameter. In the group parameter, enter the group name "TopHome". In the rotation parameter, enter "Random" to have the banner ads rotate randomly and enter "Sequential" to have the banner ads rotate sequentially. Banner ads rotate randomly by default.

```
<mgiBannerAd group="TopHome" rotation="Random">
```

Step 9: Save the home page.

Save the changes you have made to the home page.

Step 10: FTP the home page and the banner ad images to the web server running MGI.

Upload the home page and the banner ad images (saturn.gif and lexus.gif in our example) to the web server running MGI.

Step 11: View the home page in a web browser.

View the home page in a browser. At the top of the page, either the Saturn banner ad or the Lexus banner ad appears. If you reload the page, a new (not necessarily different) banner ad appears. An algorithm that accounts for the number of banner ads, the length of banner ad contracts and the projected hits or clicks determines which banner ad from the set is displayed when the page is accessed.

Comments and Notes

The mgiBannerAd tag in Admin mode gives access to modify banner ad records. Keep your banner ads secure by password-protecting the banner ad administration page with an mgiAuthenticate or mgiAuthenticateDB tag.

[Return to the Banner Ads Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Displaying Client Statistics

Introduction

Banner ad clients may wish to check the impression and click-through statistics for their current banners. Using the clientStats mode of mgiBannerAd you can create a page to display each client's stats.

MGI Tags

- mgiBannerAd

Steps

1. Create a client stats page and open it in a text editor.
2. Insert the mgiBannerAd tag in clientStats mode.
3. Save the client stats page.
4. FTP the client stats page to the web server running MGI.
5. View the client stats page.

Step 1: Create a banner ad administration page and open it in a text editor.

Create a page named "clientstats.mgi" to display the banner ad statistics for the client. Open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiBannerAd tag in clientStats mode.

Enter the mgiBannerAd tag, mode parameter, and client parameter. In the mode parameter enter "clientStats". In the client parameter, enter the name of the client as it appears in the Banner Ad administration interface. If the client has multiple ads, all ads will be displayed in the client stats table.

A table containing the client's statistics will display at the location of the mgiBannerAd tag.

```
<mgiBannerAd mode="clientStats" client="Lexus">
```

Step 3: Save the client stats page.

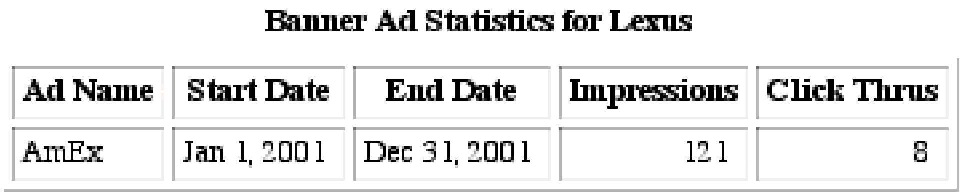
Save the changes you have made to the client stats page.

Step 4: FTP the client stats page to the web server running MGI.

Upload the client stats page from your local computer to the web server using an FTP program.

Step 5: View the client stats page.

View the client stats page in a web browser. A table containing the Ad Name, Start Date, End Date, and current Impressions and Click Thrus displays.



Banner Ad Statistics for Lexus

Ad Name	Start Date	End Date	Impressions	Click Thrus
AmEx	Jan 1, 2001	Dec 31, 2001	121	8

[Return to the Banner Ads Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Password Protecting Pages (Authentication)

In the **Beginner Tutorial** section, learn how to password protect pages with a single username and password or via IP number. In the **Advanced Tutorial** section, learn advanced authentication techniques including using multiple usernames and passwords with user groups, importing and exporting authentication data, and form administration of usernames and passwords (adding, deleting, and verifying usernames, changing passwords, and sending "forgotten" passwords). In the **Reference** section, view a complete technical reference for each tag used for authentication. In the **Downloads** section, download example code for full beginner and advanced tutorials. In the **FAQs** section, read answers to frequently asked questions.

Beginner Authentication Tutorials

- B1. Single Username and Password Protection
- B2. Multiple Username and Password Protection and User Groups

Advanced Authentication Tutorials

- A1. IP Authentication
- A2. Exporting Authentication Data
- A3. Importing Authentication Data

Authentication MGI Tag Reference

- mgiAuthenticate
- mgiAuthenticateDB

Authentication Downloads

- B1. Download a contract example password-protected with a single username and password (B1SingleAuth.zip - 4 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

Single Username and Password Protection

Introduction

The mgiAuthenticate and mgiAuthenticateDB tags password-protect individual pages of your web site. When a visitor accesses a page with an mgiAuthenticate or mgiAuthenticateDB tag, the web browser displays a dialogue box and prompts the user to enter a username and password. If the correct username and password are entered, the page displays. If either an incorrect username or password is entered, an error page displays.

Use the mgiAuthenticate tag to password-protect a page with one static username and password. Use the mgiAuthenticateDB tag to password-protect a page with multiple usernames and passwords. The mgiAuthenticateDB tag can also be used to provide levels of access via groups. The authentication tags can only be used to password-protected individual pages. The authentication tags cannot be used to password-protect entire directories (folders) unless all pages in the directory contain the mgiAuthenticate or mgiAuthenticateDB tag.

In this example, an online client contract is password-protected.

MGI Tags

- mgiAuthenticate

Steps

1. Create a contract page and open it in a text editor.
2. Insert the mgiAuthenticate tag.
3. Save the contract page.
4. FTP the contract page to the web server running MGI.
5. View the contract page in a browser.

Step 1: Create a contract page and open it in a text editor.

Create a client contract page named "contract.mgi" and open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiAuthenticate tag.

Insert your cursor after the beginning <BODY> element and enter the mgiAuthenticate tag, username parameter and password parameter. Enter "SysTek" as the value of the username parameter. Enter "3H84cy7" as the value of the password parameter:

```
<mgiAuthenticate username="SysTek" password="3H84cy7">
```

Step 3: Save the contract page.

Save the changes you have made to the contract page.

Step 4: FTP the contract page to the web server running MGI.

Upload the contract page from your local computer to the web server using an FTP program.

Step 5: View the contract page in a browser.

Access the contract page in a web browser. An authentication dialogue box requesting a username and password appears. Enter "SysTek" in the username field. Enter "3H84cy7" in the password field and click the "OK" button. If the correct username and password are entered, the contract page is displayed. If an incorrect username or password is entered, an error page is displayed.

Comments and Notes

Usernames and passwords are both case-sensitive. When you distribute a username and password, note the importance of capitalization to the recipient.

[Return to the Authentication Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Multiple Username and Password Protection and User Groups

Introduction

The mgiAuthenticate and mgiAuthenticateDB tags password-protect individual pages of your web site. When a visitor accesses a page with an mgiAuthenticate or mgiAuthenticateDB tag, the web browser displays a dialogue box and prompts the user to enter a username and password. If the correct username and password are entered, the page displays. If either an incorrect username or password is entered, an error page displays.

Use the mgiAuthenticate tag to password-protect a page with one static username and password. Use the mgiAuthenticateDB tag to password-protect a page with multiple usernames and passwords. The mgiAuthenticateDB tag can also be used to provide levels of access via groups. The authentication tags can only be used to password-protected individual pages. The authentication tags cannot be used to password-protect entire directories (folders) unless all pages in the directory contain the mgiAuthenticate or mgiAuthenticateDB tag.

In this example, password protection is based on two groups, admin and staff. If a group parameter is not included in the mgiAuthenticateDB tag, users from any group with a valid username and password may access the page.

MGI Tags

- mgiAuthenticateDB

Steps

1. Create an authentication administration page.
2. Insert the mgiAuthenticateDB tag in Admin mode.
3. Save the authentication administration page.
4. FTP the authentication administration page to the web server running MGI.

5. View the authentication administration page.
6. Add users.
7. Open pages to password protect in a text editor.
8. Insert the mgiAuthenticateDB tag and group parameter.
9. Save the pages.
10. FTP the pages to the web server running MGI.
11. View the pages in a browser.

Step 1: Create an authentication administration page.

Create a page named "authadmin.mgi" to display the web-based authentication interface. Open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiAuthenticateDB tag in Admin mode.

Enter the mgiAuthenticateDB tag and mode parameter. In the mode parameter enter "Admin". The admin interface will display at the location of the mgiAuthenticateDB tag.

```
<mgiAuthenticateDB mode="Admin">
```

Step 3: Save the authentication administration page.

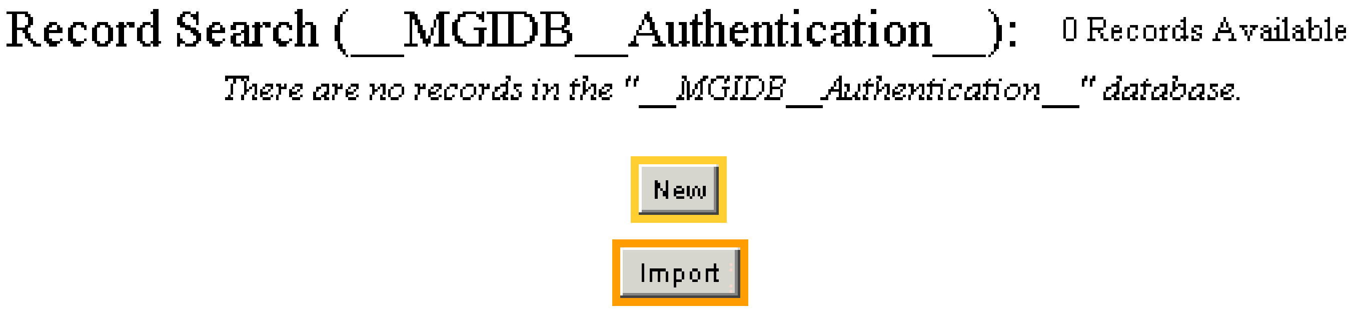
Save the changes you have made to the authentication administration page.

Step 4: FTP the authentication administration page to the web server running MGI.

Upload the authentication administration page from your local computer to the web server using an FTP program.

Step 5: View the authentication administration page.

The first screen of the authentication administration interface allows you to add or import users (or search for users if they already exist). See instruction for importing in the advanced tutorials.



Record Search (__MGIDB__Authentication__): 0 Records Available

There are no records in the "__MGIDB__Authentication__" database.

Step 6: Add users.

To create a new user, click the "New" button. Enter the new user information in the form that displays.

The username and password fields are required. In the Username field, enter the user's unique, case-sensitive login identification. Usernames must be unique, even across groups. Since the username is case-sensitive, the same name with different capitalization qualifies as a unique username. In the Password field, enter the user's case-sensitive security code.

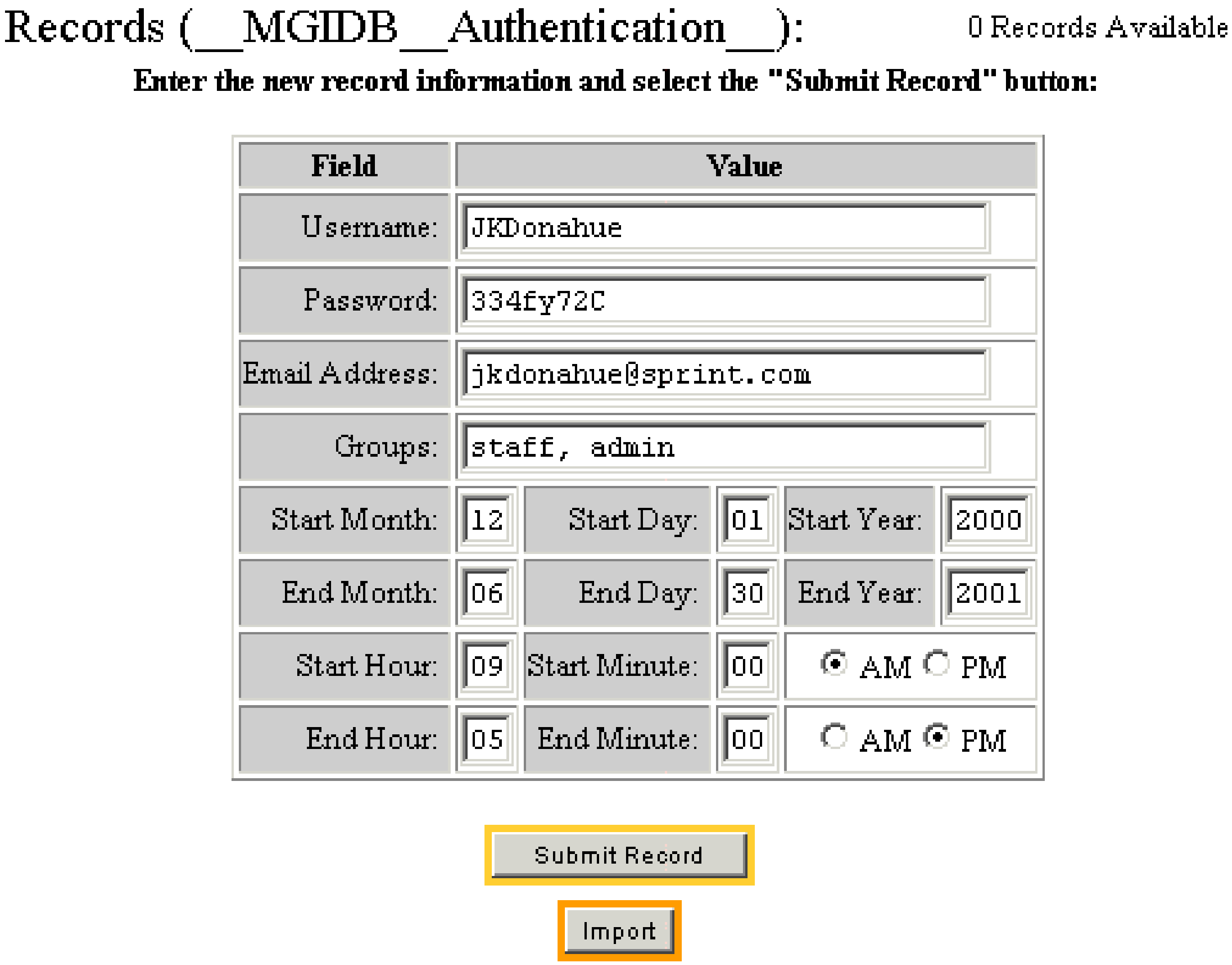
In the Email Address field, enter the user's email address. The email address can be used to send the user's password if it is forgotten.

In the Groups field, enter the group name that the user belongs to. For multiple groups, enter a comma-delimited list in the Groups field.

In the start date fields, enter the numeric month, day and 4-digit year when the user's username and password become valid. In the end date fields, enter the numeric month, day and 4-digit year when the user's username and password are no longer valid. To give a user access for all dates, leave the start and end date fields blank - they will default automatically.

In the start time fields, enter the time that the user's username and password become valid on the start date. In the end time fields, enter the time that the user's username and password are no longer valid on the end date. To give a user access for all times, leave the start and end time fields blank - they will default automatically.

Click "Submit Record" to add the user. The message "Record successfully added." and a blank form for adding additional users displays when the addition is complete.



Records (__MGIDB__Authentication__): 0 Records Available

Enter the new record information and select the "Submit Record" button:

Field	Value				
Username:	JKDonahue				
Password:	334fy72C				
Email Address:	jkdonahue@sprint.com				
Groups:	staff, admin				
Start Month:	12	Start Day:	01	Start Year:	2000
End Month:	06	End Day:	30	End Year:	2001
Start Hour:	09	Start Minute:	00	● AM ○ PM	
End Hour:	05	End Minute:	00	○ AM ● PM	

Step 7: Open pages to password protect in a text editor.

Open any page you wish too password protect in a text editing program that allows you to view and modify the HTML and code of the page.

Step 8: Insert the mgiAuthenticateDB tag and group parameter.

Enter the mgiAuthenticateDB tag and group parameter. In the group parameter, enter the name of the group that is allowed to access the page. You may enter only one group name. If the group parameter is not included, a user from any group with a valid username and password may access the page.

```
<mgiAuthenticateDB group="staff">
```

Step 9: Save the pages.

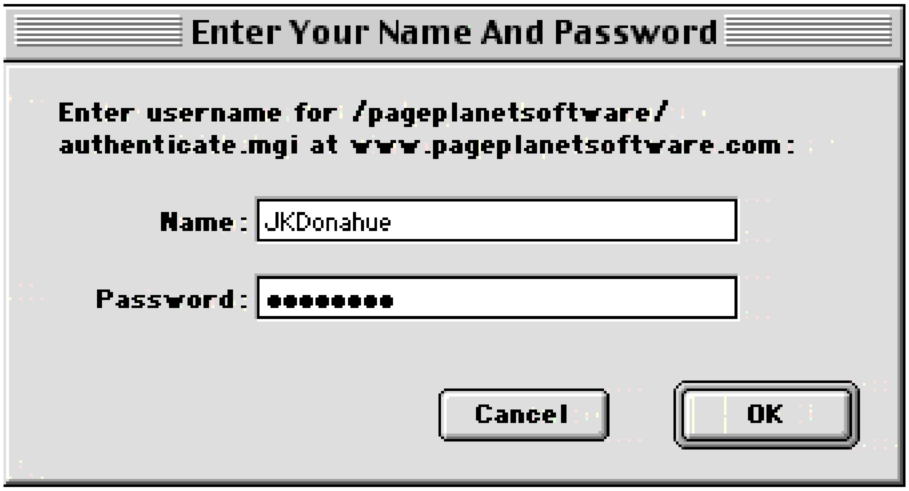
Save the changes you have added to password protected pages.

Step 10: FTP the pages to the web server running MGI.

Upload the password protected pages from your local computer to the web server using an FTP program.

Step 11: View the pages in a browser.

View a password protected page in a browser. A dialogue box prompts you for your username and password. If you enter a valid username and password, the page displays. If you enter a valid username and password, but you are not a member of the specified group, an error displays. If you enter an invalid username or password, an error displays.



Enter Your Name And Password

Enter username for /pageplanetsoftware/ authenticate.mgi at www.pageplanetsoftware.com:

Name: JKDonahue

Password: ●●●●●●●●●

Comments and Notes

Usernames and passwords are both case-sensitive. When you distribute a username and password, note the importance of capitalization to the recipient.

[Return to the Authentication Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

IP Authentication

Introduction

The "AuthenticateIPOnly" mode of mgiAuthenticateDB allows you to grant access to pages based on the visitor's IP number or range. You may also block access to a page based on the visitor's IP number or range.

If the visitor accesses a protected page from an allowed IP, the page displays. If the visitor accesses the page from an IP that is specifically denied an error displays. If the allowedIP parameter is not included, the page displays for visitors from IP addresses that are not specifically denied. If the deniedIP parameter is not included, an error displays for visitors from IP addresses that are not specifically allowed.

The following is a generic example of IP authentication of a page.

MGI Tags

- mgiAuthenticateDB

Steps

1. Open the page to protect in a text editor.
2. Insert the mgiAuthenticateDB tag in AuthenticateIPOnly mode.
3. Save the page.
4. FTP the page to the web server running MGI.
5. View the page in a browser.

Step 1: Open the page to protect in a text editor.

Open the page you wish to protect via IP number or hostname in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiAuthenticateDB tag in AuthenticateIPOnly mode.

At the top of the page, enter the mgiAuthenticateDB tag, mode parameter, allowedIP parameters, and deniedIP parameters.

In the mode parameter, enter "AuthenticateIPOnly". At least one allowedIP or deniedIP parameter is required. To allow a specific IP or range of IPs, enter the IP in an allowedIP parameter. To deny a specific IP or range of IPs, enter the IP in a deniedIP parameter. You may use the asterisk (*) as a wildcard in any part of the IP number to represent the full range of an IP block. For example, if you want to allow all visitors from the "205.160.14" Class C block to access a page, you would enter "205.160.14.*" as the parameter value. You may also use multiple allowedIP and deniedIP parameters in one mgiAuthenticateDB tag to create any combination of allowed and denied IPs you prefer.

Should an IP number fall within more than one allowedIP or deniedIP parameters, the most restrictive parameter will execute. For example, if you have an allowedIP for "205.160.14.*" and a deniedIP for "205.160.14.230", then a visitor from "205.160.14.230" will be denied.

The following example allows all users from the "205.145.26.*" Class C block and the "206.125.12.*" Class C block and denies access to the users "205.145.26.15" and "206.125.12.145".

```
<mgiAuthenticateDB mode="AuthenticateIPOnly"
allowedIP="205.145.26.*" allowedIP="206.125.12.*"
deniedIP="205.145.26.15" deniedIP="206.125.12.145">
```

Step 3: Save the page.

Save the changes you have made to the page.

Step 4: FTP the page to the web server running MGI.

Upload the page from your local computer to the web server using an FTP program.

Step 5: View the page in a browser.

Access the IP authenticated page in a web browser. If your IP address matches an allowed IP address or range and does not match a denied IP address or range, the page displays. If your IP address does not match an allowed IP address or range or does match a denied IP address or range, an error displays.

[Return to the Authentication Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Exporting Authentication Data

Introduction

MGI stores usernames, passwords, and other authentication data in a proprietary database format for the database that is built into the MGI code. However, you can use the export and import features of the mgiAuthenticateDB tag in admin mode to transfer authentication records. Passwords are stored in encrypted format in the internal "__MGIDB__Authentication__" database. Do not export records using the mgiEditDatabase tag or the passwords will export in encrypted format. The mgiAuthenticateDB tag in admin mode will decrypt passwords as they are exported and will export them in plain text.

MGI Tags

- mgiAuthenticateDB

Steps

1. Create an authentication administration page.
2. Insert the mgiAuthenticateDB tag in Admin mode.
3. Save the authentication administration page.
4. FTP the authentication administration page to the web server running MGI.
5. View the authentication administration page.
6. Export Users.

Step 1: Create an authentication administration page.

Create a page named "authadmin.mgi" to display the web-based authentication interface. Open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiAuthenticateDB tag in Admin mode.

Enter the mgiAuthenticateDB tag and mode parameter. In the mode parameter enter "Admin". The admin interface will display at the location of the mgiAuthenticateDB tag.

```
<mgiAuthenticateDB mode="Admin">
```

Step 3: Save the authentication administration page.

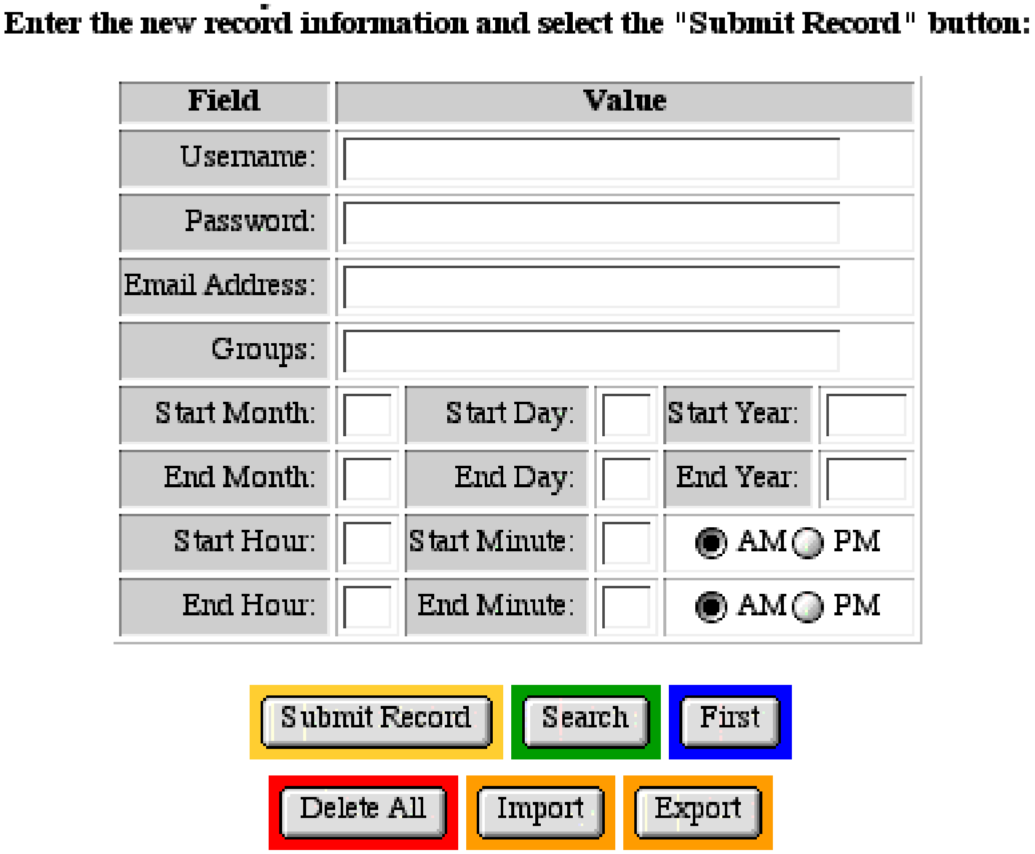
Save the changes you have made to the authentication administration page.

Step 4: FTP the authentication administration page to the web server running MGI.

Upload the authentication administration page from your local computer to the web server using an FTP program.

Step 5: View the authentication administration page.

The first screen of the authentication administration interface is the search screen. Click the "Export" button to view the export screen.



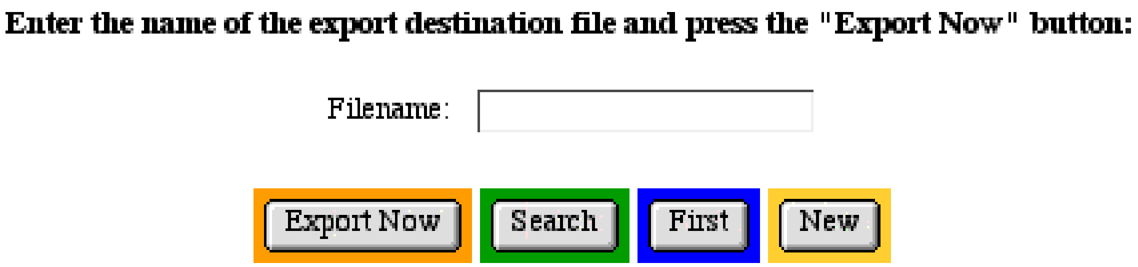
Enter the new record information and select the "Submit Record" button:

Field	Value				
Username:					
Password:					
Email Address:					
Groups:					
Start Month:		Start Day:		Start Year:	
End Month:		End Day:		End Year:	
Start Hour:		Start Minute:		● AM ○ PM	
End Hour:		End Minute:		● AM ○ PM	

Step 6: Export Users.

Beside "Filename", enter the name of the file to export. Exported files are saved to the web server in the same folder as the database administration page. Use a unique name if you do not wish to overwrite an existing file!

To export, click the "Export Now" button. Records are exported to the specified file on the web server. When the export is complete the search screen displays.The exported file is written in tab-delimited format to the same folder as the database administration page. You may use an FTP program to download the exported file.



Enter the name of the export destination file and press the "Export Now" button:

Filename:

The first line of the exported file contains a comment "!Generated by MGI!". The second line of the exported file contains the field definitions (in parentheses) to the left of each field name in the internal Authentication database. Each type of MGI field (text, long text, boolean, etc.) is represented by a letter:

- **B** - True/False (Boolean)
- **I** - Whole Number (Integer)
- **U** - Positive Number (Unsigned Integer)
- **N** - Decimal Number (Multi-Precision Float)
- **T** - Text
- **L** - Long Text

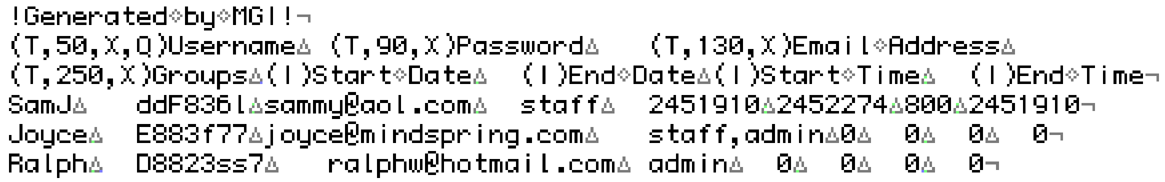
Text and decimal number fields are followed by the specified number of characters or decimal places. Field definitions with an "X" are indexed. Field definitions with a "Q" are unique.

The fields exported with the authentication data are the following and are in the following order:

- **Username**
- **Password**
- **Email Address**
- **Group**
- **Start Date** (in Julian Day format)
- **End Date** (in julian Day format)
- **Start Time** (in military format)
- **End Time** (in military format)

The remaining lines in the exported file are the tab-delimited authentication data of each user. All lines in the exported file end with a carriage return (Mac) or carriage return/line fee (NT).

The following is an example exported authentication file. The text in this example is wrapped for viewing purposes, however each record is actually only one row of tab-delimited text. The triangles in this example represent tabs and the horizontal lines as the end of each row represent a carriage return.



```
!Generated◊by◊MGI!¬
(T,50,X,Q)Username△ (T,90,X)Password△   (T,130,X)Email◊Address△
(T,250,X)Groups△(I)Start◊Date△  (I)End◊Date△(I)Start◊Time△  (I)End◊Time¬
SamJ△   ddF836l△sammy@aol.com△  staff△  2451910△2452274△800△2451910¬
Joyce△  E883f77△joyce@mindspring.com△   staff,admin△0△  0△  0△  0¬
Ralph△  D8823ss7△   ralphw@hotmail.com△ admin△  0△  0△  0△  0¬
```

[Return to the Authentication Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Importing Authentication Data

Introduction

MGI stores usernames, passwords, and other authentication data in a proprietary database format for the database that is built into the MGI code. However, you can use the export and import features of the mgiAuthenticateDB tag in admin mode to transfer authentication records. Passwords are stored in encrypted format in the internal "__MGIDB__Authentication__" database. Do not import records using the mgiEditDatabase tag or the passwords will import in plain text (and will erroneously decrypt in the admin). The mgiAuthenticateDB tag in admin mode will encrypt passwords as they are imported .

MGI Tags

- mgiAuthenticateDB

Steps

1. Create an authentication administration page.
2. Insert the mgiAuthenticateDB tag in Admin mode.
3. Save the authentication administration page.
4. Create a tab-delimited import file.
5. Save the import file.
6. FTP the authentication administration page and import file to the web server running MGI.
7. View the authentication administration page.
8. Import Users.

Step 1: Create an authentication administration page.

Create a page named "authadmin.mgi" to display the web-based authentication interface. Open the page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiAuthenticateDB tag in Admin mode.

Enter the mgiAuthenticateDB tag and mode parameter. In the mode parameter enter "Admin". The admin interface will display at the location of the mgiAuthenticateDB tag.

```
<mgiAuthenticateDB mode="Admin">
```

Step 3: Save the authentication administration page.

Save the changes you have made to the authentication administration page.

Step 4: Create a tab-delimited import file.

Depending on the current format of your data, you may need to "export" or "save as..." from an existing program in order to create a tab-delimited file. If you choose, you may also prepare your data directly in a text file. In any case, create a text file of the tab-delimited authentication data to import.

Your data should be in the following field order and format:

- **Username**
- **Password**
- **Email Address**
- **Group**
- **Start Date** (in Julian Day format)
- **End Date** (in julian Day format)
- **Start Time** (in military format)
- **End Time** (in military format)

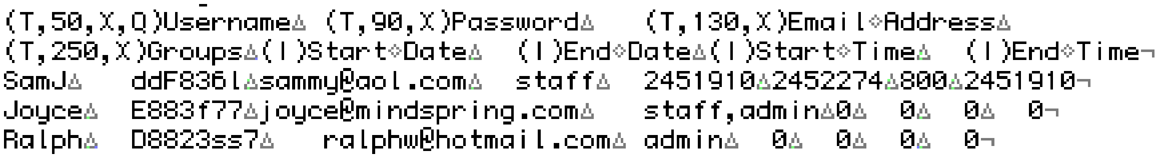
At least a username and password is required for each user.

For the Username, enter the user's unique, case-sensitive login identification. For the Password, enter the user's case-sensitive security code. For the Email Address, enter the user's email address. The email address will be used to send the user's password in SendPassword mode. For the Groups, enter the group name that the user belongs to. For multiple groups, enter a comma-delimited list in the Groups field. For the start date, enter the julian date when the user's username and password become valid. For the end date, enter the julian date when the user's username and password are no longer valid. For the start time, enter the time (in military format) that the user's username and password become valid on the start date. For the end time, enter the time (in military format) that the user's username and password are no longer valid on the end date.

The first line of your import file should be the authentication database field definitions and names. Note that there are tabs between each field name and a carriage return (Mac) or carriage return and line feed (PC) at the end of the first line which may not copy directly from the example below..

(T,50,X,Q)Username (T,90,X)Password (T,130,X)Email Address (T,250,X)Groups (I)Start Date (I)End Date (I)Start Time (I)End Time

The following is an example import file. The text in this example is wrapped for viewing purposes, however each record is actually only one row of tab-delimited text. The triangles in this example represent tabs and the horizontal lines as the end of each row represent a carriage return.



```
(T,50,X,Q)Usernameᐃ (T,90,X)Passwordᐃ   (T,130,X)Email◊Addressᐃ
(T,250,X)Groupsᐃ(I)Start◊Dateᐃ  (I)End◊Dateᐃ(I)Start◊Timeᐃ  (I)End◊Time¬
SamJᐃ   ddF836lᐃsammy@aol.comᐃ  staffᐃ  2451910ᐃ2452274ᐃ800ᐃ2451910¬
Joyceᐃ  E883f77ᐃjoyce@mindspring.comᐃ   staff,adminᐃ0ᐃ  0ᐃ  0ᐃ  0¬
Ralphᐃ  D8823ss7ᐃ   ralphw@hotmail.comᐃ adminᐃ  0ᐃ  0ᐃ  0ᐃ  0¬
```

Step 5: Save the import file.

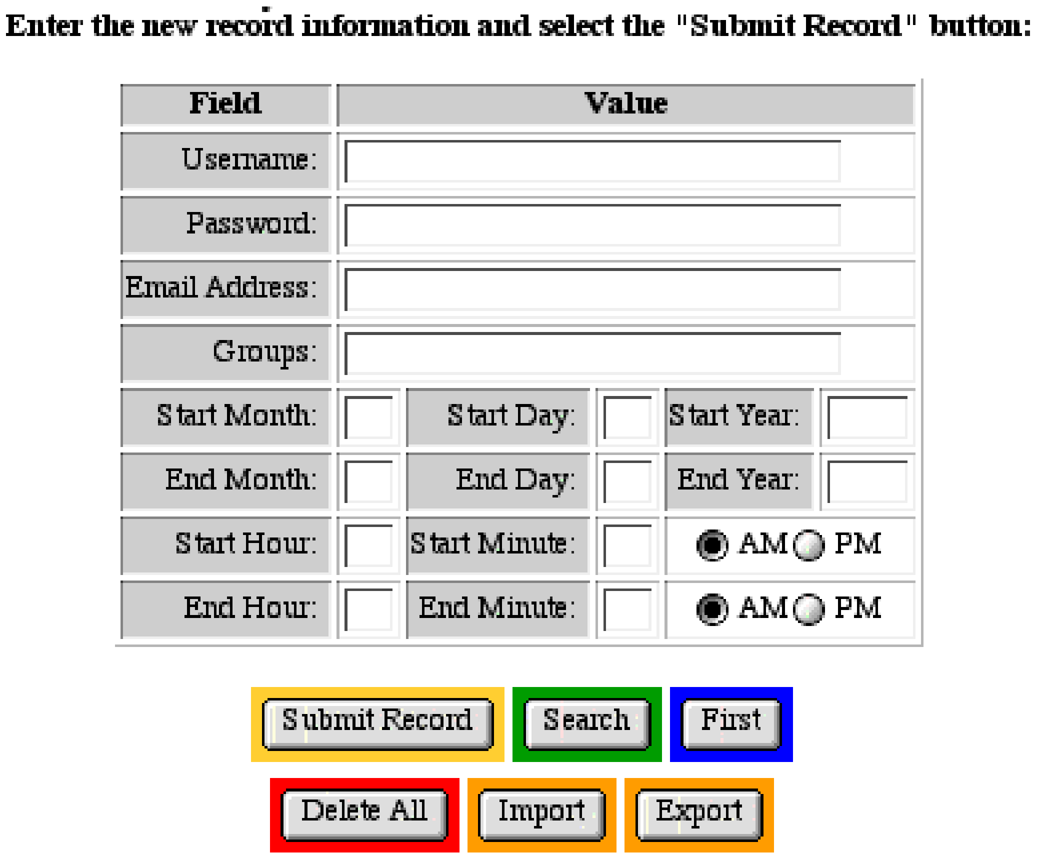
Save the import file. In this example the import file is named "authimport.txt", but you may choose any name.

Step 6: FTP the authentication administration page and import file to the web server running MGI.

Upload the authentication administration page and import file from your local computer to the web server using an FTP program.

Step 7: View the authentication administration page.

The first screen of the authentication administration interface is the search screen if there is existing data or a "New" and "Import" button if there is no existing data. Click the "Import" button to view the import screen.

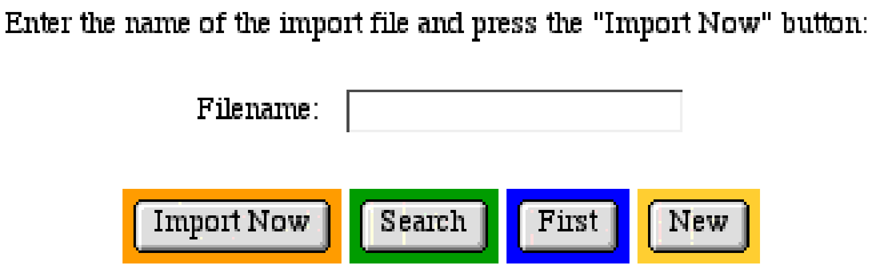


Enter the new record information and select the "Submit Record" button:

Field	Value				
Username:					
Password:					
Email Address:					
Groups:					
Start Month:		Start Day:		Start Year:	
End Month:		End Day:		End Year:	
Start Hour:		Start Minute:		● AM ○ PM	
End Hour:		End Minute:		● AM ○ PM	

Step 8: Import Users.

Beside "Filename", enter the name of the file to import and click the "Import Now" button. Imported users are added to the existing users in the database. If a username in the import file exactly matches an existing username in the authentication database, the user's information is not imported. When the import is complete, the main interface displays.



Enter the name of the import file and press the "Import Now" button:

Filename:

[Return to the Authentication Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Administering Polls and Surveys

In the **Beginner Tutorial** section, learn how to collect and display survey information. In the **Advanced Tutorial** section, learn advanced poll techniques including custom results. In the **Reference** section, view a complete technical reference for each tag used for polls. In the **Downloads** section, download example code for each beginner and advanced tutorial. In the **FAQs** section, read answers to frequently asked questions.

Beginner Poll Tutorials

- B1. Collecting and Displaying Poll Results

Poll MGI Tag Reference

- mgiPoll

Poll Downloads

- B1. Download a presidential poll example (B1Poll.zip - 2.0 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Collecting and Displaying Poll Results

Introduction

Use the mgiPoll tag to get opinions and preferences from visitors to your site. Allow visitors to participate in the poll and/or simply view the current poll results.

In this example, you will create a poll to measure presidential nominee preferences.

MGI Tags

- mgiPoll

Steps

1. Create a poll form.
2. Create a poll administration page and open it in a text editor.
3. Insert the mgiPoll tag in Admin mode.
4. Save the poll administration page.
5. FTP the poll administration page to the web server running MGI.
6. View the poll administration page in a web browser.
7. Add a new poll.
8. Add post arguments.
9. Add post argument responses.
10. Set poll preferences.
11. Activate the poll.
12. Create a poll results page and open it in a text editor.
13. Insert an mgiPoll tag in Collect mode and an mgiPoll tag in Tally mode.
14. Save the poll results page.
15. FTP the poll form and poll results page to the web server running MGI.
16. View the poll form in a web browser and complete the poll.

Step 1: Create a poll form.

Create a form named "poll.mgi" to collect presidential preferences using form elements such as radio buttons, checkboxes, and pop-up menus. Text fields and text areas cannot be used to collect and tally poll data.

Name each form element uniquely. For radio buttons, name the set of radio buttons with one name and enter a unique value for each radio button. For checkboxes, name the set of checkboxes with one name and enter a unique value for each checkbox. If you name each checkbox separately, they will show up as different results rather than components of one question. For selects (pop-up menus), name the pop-up menu uniquely and enter a value for each option.

The post argument names cannot exceed 35 characters. Post argument names are not shown to the visitor and can be abbreviated (e.g., "Q1", "Q2", etc.). The value of the post argument is not limited and may be any length. The post argument value is shown to the visitor as a label for poll results with the text entered into the mgiPoll admin (see Step 10).

If a visitor answers some poll questions, unanswered questions are tallied as a "No Response" value by default. If a visitor answers no poll questions, but submits the poll, that visitor's responses are not tallied. To control how your visitors answer poll questions, you may choose to include your own "No Answer" value for each question or you may wish to simply require an answer to each question by validating each question on the poll results page (using mgiValidataData or a custom checking method).

After the form elements, enter a submit button named "Submit".

Enclose all form elements with HTML <FORM> tags and post the form to the poll results page (results.mgi).

The following is an example poll form with three questions.

```
<FORM ACTION="results.mgi" METHOD="POST">

<CENTER>

<P><TABLE WIDTH="300" BORDER="0" CELLSPACING="0"
CELLPADDING="5">
<TR>
```

```
<TD WIDTH="100%" BGCOLOR="#00ff00">If you had to vote
today, who would you choose to represent the Republican
party in the next presidential election?</TD></TR>
<TR>
<TD WIDTH="100%" BGCOLOR="#ccffcc">
<P><INPUT TYPE="radio" VALUE="John McCain"
NAME="Republican"> John McCain</P>
<P><INPUT TYPE="radio" VALUE="George W. Bush"
NAME="Republican"> George W. Bush</P>
<P><INPUT TYPE="radio" VALUE="Elizabeth Dole"
NAME="Republican"> Elizabeth Dole</P>
<P><INPUT TYPE="radio" VALUE="Other"
NAME="Republican"> Other
</TD></TR>
</TABLE>

<P><TABLE WIDTH="300" BORDER="0" CELLSPACING="0"
CELLPADDING="5">
<TR>
<TD WIDTH="100%" BGCOLOR="#ffff00">If you had to
vote today, who would you choose to represent the
Democratic party in the next presidential election?
</TD></TR>
<TR>
<TD WIDTH="100%" BGCOLOR="#ffffcc">
<P><INPUT TYPE="radio" VALUE="Hillary Clinton"
NAME="Democrat"> Hillary Clinton</P>
<P><INPUT TYPE="radio" VALUE="Dick Gephart"
NAME="Democrat"> Dick Gephart</P>
<P><INPUT TYPE="radio" VALUE="Al Gore"
NAME="Democrat"> Al Gore</P>
<P><INPUT TYPE="radio" VALUE="Other"
NAME="Democrat"> Other
</TD></TR>
</TABLE>

<P><TABLE WIDTH="300" BORDER="0" CELLSPACING="0"
CELLPADDING="5">
<TR>
<TD WIDTH="100%" BGCOLOR="#00ccff">What is your age?
</TD></TR>
<TR>
<TD WIDTH="100%" BGCOLOR="#ccffff">
<SELECT NAME="Age">
<OPTION SELECTED>18 to 25
```

```
<OPTION>26 to 35
<OPTION>36 to 45
<OPTION>46 to 55
<OPTION>56 to 65
<OPTION>over 65
</SELECT>
</TD></TR>
</TABLE>

<P><mgiButton value="Submit"></H2>

</CENTER>

</FORM>
```

Step 2: Create a poll administration page and open it in a text editor.

Create a page named "polladmin.mgi" to display the web-based poll administration interface. Open the poll administration page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 4: Insert the mgiPoll tag in Admin mode.

Insert your cursor after the beginning <BODY> element and enter the mgiPoll tag and mode parameter. In the mode parameter, enter "Admin".

```
<mgiPoll mode="admin">
```

Step 5: Save the poll administration page.

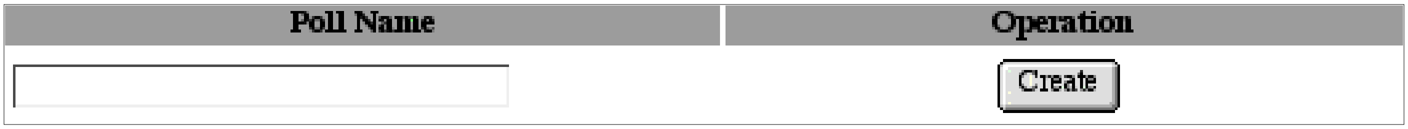
Save the changes you have made to the poll administration page.

Step 6: FTP the poll administration page to the web server running MGI.

Upload the poll administration page from your local computer to the web server using an FTP program.

Step 7: View the poll administration page in a web browser.

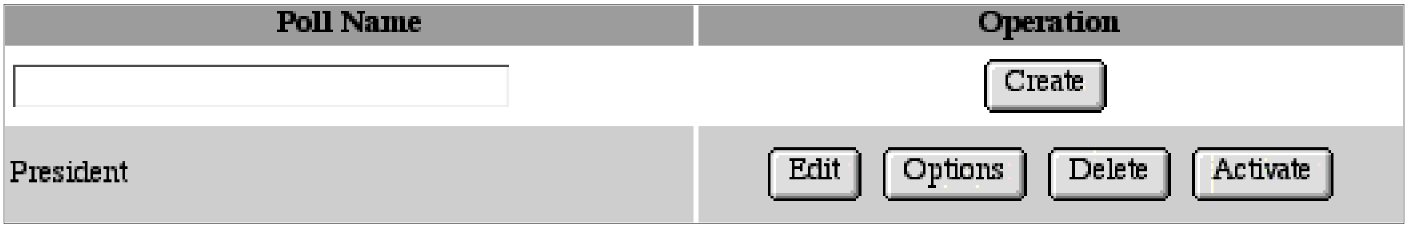
View the poll administration page in a web browser. The first screen of the web-based, administration interface displays.



Poll Name	Operation
	Create

Step 8: Add a new poll.

In the text field under the "Poll Name" column, enter the case-sensitive poll name ("President" for this example) and click the "Create" button.

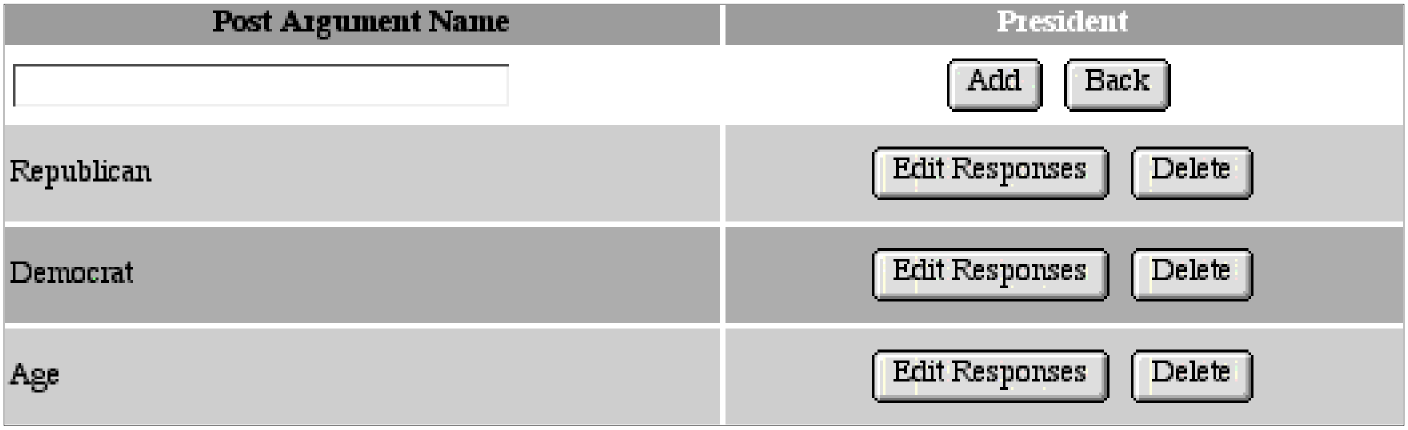


Poll Name	Operation
	Create
President	Edit Options Delete Activate

Step 9: Add post arguments.

Click the "Edit" button beside the poll to access the post argument interface. For each poll question (i.e., for each unique post argument) add a post argument to the poll.

In the text field under the "Post Argument Name" column, enter the post argument name (e.g., "Republican") and click the "Add" button.

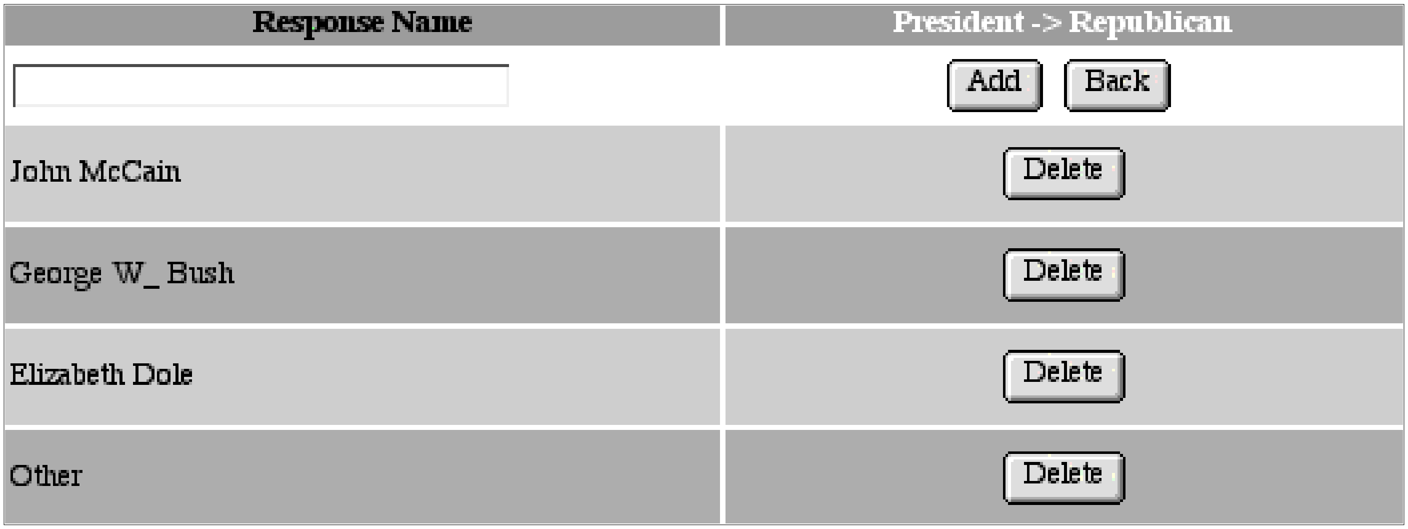


Post Argument Name	President
	Add Back
Republican	Edit Responses Delete
Democrat	Edit Responses Delete
Age	Edit Responses Delete

Step 10: Add post argument responses.

Add all possible responses for each poll question. Click the "Edit Responses" button beside the post argument name to access the post argument response interface. Add a response for each answer to the selected poll question. To add a response, enter the post argument value in the text field under the "Response Name" column and click the "Add" button.

Click the "Back" button to return to the post argument interface.



Response Name	President -> Republican
	Add Back
John McCain	Delete
George W_ Bush	Delete
Elizabeth Dole	Delete
Other	Delete

Step 11: Set poll preferences.

Return to the first screen of the poll interface. Click the "Options" button to access the poll preferences. You may set the method, graph dimensions, protection, color cycle method, and color cycle.

Select the tally method. The "Total Votes" method displays results as the raw number of responses for each value. The "Percentage of Total Votes" method displays the results the percentage of the total responses for the question.

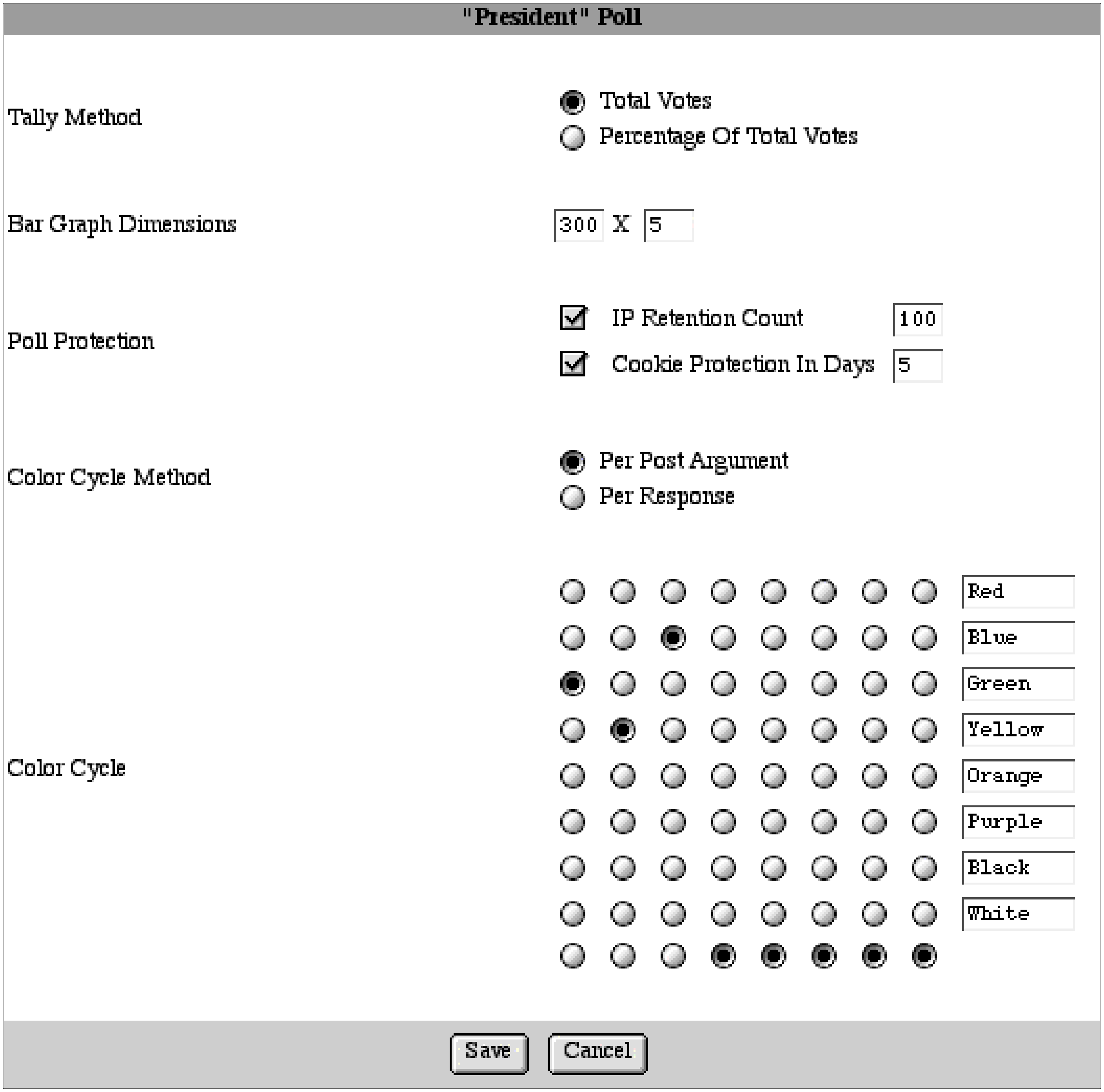
Enter the width and height (in pixels) of the bar for each response. If the width is greater than the height, bars are displayed horizontally for each question. If the height is greater than the width, bars are displayed vertically for each question.

To prevent visitors from completing the poll multiple times, set IP or Cookie protection. For IP protection, select the IP Retention Count checkbox and enter the number of IPs to store. If a visitor's IP number matches an IP number in the poll database, the visitor's poll results are not stored or tallied. Once the poll IP database is full, the oldest IP number in the database is replaced with the IP number of the next visitor. For cookie protection, select the Cookie Protection in Days checkbox and enter the number of days to store a cookie on the visitor's machine. If a visitor with a poll cookie completes the poll, their poll results are not stored or tallied.

Select the color cycle method. The "Per Post Argument" method cycles the specified colors by each question (e.g., the first question's bars are blue, the second question's bars are green, etc.). The "Per Response" method cycles the specified colors by each response (e.g., the first response to question one is blue, the second response to question one is green, etc.).

Set the color cycle by choosing the order of colors. In the first column select the first color, in the second column select the second color, etc. Only selected colors will be cycled even if a column is skipped.

Click the "Save" button to apply poll preferences.



"President" Poll

Tally Method

● Total Votes
○ Percentage Of Total Votes

Bar Graph Dimensions

300 X 5

Poll Protection

☑ IP Retention Count 100
☑ Cookie Protection In Days 5

Color Cycle Method

● Per Post Argument
○ Per Response

Color Cycle

○	○	○	○	○	○	○	○	Red
○	○	●	○	○	○	○	○	Blue
●	○	○	○	○	○	○	○	Green
○	●	○	○	○	○	○	○	Yellow
○	○	○	○	○	○	○	○	Orange
○	○	○	○	○	○	○	○	Purple
○	○	○	○	○	○	○	○	Black
○	○	○	○	○	○	○	○	White
○	○	○	●	●	●	●	●	

Step 12: Activate the poll.

On the first screen of the poll interface, activate the poll by clicking the "Activate" button. Poll data cannot be collected or tallied while the poll is Deactivated. A poll may be Deactivated for changes, but must be re-Activated before the poll can proceed.

If a visitor attempts to submit to a deactivated poll, an error message will display indicating that the poll is not active.

Step 13: Create a poll results page and open it in a text editor.

Create a poll results page named "results.mgi" to collect poll data and display poll results and open the poll results page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 14: Insert an mgiPoll tag in Collect mode and an mgiPoll tag in Tally mode.

On the poll results page, enter two mgiPoll tags. The first mgiPoll tag collects poll data and adds it to the poll database. The second mgiPoll tag displays poll results. If you prefer, the data collection and results display can be performed on two different pages.

For data collection, enter the mgiPoll tag and name parameter. In the name parameter, enter the case-sensitive name of the poll where data is stored.

For results display, enter the mgiPoll tag, mode parameter and name parameter. In the mode parameter, enter "Tally". In the name parameter, enter the case-sensitive name of the poll to tally and display.

```
<mgiComment>Collect Poll Data</mgiComment>
<mgiPoll name="President">

<mgiComment>Display Poll Results</mgiComment>
<H2>Poll Results</H2>
<mgiPoll mode="tally" name="President">
```

Step 15: Save the poll results page.

Save the changes you have made to the poll results page.

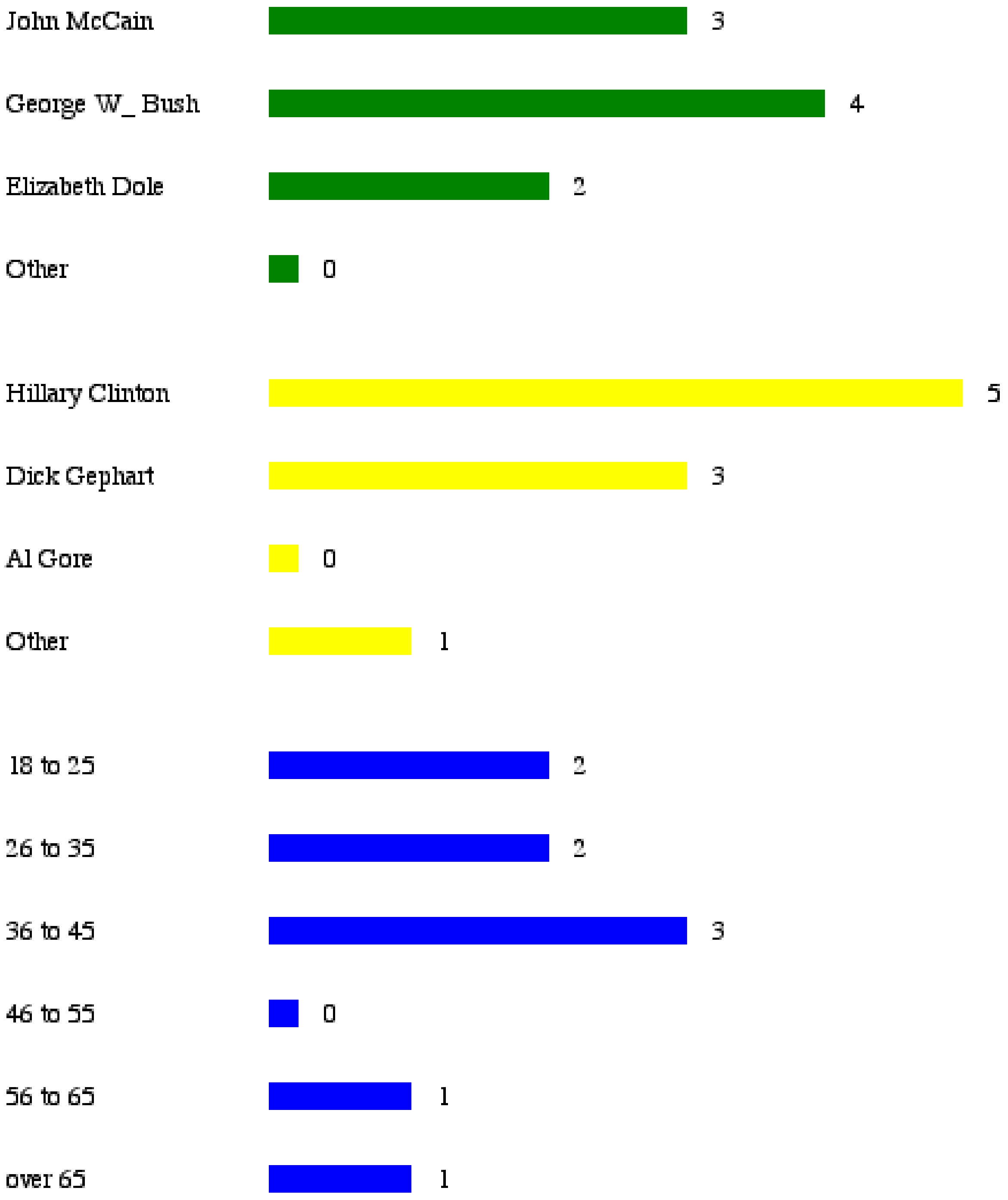
Step 16: FTP the poll form and poll results page to the web server running MGI.

Upload the poll form and poll results page from your local computer to the web server using an FTP program.

Step 17: View the poll form in a web browser and complete the poll.

View the poll form in a web browser. Complete and submit the poll form. On the results page, your poll information is added to the poll database and the current poll results are displayed. For example,

Poll Results



[Return to the Administering Polls and Surveys Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

Administering Online Quizzes

In the **Beginner Tutorial** section, learn how to create, grade and manage online quizzes. In the **Advanced Tutorial** section, learn advanced quiz techniques including custom quiz grading. In the **Reference** section, view a complete technical reference for each tag used for quizzes. In the **Downloads** section, download example code for each beginner and advanced tutorial. In the **FAQs** section, read answers to frequently asked questions.

Beginner Quiz Tutorials

Advanced Quiz Tutorials

Quiz MGI Tag Reference

Quiz Downloads

- B1. Download a composite quiz example (B1Composite.zip - 1.5 KB)
- A1. Download a custom quiz example (A1Custom.zip - 2.7 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Creating, Grading and Managing Composite Quizzes

Introduction

The mgiQuiz tag can be used with two types of quizzes, Composite and Custom.

Composite quizzes are created manually or randomly from questions entered in the admin mode of the mgiQuiz tag and stored in the mgiQuiz database. Composite quizzes are graded by the mgiQuiz tag and quiz results are stored in the mgiQuiz database for later management.

Custom quizzes are created as "hard-coded" pages with text, HTML, and form elements. Custom quizzes can be graded by the mgiQuiz tag, but results are not tracked or stored. Combining the mgiQuiz tag with a database submit or file write will allow you to store quiz grades.

In this example, you will create a math quiz from several different math topics.

MGI Tags

- mgiQuiz

Steps

1. Create a quiz administration page and open it in a text editor.
2. Insert the mgiQuiz tag in Admin mode.
3. Save the quiz administration page.
4. FTP the quiz administration page to the web server running MGI.
5. View the quiz administration page in a web browser.
6. Create quiz categories.
7. Add quiz questions to quiz categories.
8. Create a composite quiz.
9. Print the quiz form.
10. Create the quiz page and open it in a text editor.

11. File include the printed quiz form.
12. Save the quiz page.
13. Create a quiz grading page and open it in a text editor.
14. Insert the mgiQuiz tag in Grade mode.
15. Save the quiz grading page.
16. FTP the quiz page and quiz grading page to the web server running MGI.
17. View the quiz page in a web browser and take the quiz.
18. View the quiz administration page in a web browser to manage quiz grades.

Step 1: Create a quiz administration page and open it in a text editor.

Create a page named "quizadmin.mgi" to display the web-based quiz administration interface. Open the quiz administration page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Insert the mgiQuiz tag in Admin mode.

Insert your cursor after the beginning <BODY> element and enter the mgiQuiz tag and mode parameter. In the mode parameter, enter "Admin".

```
<mgiQuiz mode="Admin">
```

Step 3: Save the quiz administration page.

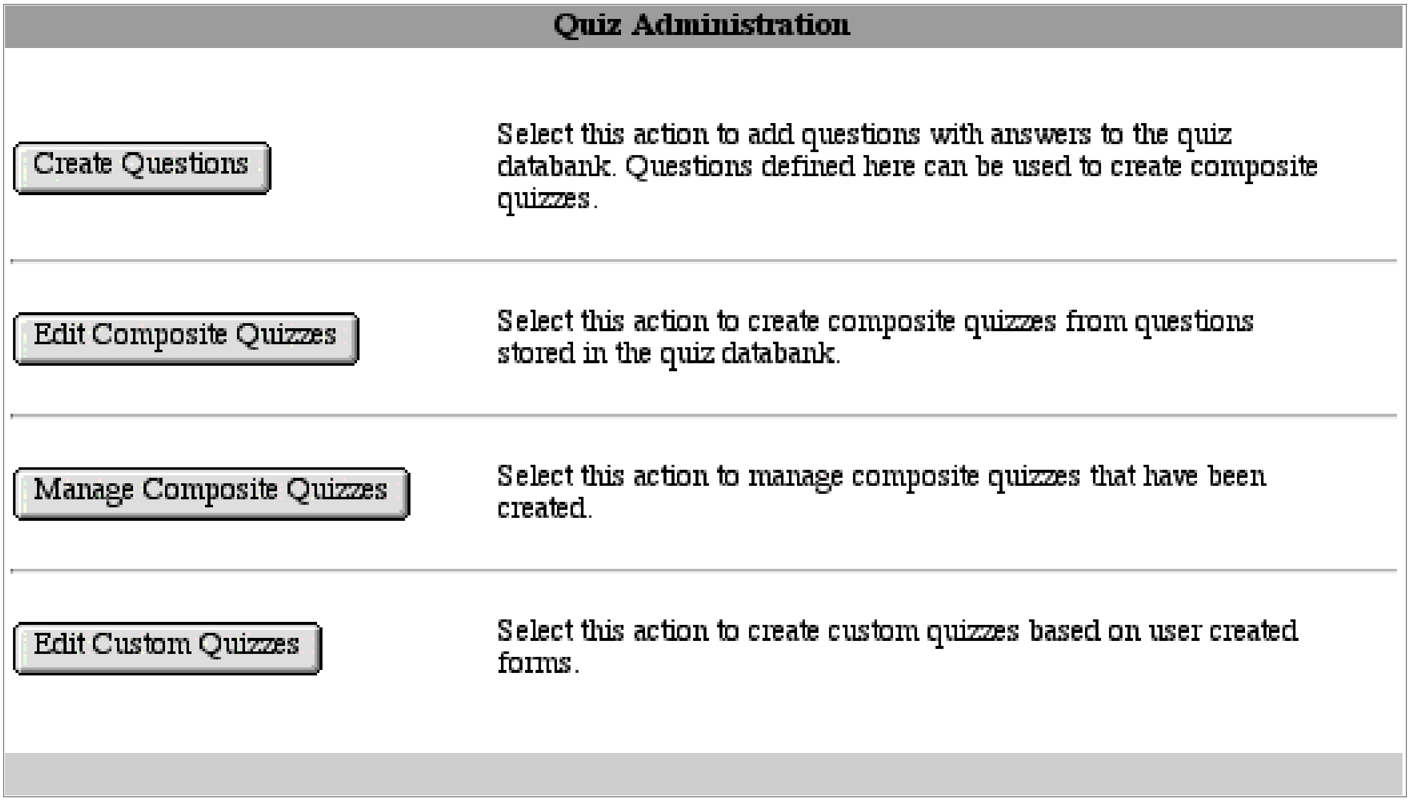
Save the changes you have made to the quiz administration page.

Step 4: FTP the quiz administration page to the web server running MGI.

Upload the quiz administration page from your local computer to the web server using an FTP program.

Step 5: View the quiz administration page in a web browser.

View the quiz administration page in a web browser. The first screen of the web-based, administration interface displays four options: 1) Create Questions 2) Edit Composite Quizzes 3) Manage Composite Quizzes and 4) Edit Custom Quizzes.



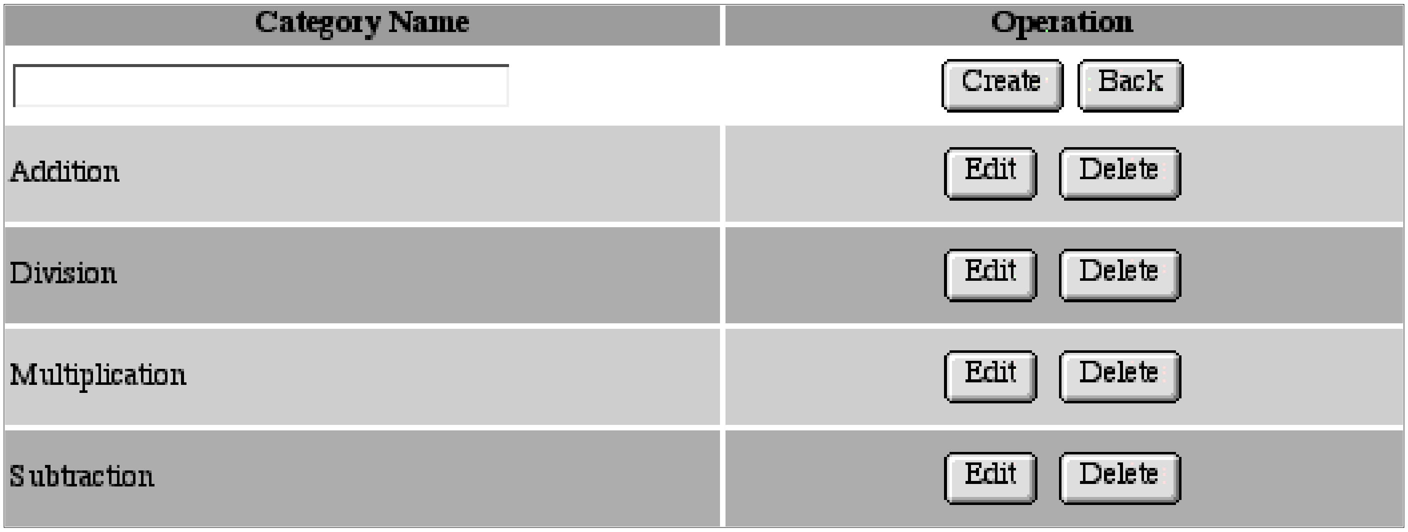
Quiz Administration

Create Questions	Select this action to add questions with answers to the quiz databank. Questions defined here can be used to create composite quizzes.
Edit Composite Quizzes	Select this action to create composite quizzes from questions stored in the quiz databank.
Manage Composite Quizzes	Select this action to manage composite quizzes that have been created.
Edit Custom Quizzes	Select this action to create custom quizzes based on user created forms.

Step 6: Create quiz categories.

Click the "Create Questions" button to access the Quiz Categories. Create new categories of questions. A category can consist of any grouping of questions such as subjects or topics within a subject.

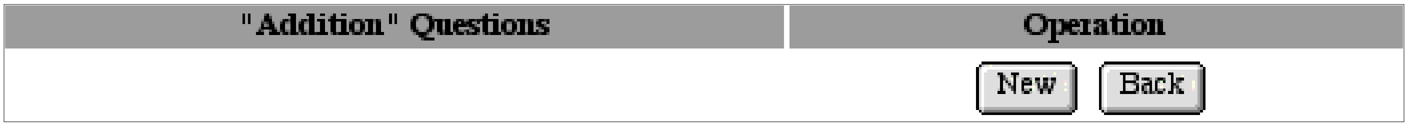
To create a new category, enter the category name under the "Category Name" column and click the "Create" button. For this example, create four categories of math questions: "Addition", "Subtraction", "Multiplication" and "Division".



Category Name	Operation
	Create Back
Addition	Edit Delete
Division	Edit Delete
Multiplication	Edit Delete
Subtraction	Edit Delete

Step 7: Add quiz questions to quiz categories.

Populate each quiz category with specific quiz questions. Click the "Edit" button beside the category name to display the question management interface for that category.



"Addition" Questions	Operation

Click the "New" button to enter a new question, answers to that question, the correct answer, the question's difficulty, and an image location for any picture associated with the question.

In the "Question" field, type the quiz question. For multiple choice questions, enter all possible answers (correct and incorrect) in the "Answer" fields. By default, there are four Answer fields: A - D. To delete an extraneous answer field, click the "Remove Answer" button. To add additional answer fields, click the "Add Answer" button. Questions with more than 8 answers will automatically display as a pop-up menu rather than as radio buttons.

For a fill-in-the-blank question, remove all but one answer field and enter the correct answer in the "Answer A" field. Questions will only one answer will be automatically displayed as a fill-in-the-blank item.

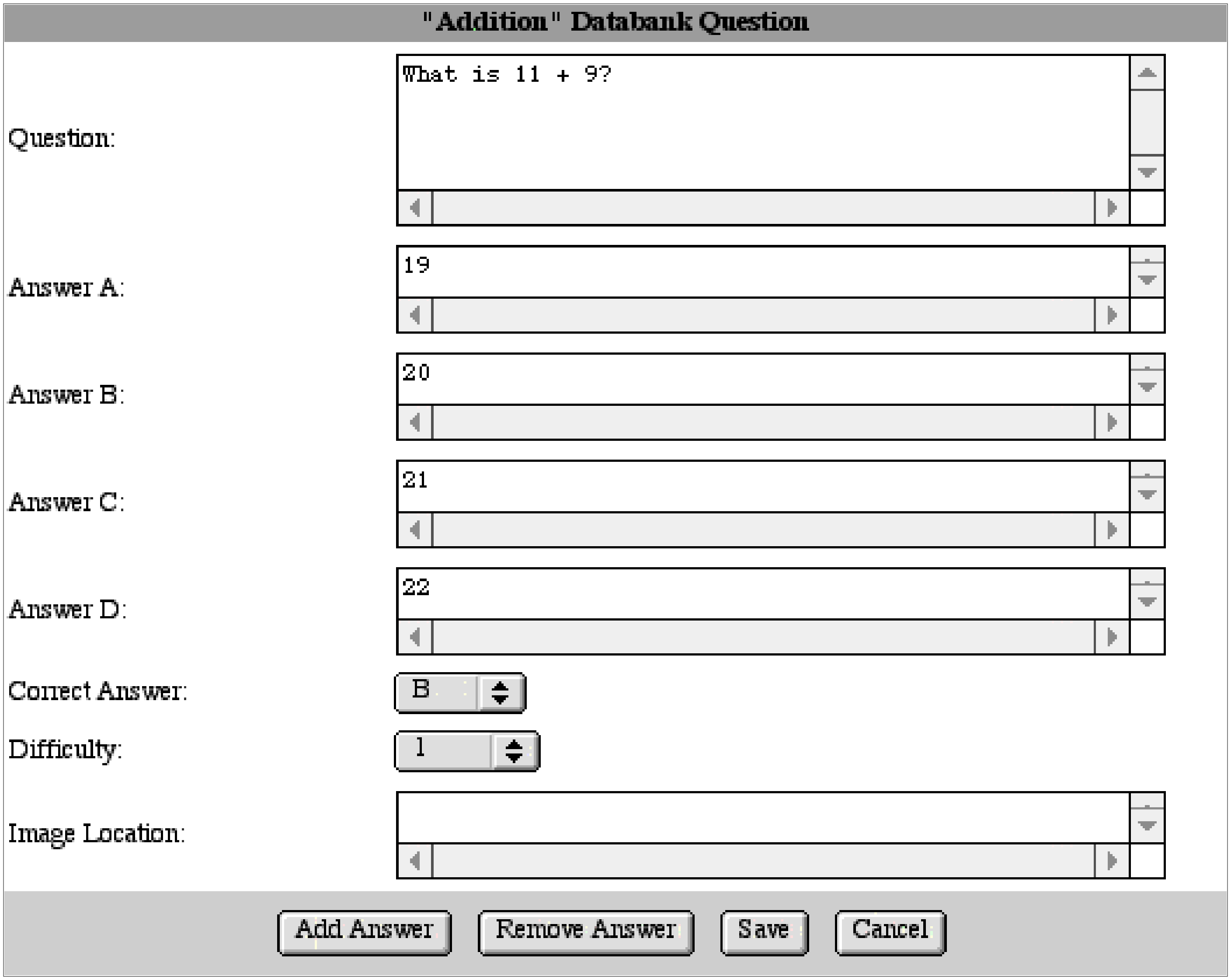
Select the correct answer for the question in the "Correct Answer" pop-up menu. The selected correct answer is used to grade quizzes.

Optionally, select the question difficulty in the "Difficulty" pop-up menu. When creating a composite quiz, you will have the option of creating a quiz with a minimum difficulty.

If you wish to display an image for the question, enter the location of the image relative to the quiz administration page or as an absolute URL.

Finally, click the "Save" button to add the question to the quiz database and return to the question management interface. When you complete the question additions to one category, click the "Back" button under the "Operation" column to select another category.

For this example, add quiz questions to the Addition, Subtraction, Multiplication and Division categories. Click the "Back" button on the category interface to return to the quiz administration main menu.



"Addition" Databank Question

Question: What is 11 + 9?

Answer A: 19

Answer B: 20

Answer C: 21

Answer D: 22

Correct Answer: B

Difficulty: 1

Image Location:

Step 8: Create a composite quiz.

From the main quiz administration menu, click the "Edit Composite Quizzes" button to create a new composite quiz or edit an existing composite quiz.

To create a new quiz, enter the quiz name under the "Composite Quiz Name" column and click the "Create" button. For this example, create a quiz named "Math Final". Quiz names are case-sensitive.

Composite quizzes may be compiled manually via the "Edit" button and/or randomly via the "Generate" button.

To manually add quiz questions, click the "Edit" button beside the quiz name. To add a new question:

1. Click the "Add" button.
2. Click the "Select" button beside the category containing the question you wish to add.

3. Click the "View Question" button beside the question you wish to add.
4. Enter the question number (the next available question number is entered automatically).
5. Enter the point value for the question.
6. Click the "Add to Quiz" button.
7. Add additional questions from the selected category or click the "Back" button to choose another category
8. When all questions have been added, click the "Back" button to view and/or edit the composite quiz questions.

   To create an entire quiz randomly OR to randomly add quiz questions to an existing quiz, click the "Generate" button beside the quiz name.

   Enter the number of questions to add in the "Question Count" field. Enter the point value for each question in the "Question Point Value" field (Specific point values may be edited after the quiz is generated). Select the category of the questions. To select questions from all categories choose "All". If you quiz questions have multiple difficulties, optionally select a minimum and/or a maximium difficulty.

   To create an entirely new quiz with the specified criteria, click the "Generate" button. Clicking the "Generate" button for a quiz with existing questions will overwrite all existing questions with the new randomly generated questions. To add random questions to an existing quiz, click the "Populate" button.

   To review or modify the quiz questions, modify the question order or modify the question point values, click the "Edit" button beside the quiz name. To modify the question order, enter a new number under the "Question #" column and click the "Update" button beside the quiz question. When the quiz is "printed", all questions will be ordered numerically by the question number.

   To modify the question's point value, enter a new point value under the "Point Value" column and click the "Update" button beside the quiz question.

   To review the quiz question, answers and details, click the "View" button beside the quiz question.

   To delete a quiz question, click the "Delete" button beside the quiz question.

   To add a new quiz question, use the manual or random methods described above.

Step 9: Print the quiz form.

The "printing" function of the mgiQuiz tag does not actually print your quiz on your printer. Rather, the "print" function creates the table, text and other HTML of the quiz form and saves the quiz as a file on the web server.

Click the "Print" button to select the preferences for the quiz form. Select the "Single Column Layout" radio button to save quiz questions in one column. Select the "Double Column Layout" radio button to save quiz questions in two columns.

Select the "Name" or "ID" optional input checkboxes to include a text field for the quiz taker's name or identification. Select the "Start Date", "End Date", "Start Time", and "End Time" checkboxes to track the date and time each quiz is taken.

Select the hours, minutes, and seconds beside "Time Limit" to time each quiz.

Enter the name of the quiz file and the path to save the quiz file in the "Output File Name". The quiz name with the .txt suffix is entered by default. The quiz file is saved at the same location in the file system as the quiz administration page by default. To save the quiz file at a different level of the file system, enter the file path relative to the quiz administration page before the quiz file name (e.g., "Folder/Quiz.txt" or "../Folder/Quiz.txt"). For this example, name the quiz "Math Final.txt"

Click the "Print" button to save the quiz form to the web server. Once a quiz is "printed", re-printing a quiz will overwrite the quiz form and delete any existing grade information from the quiz database.

Step 10: Create the quiz page and open it in a text editor.

Create a quiz page to include the "printed" quiz form named "quiz.mgi" and open the quiz page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 11: File include the printed quiz form.

On the quiz page, file include the printed quiz form. Enter the mgiIncludeFile tag and fileLocation parameter. In the fileLocation parameter, enter the relative path to the printed quiz form (e.g., "Math Final.txt").

After the file include, enter a submit button to post the quiz form to the grading page.

Enclose the file include and submit button with HTML <FORM> tags and post the form to the quiz grading page (grade.mgi).

```
<form action="grade.mgi" method="post">

<mgiIncludeFile fileLocation="Math Final.txt">

<mgiButton name="submit" value="Submit Quiz">

</form>
```

Step 12: Save the quiz page.

Save the changes you have made to the quiz page.

Step 13: Create a quiz grading page and open it in a text editor.

Create a quiz grading page named "grade.mgi" to grade the quiz and open the quiz grading page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 14: Insert the mgiQuiz tag in Grade mode.

On the quiz grading page, enter the mgiQuiz tag and name parameter. In the name parameter, enter the name of the quiz to grade (e.g., "Math Final").

```
Your quiz grade is <mgiQuiz name="Math Final">.
```

Step 15: Save the quiz grading page.

Save the changes you have made to the quiz grading page.

Step 16: FTP the quiz page and quiz grading page to the web server running MGI.

Upload the quiz page and quiz grading page from your local computer to the web server using an FTP program.

Step 17: View the quiz page in a web browser and take the quiz.

View the quiz page in a web browser. Complete the quiz and click the "submit" button. Your grade appears on the quiz grading page and your quiz information is added to the quiz database.

Step 18: View the quiz administration page in a web browser to manage quiz grades.

Vew the quiz administration page (quizadmin.mgi) in a web browser and click the "Manage Composite Quizzes" button to view or export quiz grades. Click the "View

Grades" button to view the name and score for each quiz. Click the "View Details" button to view the details for each quiz including the name, identification, time lapse, timeout, grade and answers.

To export the quiz grades, click the "Export" button. Select the quiz information to export by checking "Export Options". Enter the file name of the quiz export file in the "Export File Name" field. To save the quiz export file at a different level of the file system, enter the file path relative to the quiz administration page before the quiz export file name (e.g., "Folder/QuizExport.txt" or "../Folder/QuizExport.txt"). For this example, name the export "Quiz Export.txt". Quiz details are exported as tab-delimited files with field name headers.

[Return to the Administering Online Quizzes Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Creating and Grading Custom Quizzes

Introduction

The mgiQuiz tag can be used with two types of quizzes, Composite and Custom.

Composite quizzes are created manually or randomly from questions entered in the admin mode of the mgiQuiz tag and stored in the mgiQuiz database. Composite quizzes are graded by the mgiQuiz tag and quiz results are stored in the mgiQuiz database for later management.

Custom quizzes are created as "hard-coded" pages with text, HTML, and form elements. Custom quizzes can be graded by the mgiQuiz tag, but results are not tracked or stored. Combining the mgiQuiz tag with a database submit or file write will allow you to store quiz grades and other information for later retrieval.

In this example, you will create a custom nutrition quiz and collect quiz scores and answers in a text file.

MGI Tags

- mgiQuiz

Steps

1. Create a custom quiz form page.
2. Create a quiz administration page and open it in a text editor.
3. Insert the mgiQuiz tag in Admin mode.
4. Save the quiz administration page.
5. FTP the administration page to the web server running MGI.
6. View the quiz administration page in a web browser.
7. Add a new quiz.
8. Add quiz post arguments and answers.
9. Create a quiz grading page and open it in a text editor.
10. Insert the mgiQuiz tag in Grade mode.
11. Save the quiz grading page.
12. FTP the quiz form and quiz grading page to the web server running MGI.
13. View the quiz form page in a web browser and take the quiz.

Step 1: Create a custom quiz form page.

Create a custom quiz form named "nutrition.mgi" using any type of form element (radio buttons, checkboxes, selects, text fields, or text areas). Name each form element uniquely. For radio buttons, name the set of radio buttons with one name and enter a unique value for each radio button. For checkboxes, name **each checkbox** uniquely and enter a value for each checkbox. For selects (pup-up menus), name the pop-up menu uniquely and enter a value for each option. For text fields and text areas, name each uniquely. Text boxes can be used for fill-in-the-blank questions, but we recommend text fields for that purpose.

After the form elements (questions and answers), enter a submit button named "Submit Quiz".

Enclose all form elements with HTML <FORM> tags and post the form to the quiz grading page (results.mgi).

The following is an example quiz form with different form elements, a submit button and HTML <FORM> tags.

```
<FORM ACTION="results.mgi" METHOD="POST">

<H1><CENTER>Nutrition Quiz</CENTER></H1>

<H4><CENTER>How is your nutrition IQ?
Answer these questions to find out.</CENTER></H4>

<H1><CENTER>
<TABLE BORDER="1" CELLSPACING="0" CELLPADDING="3">

<TR>
<TD BGCOLOR="#99ccff">1. How many calories
are in 1 gram of fat?</TD>
</TR>
<TR>
<TD><INPUT NAME="calorie" TYPE="text" SIZE="5"
MAXLENGTH="1">
</TD>
</TR>

<TR>
<TD BGCOLOR="#99ccff">2. Which food is the best
source of protein?</TD>
</TR>
<TR>
<TD>
```

```
<P><INPUT TYPE="radio" VALUE="carrot" NAME="protein">
carrot</P>
<P><INPUT TYPE="radio" VALUE="sweet potato" NAME="protein">
sweet potato</P>
<P><INPUT TYPE="radio" VALUE="ground beef" NAME="protein">
ground beef</P>
<P><INPUT TYPE="radio" VALUE="butter" NAME="protein">
butter</P>
<P><INPUT TYPE="radio" VALUE="NO ANSWER" NAME="protein"
CHECKED="1"> NO
ANSWER
</TD>
</TR>

<TR>
<TD BGCOLOR="#99ccff">3. Choose 2 foods that would
help you meet your daily fruit and vegetable
requirement:</TD>
</TR>
<TR>
<TD>
<P><INPUT TYPE="checkbox" NAME="vegetableA"
VALUE="banana"> banana</P>
<P><INPUT TYPE="checkbox" NAME="vegetableB"
VALUE="roast beef"> roast beef</P>
<P><INPUT TYPE="checkbox" NAME="vegetableC"
VALUE="cheese"> cheese</P>
<P><INPUT TYPE="checkbox" NAME="vegetableD"
VALUE="waffles"> waffles</P>
<P><INPUT TYPE="checkbox" NAME="vegetableE"
VALUE="peas"> peas</P>
<P><INPUT TYPE="checkbox" NAME="vegetableF"
VALUE="tea"> tea
</TD>
</TR>

<TR>
<TD BGCOLOR="#99ccff">4. What is the recommended intake
of water per day?</TD>
</TR>
<TR>
<TD><SELECT NAME="water">
<OPTION VALUE=" " SELECTED>Choose One
<OPTION VALUE="1">1 - 8 ounce glass
<OPTION VALUE="2">2 - 8 ounce glasses
```

```
<OPTION VALUE="8">8 - 8 ounce glasses
<OPTION VALUE="20">20 - 8 ounce glasses
</SELECT></TD>
</TR>

<TR>
<TD BGCOLOR="#99ccff">5. Which of the following
practices does <B>NOT</B> promote good health?</TD>
</TR>
<TR>
<TD>
<P><INPUT TYPE="radio" VALUE="decrease salt"
NAME="health"> Lowering your intake of salt</P>
<P><INPUT TYPE="radio" VALUE="increase caffeine"
NAME="health"> Increasing your intake of caffeine</P>
<P><INPUT TYPE="radio" VALUE="decrease fat"
NAME="health"> Lowering your intake of saturated fat</P>
<P><INPUT TYPE="radio" VALUE="increase exercise"
NAME="health"> Increasing your exercise</P>
<P><INPUT TYPE="radio" VALUE="NO ANSWER"
NAME="health" CHECKED="1"> NO ANSWER
</TD>
</TR>

</TABLE></CENTER></H1>

<P><CENTER><INPUT NAME="name" TYPE="submit"
VALUE="Submit Quiz">
</CENTER></P>

</FORM>
```

Step 2: Create a quiz administration page and open it in a text editor.

Create a page named "quizadmin.mgi" to display the web-based quiz administration interface. Open the quiz administration page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Insert the mgiQuiz tag in Admin mode.

Insert your cursor after the beginning <BODY> element and enter the mgiQuiz tag and mode parameter. In the mode parameter, enter "Admin".

```
<mgiQuiz mode="Admin">
```

Step 4: Save the quiz administration page.

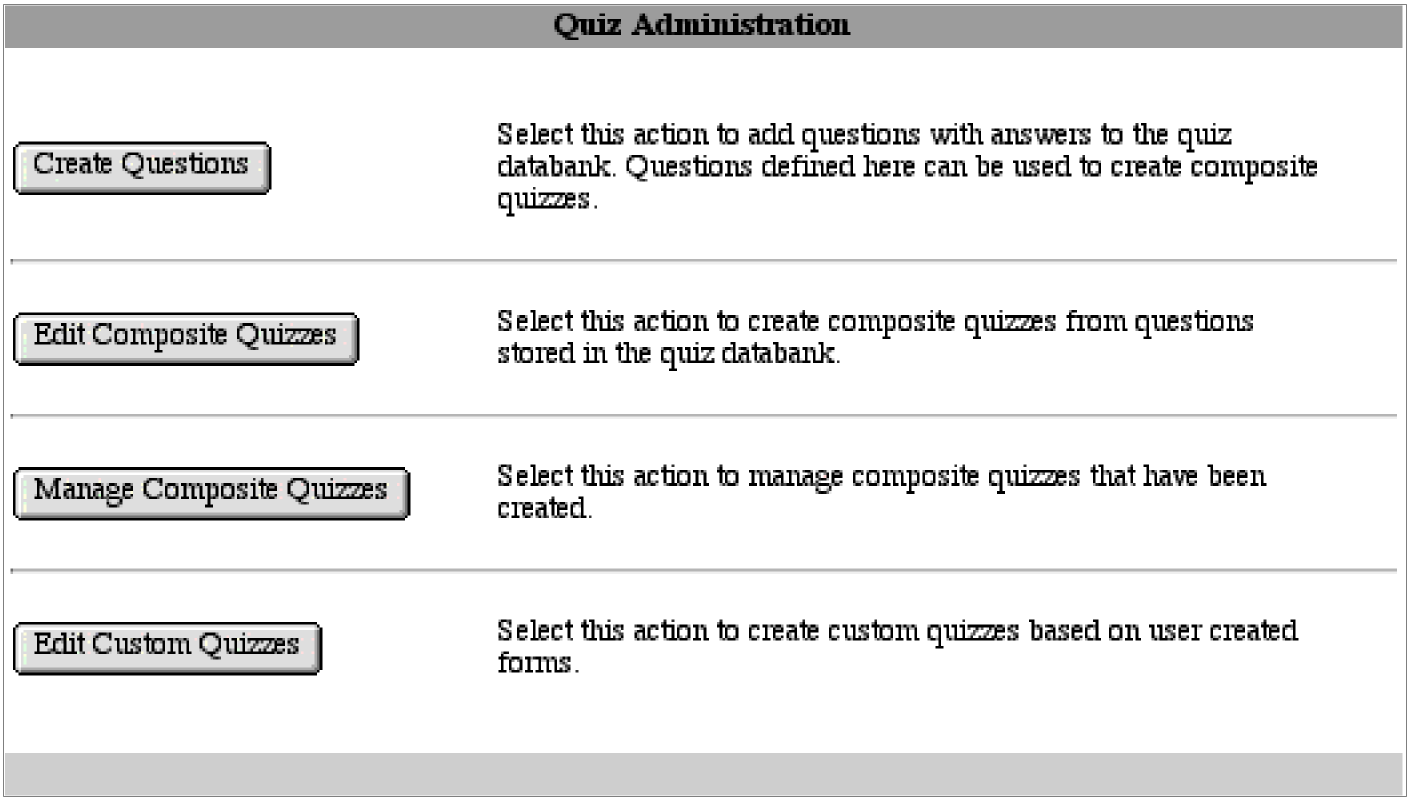
Save the changes you have made to the quiz administration page.

Step 5: FTP the quiz administration page to the web server running MGI.

Upload the quiz administration page from your local computer to the web server using an FTP program.

Step 6: View the quiz administration page in a web browser.

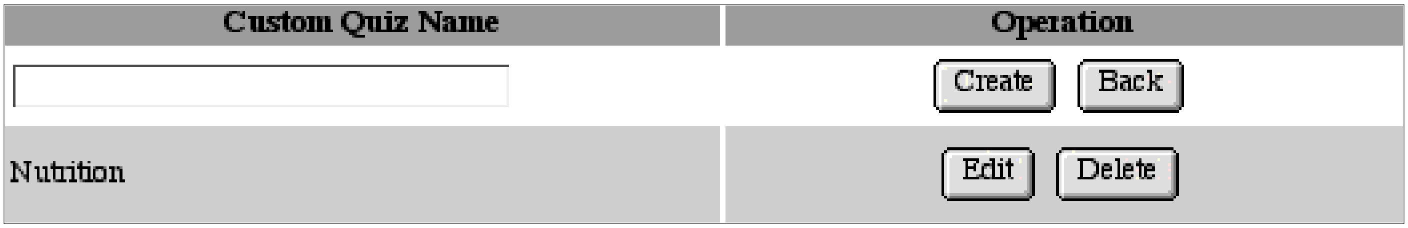
View the quiz administration page in a web browser. The first screen of the web-based, administration interface displays four options: 1) Create Questions 2) Edit Composite Quizzes 3) Manage Composite Quizzes and 4) Edit Custom Quizzes.



Quiz Administration	
Create Questions	Select this action to add questions with answers to the quiz databank. Questions defined here can be used to create composite quizzes.
Edit Composite Quizzes	Select this action to create composite quizzes from questions stored in the quiz databank.
Manage Composite Quizzes	Select this action to manage composite quizzes that have been created.
Edit Custom Quizzes	Select this action to create custom quizzes based on user created forms.

Step 7: Add a new quiz.

Click the "Edit Custom Quizzes" button. In the text field under the "Custom Quiz Name" column, enter the case-sensitive quiz name ("Nutrition" for this example). and click the "Create" button.



Custom Quiz Name	Operation
	Create Back
Nutrition	Edit Delete

Step 8: Add quiz post arguments and answers.

Click the "Edit" button beside the quiz to access the post argument interface. For each quiz question including each checkbox (i.e., for each unique post argument) add a question number, post argument name, correct answer, and points.

In the text field under the "Question #" column, enter the question number. Question numbers are entered automatically based on the previous question number. For checkboxes (i.e., questions with multiple answers) we suggest that you use a suffix such as 3.1, 3.2, etc.

In the text field under the "Post Argument Name" column, enter the post argument name to grade. Enter one name for each text field, one name for each text area, one name for each **set** of radio buttons, one name for **each** checkbox and one name for each select (pop-up menu).

In the text field under the "Answer" column, enter the value for the correct answer. The value is not necessarily the same as the answer appearring on the quiz form since radio buttons, checkboxes and selects (pop-up menus) can have post argument values that differ from the option description.

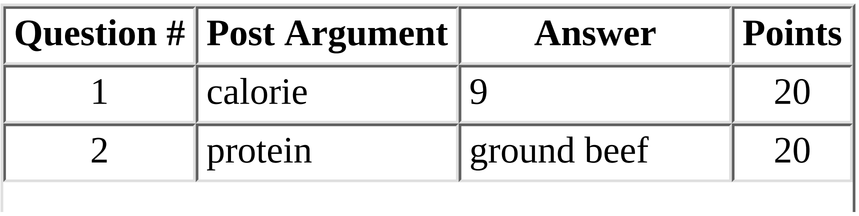
In the text field under the "Points" column, enter the number of points gained or lost for a value which matches the answer. For selecting a correct answer, enter positive point values. For selecting an incorrect answer (e.g., selecting an incorrect checkbox) you may enter a negative number to subtract points.

Finally, click the "Add" button under the quiz name to add the post argument information.

To edit a post argument name, answer or points once they have been created, modify the name, answer or point value and click the "Update" button beside the post argument.

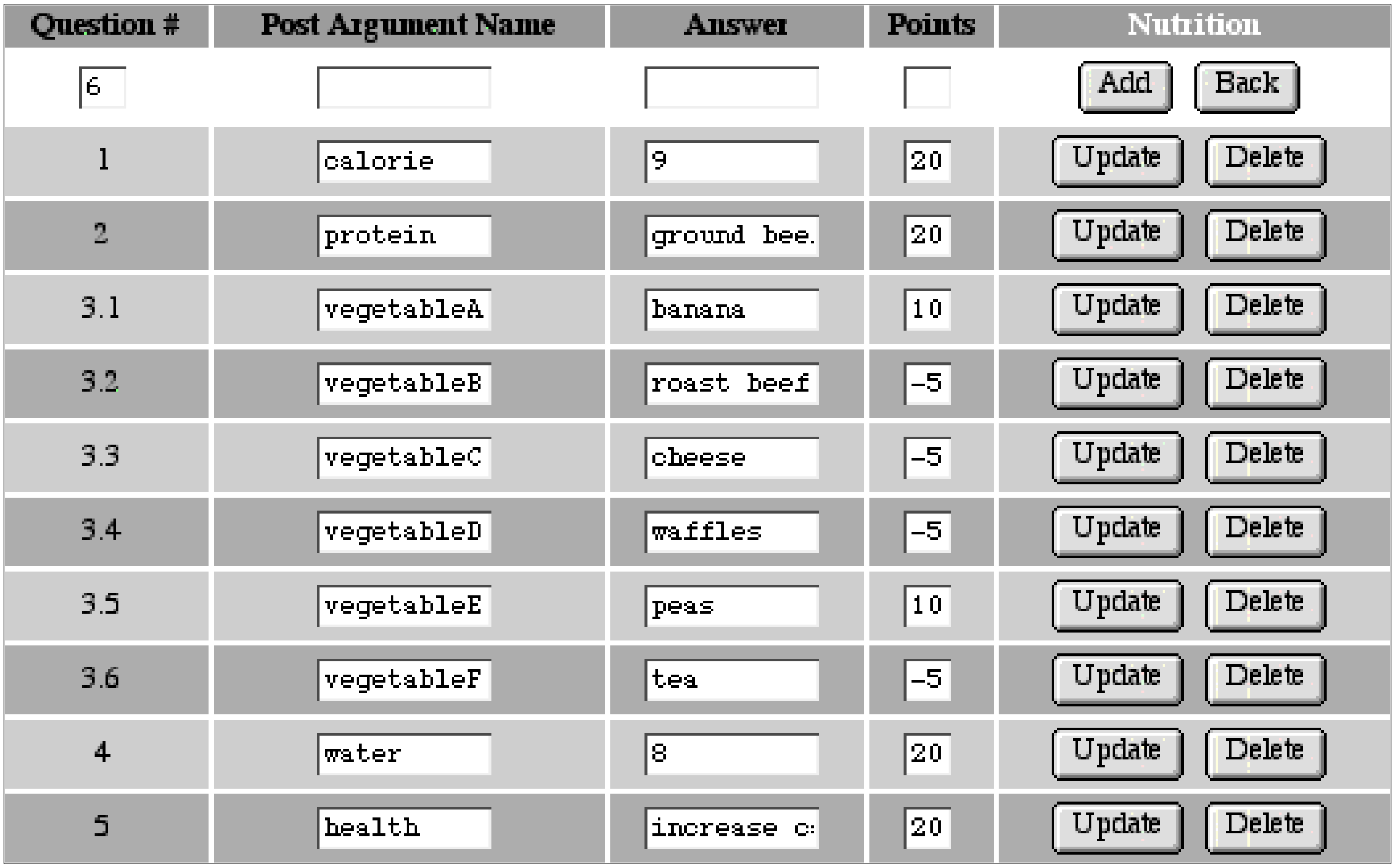
For this example, enter the following post argument information. In this example, notice that each checkbox is included as a sub-set of one question and one loses points for selecting an incorrect checkbox answer.

Nutrition Quiz Post Arguments:



Question #	Post Argument	Answer	Points
1	calorie	9	20
2	protein	ground beef	20

3.1	vegetableA	banana	10
3.2	vegetableB	roast beef	-5
3.3	vegetableC	cheese	-5
3.4	vegetableD	waffles	-5
3.5	vegetableE	peas	10
3.6	vegetableF	tea	-5
4	water	8	20
5	health	increase caffeine	20



Question #	Post Argument Name	Answer	Points	Nutrition
6				Add Back
1	calorie	9	20	Update Delete
2	protein	ground bee	20	Update Delete
3.1	vegetableA	banana	10	Update Delete
3.2	vegetableB	roast beef	-5	Update Delete
3.3	vegetableC	cheese	-5	Update Delete
3.4	vegetableD	waffles	-5	Update Delete
3.5	vegetableE	peas	10	Update Delete
3.6	vegetableF	tea	-5	Update Delete
4	water	8	20	Update Delete
5	health	increase c	20	Update Delete

Step 9: Create a quiz grading page and open it in a text editor.

Create a quiz grading page named "results.mgi" to grade the quiz and open the quiz grading page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 10: Insert the mgiQuiz tag in Grade mode.

On the quiz grading page, enter the mgiQuiz tag, name parameter and type parameter. In the name parameter, enter the name of the quiz to grade (e.g., "Nutrition"). In the type parameter, enter "Custom". For custom quizzes, the type parameter is required.

For this example, the quiz grade is stored in a variable. Based on the quiz grade, a text message is displayed. The quiz score and answers are then appended to a tab-delimited file using the mgiModifyFile tag.

```
<mgiSet name="Score">
<mgiQuiz name="Nutrition" type="Custom">
</mgiSet>

<mgiIf lhs={mgiGet name="Score"}
relationship="greaterThan" rhs="90">

<p>Excellent!  You have a very high nutrition IQ.
Keep up the good work.

<mgiElse>

<mgiIf lhs={mgiGet name="Score"} relationship="greaterThan"
rhs="80">

<p>Your nutrition IQ is good, but could use a little work.

<mgiElse>

<p>Your nutrition IQ is poor.  Read our nutrition fact
sheet to brush up on your nutrition information.

</mgiIf>

</mgiIf>

<mgiModifyFile fileLocation="scores.txt" mode="append">
<mgiGet name="Score"> <mgiPostArgument
name="calorie"> <mgiPostArgument name="protein">
<mgiPostArgument name="vegetableA">
<mgiPostArgument name="vegetableB">
<mgiPostArgument name="vegetableC">
<mgiPostArgument name="vegetableD">
<mgiPostArgument name="vegetableE">
<mgiPostArgument name="vegetableF">
<mgiPostArgument name="water">
<mgiPostArgument name="health">
</mgiModifyFile>
```

Step 11: Save the quiz grading page.

Save the changes you have made to the quiz grading page.

Step 12: FTP the quiz form and quiz grading page to the web server running MGI.

Upload the quiz form and quiz grading page from your local computer to the web server using an FTP program.

Step 13: View the quiz form page in a web browser and take the quiz.

View the quiz form page in a web browser. Complete the quiz and click the "submit" button. Based on your grade, a message displays and your score is saved in a text file with your answers.

[Return to the Administering Online Quizzes Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Embedding MGI Tags

In the **Beginner Tutorial** section, learn how to embed one MGI tag within another MGI tag. In the **Advanced Tutorial** section, learn advanced embedding techniques including multiple embedded tags using variables. In the **Reference** section, view a complete technical reference for each tag used for embedding. In the **Downloads** section, download example code for each beginner and advanced tutorial. In the **FAQs** section, read answers to frequently asked questions.

Beginner Embedding Tutorials

- B1. Embedding One MGI Tag in a Parameter

Advanced Embedding Tutorials

- A1. Embedding Multiple Tags in One Parameter

Embedding Downloads

- B1. Download an embedding example with one MGI tag in a parameter (B1Embed.zip - 4 KB)
- A1. Download an embedding example with multiple tags in one parameter (A1MultEmbed.zip - 4 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Embedding One MGI Tag in a Parameter

Introduction

MGI supports embedded tags. That is, you can use one MGI tag within the parameter value of another MGI tag or within the parameter value of an HTML tag. The embedded MGI tag replaces the entire value of the MGI or HTML parameter including the quotation marks. Embedding allows you to create "dynamic" functions that change depending on the information entered or calculated.

Only one MGI tag can be embedded within the parameter of another MGI tag unless you use variables for an extended embedded function.

In this example, the color from a path argument is embedded in the background color of a page and in the comparison value of a conditional statement to determine the text that displays.

MGI Tags

- Any MGI tag without a body can be embedded within the parameter value of another MGI or HTML tag.

Steps

1. Create a link page with path arguments.
2. Create a display page and open it in a text editor.
3. Embed the mgiPathArgument tag in the HTML Body tag and the mgiConditional tags.
4. Save the display page.
5. FTP the link page and display page to a web server running MGI.
6. View the link page in a browser and click a link.

Step 1: Create a link page with path arguments.

Create a page named "link.mgi" that contains text links to the display page (display.mgi). For each link, add a path argument with a background color. Name the path arguments"color".

This is an example link page.

```
<H2><CENTER>Backgrounds and Text</CENTER></H2>

<P>Choose a background color to test. The appropriate
```

```
text color will be shown.</P>

<OL>
  <LI><a href="display.mgi?color=white">
View a white background.</a>
  <LI><a href="display.mgi?color=black">
View a black background.</a>
  <LI><a href="display.mgi?color=red">
View a red background.</a>
  <LI><a href="display.mgi?color=green">
View a green background.</a>
  <LI><a href="display.mgi?color=orange">
View an orange background.</a>
</OL>
```

Step 2: Create a display page and open it in a text editor.

Create a page named "display.mgi" to display the chosen color and the text that is most appropriate for that color background. Open the display page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Embed the mgiPathArgument tag in the HTML Body tag and the mgiConditional tags.

MGI tags are embedded the same in HTML and in the parameter of other MGI tags. To embed a tag, replace the quotation marks and value of the parameter with braces, the tag name and the tag's parameters. In this example, replace the quotation marks and value of the bgcolor parameter (in the body tag) with braces, the mgiPathArgument tag and the name parameter. Also replace the quotation marks and value of the lhs parameter (in the mgiConditional tag) with braces, the mgiPathArgument tag and the name parameter.

This is an example display page.

```
<HTML>
<HEAD>
  <TITLE>Display</TITLE>
</HEAD>
<BODY BGCOLOR={mgiPathArgument name="color"}>

<CENTER>
<P>

<mgiIf lhs={mgiPathArgument name="color"}
relationship="equals" rhs="white">
```

```
The best text color for a white background is black.
</mgiIf>

<mgiIf lhs={mgiPathArgument name="color"}
relationship="equals" rhs="black">
<font color="white">The best text color for a
black background is white.</font>
</mgiIf>

<mgiIf lhs={mgiPathArgument name="color"}
relationship="equals" rhs="red">
The best text color for a red background is black.
</mgiIf>

<mgiIf lhs={mgiPathArgument name="color"}
relationship="equals" rhs="green">
<font color="white">The best text color for a
reen background is white.</font>
</mgiIf>

<mgiIf lhs={mgiPathArgument name="color"}
relationship="equals" rhs="orange">
The best text color for an orange background
is black.
</mgiIf>

</P>
</CENTER>
</BODY>
</HTML>
```

Step 4: Save the display page.

Save the changes you have made to the display page.

Step 5: FTP the link page and display page to a web server running MGI.

Upload the link page and display page from your local computer to the web server using an FTP program.

Step 6: View the link page in a browser and click a link.

View the link page in a web browser and click a link. The background color and text comment on the display page change according to the color path argument coded in the link you have chosen.

Embedding Multiple Tags in One Parameter

Introduction

MGI supports embedded tags. That is, you can use one MGI tag within the parameter value of another MGI tag or within the parameter value of an HTML tag. The embedded MGI tag replaces the entire value of the MGI or HTML parameter including the quotation marks. Embedding allows you to create "dynamic" functions that change depending on the information entered or calculated.

Only one MGI tag can be embedded within the parameter of another MGI tag unless you use variables for an extended embedded function.

In this example, the subject of a referral email is constructed from the visitor's name and message.

MGI Tags

- Any MGI tag may be combined in variables and embedded in the parameter value of another MGI or HTML tag using the mgiGet tag.

Steps

1. Create a referral form.
2. Create a referral processing page and open it in a text editor.
3. Construct the subject in a variable and embed the variable in the mgiSendMail tag.
4. Save the referral processing page.
5. FTP the referral form and referral processing page to a web server running MGI.
6. View the referral form in a web browser and submit the referral.

Step 1: Create a referral form.

Create a referral page named "referral.mgi" with form elements. In this example, the referral form consists of form fields for your name, your email, subject, message, recipient name, and recipient email. Name each form element uniquely. Enclose all form elements with HTML <FORM> tags and post the form to the referral processing page (referralprocess.mgi).

This is an example referral form.

```
<FORM action="referralprocess.mgi" method="post">
```

```
<H2><CENTER>Refer Our Site</CENTER></H2>

<P><CENTER>Refer our site to a friend.
Complete the form below and click
&quot;Send Now&quot;.</CENTER></P>

<P><CENTER>
<TABLE BORDER="0" CELLSPACING="3" CELLPADDING="0">
  <TR>
    <TD ALIGN="RIGHT">Your Name:</TD>
    <TD>&nbsp;
<INPUT NAME="SenderName" TYPE="text" SIZE="30"></TD>
  </TR>
  <TR>
    <TD ALIGN="RIGHT">Your Email:</TD>
    <TD>&nbsp;
<INPUT NAME="SenderEmail" TYPE="text" SIZE="30"></TD>
  </TR>
  <TR>
    <TD ALIGN="RIGHT">Friend's Name:</TD>
    <TD>&nbsp;
<INPUT NAME="RecipientName" TYPE="text" SIZE="30"></TD>
  </TR>
  <TR>
    <TD ALIGN="RIGHT">Friend's Email:</TD>
    <TD>&nbsp;
<INPUT NAME="RecipientEmail" TYPE="text" SIZE="30"></TD>
  </TR>
  <TR>
    <TD ALIGN="RIGHT">Subject:</TD>
    <TD>&nbsp;
<INPUT NAME="Subject" TYPE="text" SIZE="30"></TD>
  </TR>
  <TR>
    <TD ALIGN="RIGHT">Message:</TD>
    <TD>&nbsp;
<TEXTAREA NAME="Message" ROWS="7" COLS="27">
</TEXTAREA></TD>
  </TR>
  <TR>
    <TD>&nbsp;</TD>
    <TD><P><CENTER>
<INPUT TYPE="submit" VALUE="Send Now"></CENTER></TD>
  </TR>
</TABLE></CENTER>
```

```
</FORM>
```

Step 2: Create a referral processing page and open it in a text editor.

Create a page named "referralprocess.mgi" to send the referral email to the recipient. Open the referral processing page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Construct the subject in a variable and embed the variable in the mgiSendMail tag.

First, construct the subject of the email within the body of a page variable. Insert the beginning mgiSet tag, name parameter, and ending mgiSet tag. In the name parameter, enter "FullSubject". In the body of the mgiSet tags, enter mgiPostArgument tags for the sender's name and subject.

Second, enter the mgiSendMail tag, to parameter, from parameter, replyTo parameter, mailserver parameter, and subject parameter. Embed the recipient's email address in the "to" parameter. Enter the general info email address in the "from" parameter. Embed the sender's email address in the "replyTo" parameter. Enter the outgoing SMTP mailserver in the "mailServer" parameter. Embed the "FullSubject" page variable in the "subject:" parameter. In the body of the mgiSendMail tags, enter mgiPostArgument tags for the recipient's name, sender's name and sender's message.

This is an example referral processing page.

```
<H2><CENTER>Referral Submission</CENTER></H2>

<P><CENTER>Your referral has been sent to
<mgiPostArgument name="RecipientName">
</CENTER></P>

<mgiSet name="FullSubject">
<mgiPostArgument name="SenderName"> says
<mgiPostArgument name="Subject">
</mgiSet>

<mgiSendMail to={mgiPostArgument name="RecipientEmail"}
from="info@domain.com"
replyTo={mgiPostArgument name="RecipientEmail"}
mailServer="mail.domain.com"
subject={mgiGet name="FullSubject"}>
Dear <mgiPostArgument name="RecipientName">,
<mgiPostArgument name="SenderName"> thought you might
```

```
be interested in our web site.
<mgiPostArgument name="Message">
</mgiSendMail>
```

Step 4: Save the referral processing page.

Save the changes you have made to the referral processing page.

Step 5: FTP the referral form and referral processing page to a web server running MGI.

Upload the referral form and referral processing page from your local computer to the web server using an FTP program.

Step 6: View the referral form in a web browser and submit the referral.

View the referral form in a web browser. Complete and submit the form. The recipient is sent an email that appears with the following format. The subject of the email in this example is "Mark Willis says What Prices!".

```
Dear Andrea Shuster,

Mark Willis thought you might be interested
in our web site.

Hey Andrea, check out this site.  They have
some great holiday deals.  Buy me something
nice :)
```

[Return to the Embedding Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Using Variables

In the **Beginner Tutorial** section, learn how to set and display temporary page variables and database-driven site variables. In the **Reference** section, view a complete technical reference for each tag used for variables. In the **Downloads** section, download example code for each beginner and advanced tutorial. In the **FAQs** section, read answers to frequently asked questions.

Beginner Variable Tutorials

- B1. Setting and Displaying Page Variables
- B2. Setting and Displaying Site Variables

Variable MGI Tag Reference

- mgiGet
- mgiSet

Variable Downloads

- B1. Download a page variable example (B1PageVar.zip - 4 KB)
- B2. Download a site variable example (B2SiteVar.zip - 4 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Setting and Displaying Page Variables

Introduction

Variables are containers for information such as text, HTML and MGI tags. Variables allow you to "hold" information until you need to display it further down the page or on another page of the site. The information in page variables is available from the point at which they are set until they are overwriiten or until the end of the page that is currently processing, whichever occurs first. The information in site variables is stored in a database and is available from the point at which they are set until they are overwritten by a new value or deleted.

More often than not, the value of a variable is embedded in the parameter of another MGI tag. Read more about embedding.

In this example, a page variable is used to concatenate and hold the subject of a confirmation email from a customer's form submission. The subject is then embedded in the confirmation mgiSendMail tag.

MGI Tags

- mgiGet
- mgiSet

Steps

1. Create an information request form.
2. Create a send mail page for the information request and open it in a text editor.
3. Insert the mgiSet tag, mgiPostArgument tag, and mgiSendMail tags.
4. Save the send mail page.
5. FTP the information request form and send mail page to a web server running MGI.
6. View the form in a web broswer and complete the information request.

Step 1: Create an information request form.

Create an information request page named "inforequest.mgi" with form elements. In this example, the information request form consists of form fields for name, address, phone, email and comments. Name each form element uniquely. Enclose all form elements with HTML <FORM> tags and post the form to the send mail page (infoprocess.mgi).

This is an example information request form.

```
<FORM action="infoprocess.mgi" method="post">

<H2><CENTER>Information Request</CENTER></H2>

<P><CENTER>
Please complete all information below.
</CENTER></P>

<H2><CENTER>
<TABLE BORDER="1" CELLSPACING="1" CELLPADDING="3">
  <TR>
    <TD ALIGN="RIGHT">Name:</TD>
    <TD>&nbsp;
<INPUT NAME="Name" TYPE="text" SIZE="30"></TD>
  </TR>
  <TR>
    <TD ALIGN="RIGHT">Address:</TD>
    <TD>&nbsp;
<INPUT NAME="Address" TYPE="text" SIZE="30"></TD>
  </TR>
  <TR>
    <TD ALIGN="RIGHT">Phone:</TD>
    <TD>&nbsp;
<INPUT NAME="Phone" TYPE="text" SIZE="30"></TD>
  </TR>
  <TR>
    <TD ALIGN="RIGHT">Email:</TD>
    <TD>&nbsp;
<INPUT NAME="Email" TYPE="text" SIZE="30"></TD>
  </TR>
  <TR>
    <TD ALIGN="RIGHT">Questions/Comments:</TD>
    <TD>&nbsp;
<TEXTAREA NAME="Comments" ROWS="7" COLS="27">
</TEXTAREA></TD>
  </TR>
  <TR>
    <TD>&nbsp;</TD>
    <TD><P><CENTER>
    <INPUT TYPE="submit" VALUE="Submit Request">
    </CENTER></TD>
  </TR>
</TABLE>
```

```
</CENTER></H2>

</FORM>
```

Step 2: Create a send mail page for the information request and open it in a text editor.

Create a page named "infoprocess.mgi" to process the information request and send a confirmation email to the visitor. Open the send mail page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Insert the mgiSet tag, mgiPostArgument tag, and mgiSendMail tags.

To create the subject of the confirmation email, enter a beginning mgiSet tag, name parameter and ending mgiSet tag. In the name parameter, enter the name of the page variable "ConfirmationSubject". In the body of the mgiSet tags, construct the email subject by dynamically entering the customer's name from the information request form using an mgiPostArgument tag.

To send a confirmation email, enter the mgiSendMail tag. Embed the variable value in the subject of the mgiSendMail tag using the mgiGet tag and name parameter. In the name parameter, enter the name of the page variable to display "ConfirmationSubject". To process the information request, enter a second mgiSendMail tag.

This is an example send mail page.

```
<H2><CENTER>Information Request Submission</CENTER></H2>

<P><CENTER>Thank you, your information
request has been sent.</CENTER></P>

<mgiSet name="ConfirmationSubject">
<mgiPostArgument name="Name"> Receipt Confirmation
</mgiSet>

<mgiSendMail to={mgiPostArgument name="Email"}
from="info@domain.com" mailserver="mail.domain.com"
subject={mgiGet name="ConfirmationSubject"}>
Thank you <mgiPostArgument name="Name">.
We have received your request for additional
information. A representative will respond by
phone or email within 48 hours.
Please visit our site again.
</mgiSendMail>
```

```
<mgiSendMail to="info@domain.com"
from="info@domain.com" mailserver="mail.domain.com"
subject="Info Request">
Information Request
*******************
    Name: <mgiPostArgument name="Name">
 Address: <mgiPostArgument name="Address">
   Phone: <mgiPostArgument name="Phone">
   Email: <mgiPostArgument name="Email">
Comments: <mgiPostArgument name="Comments">
</mgiSendMail>
```

Step 4: Save the send mail page.

Save the changes you have made to the send mail page.

Step 5: FTP the information request form and send mail page to a web server running MGI.

Upload the information request form and send mail page from your local computer to the web server using an FTP program.

Step 6: View the form in a web broswer and complete the information request.

View the information request form in a web browser and complete all fields of the form. When you submit the form, a confirmation email is sent to the customer and the information request is sent to the company.

This is an example confirmation email. The subject of this email is "Andrew Carey Receipt Confirmation".

```
Thank you Valerie Crisp.
We have received your request for additional
information. A representative will respond by
phone or email within 48 hours.
Please visit our site again.
```

This is an example information request email.

```
Information Request
*******************
    Name: Andrew Carey
 Address: 125 Main St New York NY 20001
   Phone: 256-458-9654
   Email: acarey@aol.com
Comments: I am interested in product 233445. What
sizes are in stock?
```

Setting and Displaying Site Variables

Introduction

Variables are containers for information such as text, HTML and MGI tags. Variables allow you to "hold" information until you need to display it further down the page or on another page of the site. The information in page variables is available from the point at which they are set until they are overwriiten or until the end of the page that is currently processing, whichever occurs first. The information in site variables is stored in a database and is available from the point at which they are set until they are overwritten by a new value or deleted.

More often than not, the value of a variable is embedded in the parameter of another MGI tag. Read more about embedding.

In this example, a site variable is used to display the latest version of a software release throughout a web site. Once the version is set in a site variable, it can be displayed on any page of the site by using the mgiGet tag. The following example includes the version update page and one news page that displays the software version.

MGI Tags

- mgiGet
- mgiSet

Steps

1. Create a version update form.
2. Create a version update processing page and open it in a text editor.
3. Insert the mgiSet tag with a Site scope.
4. Save the version update processing page.
5. Create a software news page and open it in a text editor.
6. Insert the mgiGet tag with a Site scope.
7. Save the software news page.
8. FTP the version update form, version update processing page and software news page to a web server running MGI.
9. View the version update form in a web broswer and submit the new version number.

Step 1: Create a version update form.

Create a version update page named "versionupdate.mgi" with form elements. In this example, the version update form consists of a label and text field. Name each form element uniquely. Enclose all form elements with HTML <FORM> tags and post the form to the version update processing page (versionprocess.mgi).

This is an version update form.

```
<FORM action="versionprocess.mgi" method="post">

<H2><CENTER>Version Update</CENTER></H2>

<P><CENTER>New Version Number:
<INPUT NAME="version" TYPE="text" SIZE="10">
<INPUT TYPE="submit" VALUE="Update Version">
</CENTER>

</FORM>
```

Step 2: Create a version update processing page and open it in a text editor.

Create a page named "versionprocess.mgi" to save the version update in the site variable database. Open the version update processing page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Insert the mgiSet tag with a Site scope.

Insert the beginning mgiSet tag, name parameter, scope parameter, and ending mgiSet tag. In the name parameter, enter the name of the site variable "Version". In the scope parameter, enter "Site". In the body of the mgiSet tags, dynamically enter the new version number from the version update form using an mgiPostArgument tag.

This is an example version update processing page.

```
<H2><CENTER>Version Process</CENTER></H2>
<P>The version has been updated to
<mgiPostArgument name="version">.</P>
<mgiSet name="Version" scope="Site">
<mgiPostArgument name="version">
</mgiSet>
```

Step 4: Save the version update processing page.

Save the changes you have made to the version update processing page.

Step 5: Create a software news page and open it in a text editor.

Create a page named "news.mgi" to display software new. Open the software news page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 6: Insert the mgiGet tag with a Site scope.

Insert your cursor in the HTML where you want the version to appear and enter the mgiGet tag, name parameter and scope parameter. In the name parameter, enter the name of the site variable to display ("Version"). In the scope parameter, enter "Site".

This is an example software news page.

```
<H2><CENTER>Software News</CENTER></H2>
<CENTER>
<P><TABLE WIDTH="400" BORDER="0"
CELLSPACING="0" CELLPADDING="3">
  <TR>
    <TD><B>Latest Version</B>: Our software is constantly
updated to add new features and improve existing features.
Make sure you are up-to-date, download version
<mgiGet name="Version" scope="Site"> today!</TD>
  </TR>
</TABLE></P>
</CENTER>
```

Step 7: Save the software news page.

Save the changes you have made to the software news page.

Step 5: FTP the version update form, version update processing page and software news page to a web server running MGI.

Upload the version update form, version update processing page and software news page from your local computer to the web server using an FTP program.

Step 9: View the version update form in a web broswer and submit the new version number.

View the version update form (versionupdate.mgi) in a web browser. Enter a new version number (e.g., "2.0") and click the "Update Version" button. The new version number is then set in the page variable named "Version". View the software news page (news.mgi) in a web browser and the version you set appears in the paragraph. For example, if you updated the version to 2.0, the paragraph would read:

> **Latest Version**: Our software is constantly updated to add new features and improve existing features. Make sure you are up-to-date, download version 2.0 today!

The version will display until the version update form is submitted and the "Version" site variable is overwritten with a new version number.

[Return to the Variables Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Using If, Then, Else Comparisons

In the **Beginner Tutorial** section, learn how to make simple inline and regular if, then, else comparisons. In the **Advanced Tutorial** section, learn advanced conditional comparison techniques such as nested if statements. In the **Reference** section, view a complete technical reference for each tag used for conditional comparisons. In the **Downloads** section, download example code for each beginner and advanced tutorial. In the **FAQs** section, read answers to frequently asked questions.

Beginner If, Then, Else Tutorials

- B1. Simple Conditional Comparison with Text Result

If, Then, Else MGI Tag Reference

- mgiIf
- mgiInlineIf
- mgiSwitch

If, Then, Else Downloads

- B1. Download quiz example of a simple conditional comparison with text result (B1Conditional.zip - 4 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Simple Conditional Comparison with Text Result

Introduction

Conditional comparisons are one of the most versatile functions in MGI. The conditional tags allow you to perform logical comparisons between two values and display a result based on whether the comparison is true or false.

The mgiInlineIf tag is used when there is only one conditional comparison and the result is a string of text. The mgiIf tag is used when there are multiple conditional comparisons or the result of the comparison contains code in addition to a string of text (e.g., MGI tags). The mgiSwitch tag is used when there are multiple comparisons and multiple matching results. Text, HTML tags, and MGI tags may be included in the result of an mgiIf or mgiSwitch comparison. See the advanced tutorials for examples of mgiIf and mgiSwitch.

More often than not, one of the values in a conditional comparison is embedded in the tag's parameter from another source (page or site variables, post arguments, path arguments, etc.). Read more about embedding.

In this example, a conditional comparisons is used to determine a passing or failing grade from an online quiz. The example consists of a hard-coded quiz form and a grade processing page. On the processing page, the numeric quiz score is tabulated from the quiz answers using an mgiMath tag and the total quiz score is compared with a the passing grade to determine which text displays.

MGI Tags

- mgiInlineIf

Steps

1. Create a quiz form.
2. Create a grade processing page and open it in a text editor.
3. Insert the mgiInlineIf tag.
4. Save the grade processing page.
5. FTP the quiz form and grade processing page to the web server running MGI.
6. View the quiz form in a web browser and take the quiz.

Step 1: Create a quiz form.

Create a quiz page named "quiz.mgi" with form elements. In this example, the quiz is multiple choice and is constructed of radio buttons. Name each form element uniquely and assign a grade value to each answer. Enclose all form elements with HTML <FORM> tags and post the form to the grade processing page (grade.mgi)

This is an example quiz form.

```
<FORM action="grade.mgi" method="post">

<H2><CENTER>Grade 1 Quiz</CENTER></H2>

<P><CENTER><FONT COLOR="#006600" SIZE="-1">
<mgiDate></FONT></CENTER></P>

<P>Student Name:
<INPUT NAME="StudentName" TYPE="text" SIZE="30"></P>

<P>1. What is the result of 1 + 2?</P>

<P><INPUT TYPE="radio" VALUE="0" NAME="Q1">1<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q1">2<BR>
<INPUT TYPE="radio" VALUE="20" NAME="Q1">3<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q1">4</P>

<P>2. What noise does a cow make?</P>

<P><INPUT TYPE="radio" VALUE="0" NAME="Q2">oink<BR>
<INPUT TYPE="radio" VALUE="20" NAME="Q2">moo<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q2">bark<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q2">meow</P>

<P>3. <FONT COLOR="#ff0000">
What color is this sentence?</FONT></P>

<P><INPUT TYPE="radio" VALUE="0" NAME="Q3">green<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q3">blue<BR>
<INPUT TYPE="radio" VALUE="20" NAME="Q3">red<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q3">white</P>

<P>4. What holiday is on December 25th?</P>

<P><INPUT TYPE="radio" VALUE="20" NAME="Q4">Christmas<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q4">Halloween<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q4">Thanksgiving<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q4">Easter</P>
```

```
<P>5. What is your teacher's name?</P>

<P><INPUT TYPE="radio" VALUE="0" NAME="Q5">Mrs. Smith<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q5">Mrs. Arnold<BR>
<INPUT TYPE="radio" VALUE="0" NAME="Q5">Ms. Pepperdyne<BR>
<INPUT TYPE="radio" VALUE="20" NAME="Q5">Mr. Clemmons</P>

<P><INPUT NAME="name" TYPE="submit"
VALUE="Submit Quiz for Grading">
<INPUT NAME="name" TYPE="reset"
VALUE="Reset Quiz Answers">

</FORM>
```

Step 2: Create a grade processing page and open it in a text editor.

Create a page named "grade.mgi" to process the quiz grade and display the passing or failing grade message. Open the grade processing page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Insert the mgiInlineIf tag.

Insert your cursor in the HTML where you want the quiz grade result to display. Enter the mgiMath tag with mgiPostArgument tags to calculate the total quiz score. Set the total quiz score value in a variable using the mgiSet tag. Below the variable, enter the mgiInlineIf tag, lhs parameter, relationship parameter, rhs parameter, then parameter and else parameter.

In the lhs parameter, embed the quiz score total from the variable using an mgiGet tag. In the relationship parameter, enter the type of comparison to perform. In this case, test whether the lhs (left-hand side) value is greater than or equal to the rhs (right-hand side) value. In the rhs parameter, enter the passing quiz score. In the "then" parameter, enter the text that displays if the student's quiz score is greater than or equal to the passing quiz score (e.g., "Congratulations, you pass and receive 1 gold quiz star"). In the "else" parameter, enter the text that displays if the student's quiz score is less than the passing quiz score (e.g., "You did not pass. Please see the teacher.").

This is an example grade processing page.

```
<H2><CENTER>Grade 1 Quiz Score</CENTER></H2>

<P><CENTER><FONT COLOR="#006600" SIZE="-1">
<mgiDate></FONT></CENTER></P>
```

```
<mgiSet name="QuizScore">
<mgiMath resultPrecision="0">
<mgiPostArgument name="Q1">+<mgiPostArgument name="Q2">+
<mgiPostArgument name="Q3">+<mgiPostArgument name="Q4">+
<mgiPostArgument name="Q5">
</mgiMath>
</mgiSet>

<CENTER>

<P><mgiPostArgument name="StudentName">,
your quiz score is <mgiGet name="QuizScore">.</P>

<P>
<mgiInlineIf lhs={mgiGet name="QuizScore"}
relationship="greaterThanOrEqualTo" rhs="60"
then="Congratulations, you pass and
receive 1 gold quiz star."
else="You did not pass. Please see the teacher.">
</P>

</CENTER>
```

Step 4: Save the grade processing page.

Save the changes you have made to the grade processing page.

Step 5: FTP the quiz form and grade processing page to the web server running MGI.

Upload the quiz form and grade processing page from your local computer to the web server using an FTP program.

Step 6: View the quiz form in a web browser and take the quiz.

View the quiz form in a web bowser. Complete and submit the quiz. If your score is 60 or greater, the message "Congratulations, you pass and receive 1 gold quiz star." displays. If your score is less than 60, the message "You did not pass. Please see the teacher." displays.

[Return to the If, Then, Else Menu]

Creating and Populating Databases

In the **Beginner Tutorial** section, learn how to create, modify, and delete databases, fields, and records. In the **Advanced Tutorial** section, learn importing and exporting and advanced string searches. In the **Reference** section, view a complete technical reference for each tag used to create and populate databases. In the **Downloads** section, download example code for each beginner and advanced tutorial. In the **FAQs** section, read answers to frequently asked database creation and population questions.

Beginner Database Creation and Population Tutorials

- B1. Creating a Database, Adding Database Fields, and Populating the Database with Records

---

Advanced Database Creation and Population Tutorials

- A1. Exporting Database Records
- A2. Importing Database Records into an Existing Database or to Create a New Database
- A3. Locating Records in the Admin with an Advanced Search

Database Creation and Population MGI Tag Reference

- mgiEditDatabase

Database Creation and Population Downloads

- B1. Download a sample database administration page (B1DB.zip - 4 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Creating a Database and Database Fields

Introduction

Creating databases, adding the database field structure and adding database records is accomplished with one page and the mgiEditDatabase tag.

MGI Tags

- mgiEditDatabase

Steps

1. Create a database administration page in a text editor.
2. Insert the mgiEditDatabase tag.
3. Save the database administration page.
4. FTP the database administration page to the web server running MGI.
5. View the database administration page in a web browser.
6. Enter the name of a new database.
7. Add each database field.
8. Add database records.

Step 1: Create a database administration page in a text editor.

Create a new page in a text editing program to display the web-based, database administration interface.

Step 2: Insert the mgiEditDatabase tag.

On the database administration page, enter the mgiEditDatabase tag.

```
<mgiEditDatabase>
```

Step 3: Save the database administration page.

Save the database administration page and name it "dbadmin.mgi".

Step 4: FTP the database administration page to the web server running MGI.

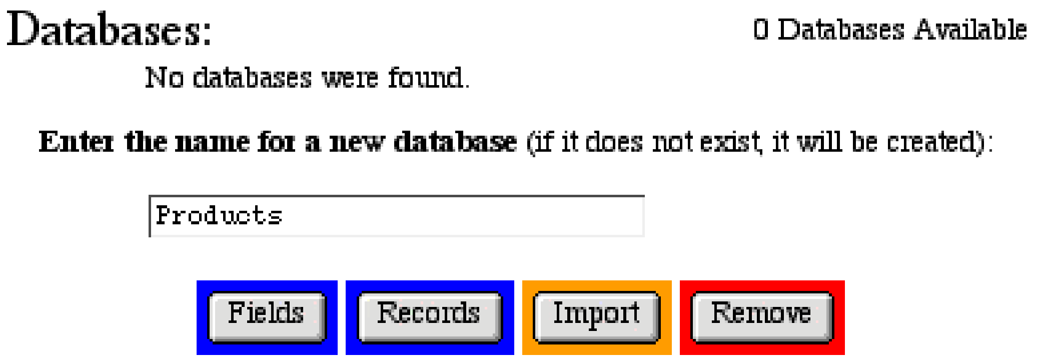
Upload the database administration page (dbadmin.mgi) from your local computer to the web server using an FTP program.

Step 5: View the database administration page in a web browser.

View the database administration page (dbadmin.mgi) in a browser. The first screen of the web-based, administration interface is displayed:

Step 6: Enter the name of a new database.

Below the sentence "Enter the name for a new database (if it doesn't exist, it will be created):" enter the name of a new database ("Products" in this example). Remember, database names are case-sensitive. Click the "Fields" button to create the new database and proceed to the field screen of the interface. New databases are unique to and available to all files in the region in which they are created.



Databases:

0 Databases Available

No databases were found.

Enter the name for a new database (if it does not exist, it will be created):

Products

Step 7: Add each database field.

In the field screen of the administration interface, enter a name, type, index selection, uniqueness selection, and length/precision for each database field. Before creating the fields, consider what type of information will be entered in the database records. Your database structure can be changed after you create the field structure, but it is easier to plan ahead for each piece of information that you need.

Once you have decided which fields to include in the database, consider the type of information that will be entered in each field. An MGI database field can contain the following types of information:

- **True/False (Boolean):** a field containing the values "True" or "False" to discriminate records.
- **Whole Number (Integer):** a whole number field can contain only whole numbers without decimals such as quantities, etc.
- **Positive Number (Unsigned Integer):** a positive number field can contain only positive whole numbers without decimals such as years, etc.
- **Decimal Number (Multi-Precision Float):** a decimal number field can contain decimal numbers with multiple decimal places such as prices, weights, tax rates, percents, etc. Enter the number of decimal places in the "Length/Prec" box when creating a Decimal Number field.
- **Text (Max Length 250):** a text field can contain up to 250 alpha-numeric characters. Enter the maximum number of characters (up to 250) in the "Length/Prec" box when creating a text field. The default length is 25 characters.
- **Long Text:** a long text field can contain text that is greater than 250 alpha-numeric characters, but the information in a long text field cannot be used for ordering and cannot be designated as unique.

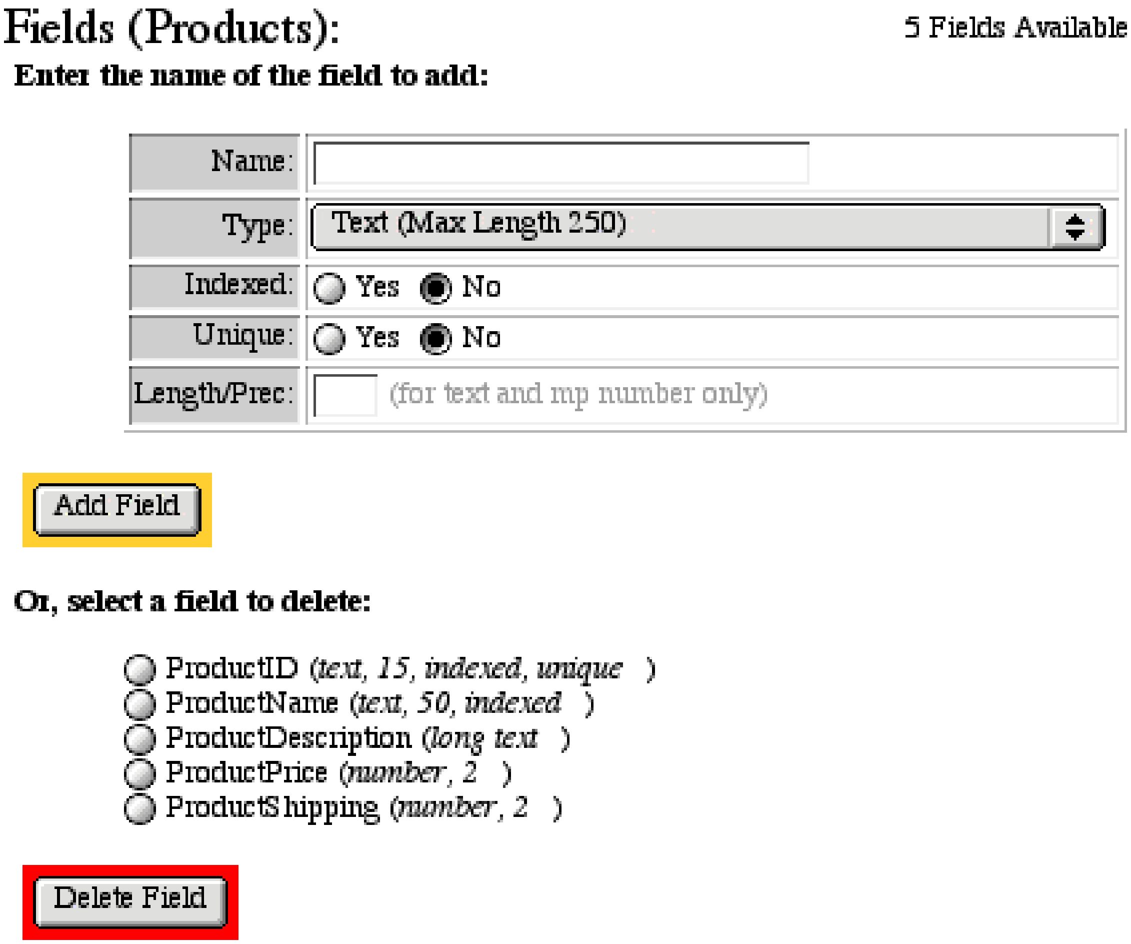
Next, consider which fields may be need to be searched (in the database record administraton screen or using an mgiSearchDatabase tag) or used to order database search results. **Any field that you want to search or use for ordering search results must be indexed when you create the field**. Creating an indexed field does not predetermine the order of your search results; it only allows you to order results by that criteria. The order of results is actually specified in the mgiSearchDatabase tag. Indexing fields does add some overhead to the searching process, so only index fields that you need to use for searching or ordering search results. Do not index every field!

Finally, consider whether information in the field should be designated as unique. Unique fields do not allow duplicate information (including multiple records with blank information in the unique field). For example, you may want to insure that Product Indentification numbers are unique.

To create a field, enter the case-sensitive field name in the area labeled "Name". The length of field names in the internal MGI database is limited according to the type of field and operating system. All field names that are not indexed are limited to 63 characters. The limit for non-indexed field names applies to all operating systems. Indexed field names on Macintosh operating systems 9.04 and prior are limited to 25 characters. Indexed field names on all supported Windows operating systems are limited to 63 characters.

Select a field type from the pop-up menu labeled "Type". Select "Yes" beside the "Indexed" label if the field will be searched or used to order database search results. Select "No" beside the "Indexed" label if the field will not be searched or used to order database search results. Select "Yes" beside the "Unique" label if the contents of the field are required to be unique. Select "No" beside the "Unique" label if the contents of the field are not required to be unique. For text field types and decimal number field types, enter the maximum number of characters or the number of decimal places, respectively, in the area labeled "Length/Prec". Click the "Add Field" button to add the field to the database. Fields are listed in the order they are created below the new field interface.

In this example, fields for a product database were created:



Fields (Products): 5 Fields Available

Enter the name of the field to add:

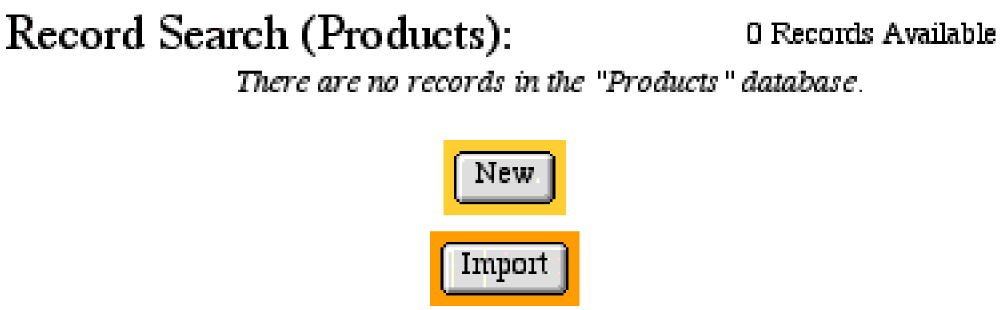
Name:	
Type:	Text (Max Length 250)
Indexed:	○ Yes ● No
Unique:	○ Yes ● No
Length/Prec:	(for text and mp number only)

Or, select a field to delete:

○ ProductID (*text, 15, indexed, unique*)
○ ProductName (*text, 50, indexed*)
○ ProductDescription (*long text*)
○ ProductPrice (*number, 2*)
○ ProductShipping (*number, 2*)

Step 8: Add database records.

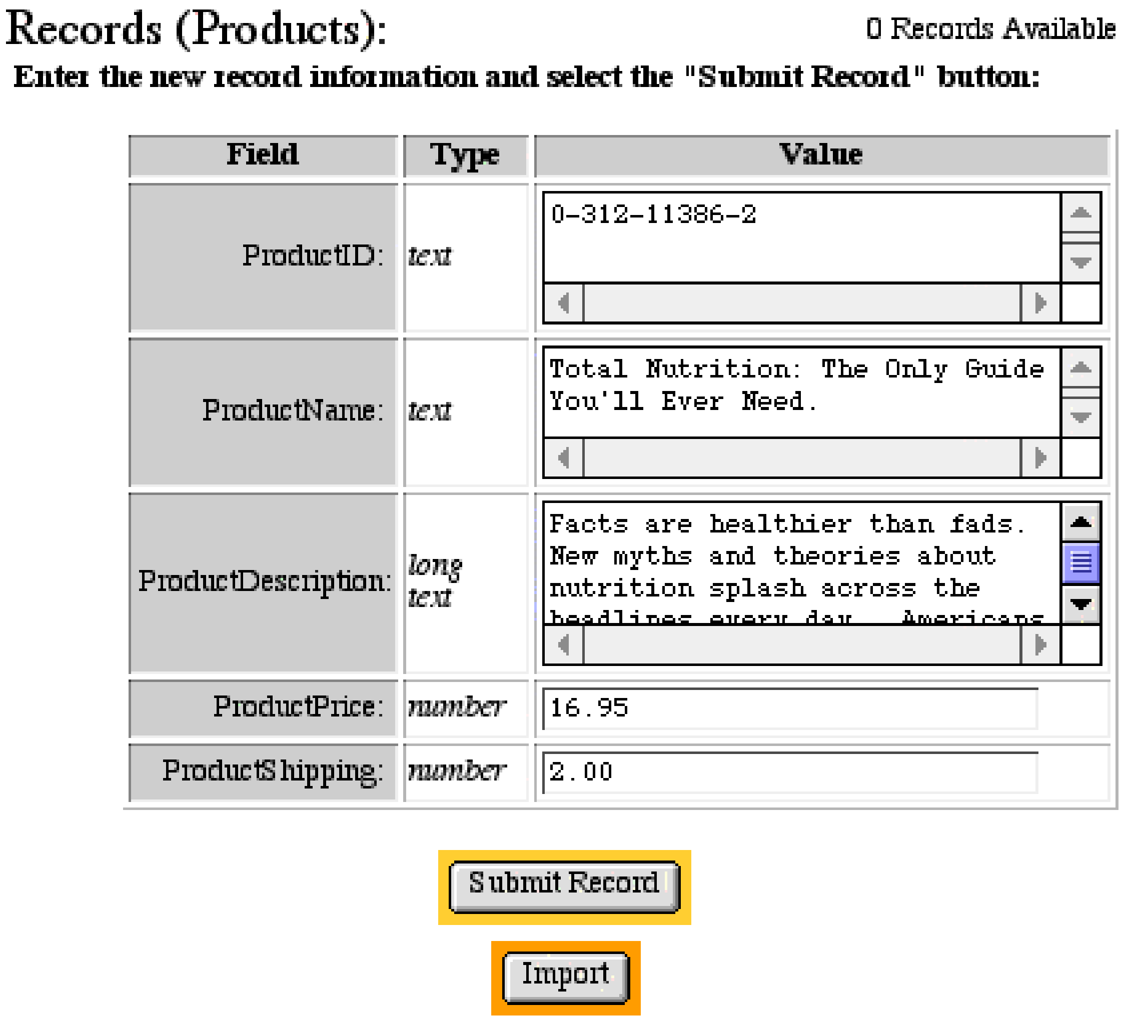
Return to the main database administration screen by entering and viewing the database administration page (dbadmin.mgi) page in a web browser. Select the radio button beside the database you wish to populate and click the "Records" button. If no records are present, the "New" and "Import" buttons are displayed. Click the "New" button to add a database record.



Record Search (Products): 0 Records Available

There are no records in the "Products" database.

On the new record screen of the database administration interface, enter information in each field and click the "Submit Record" button to save the new data.

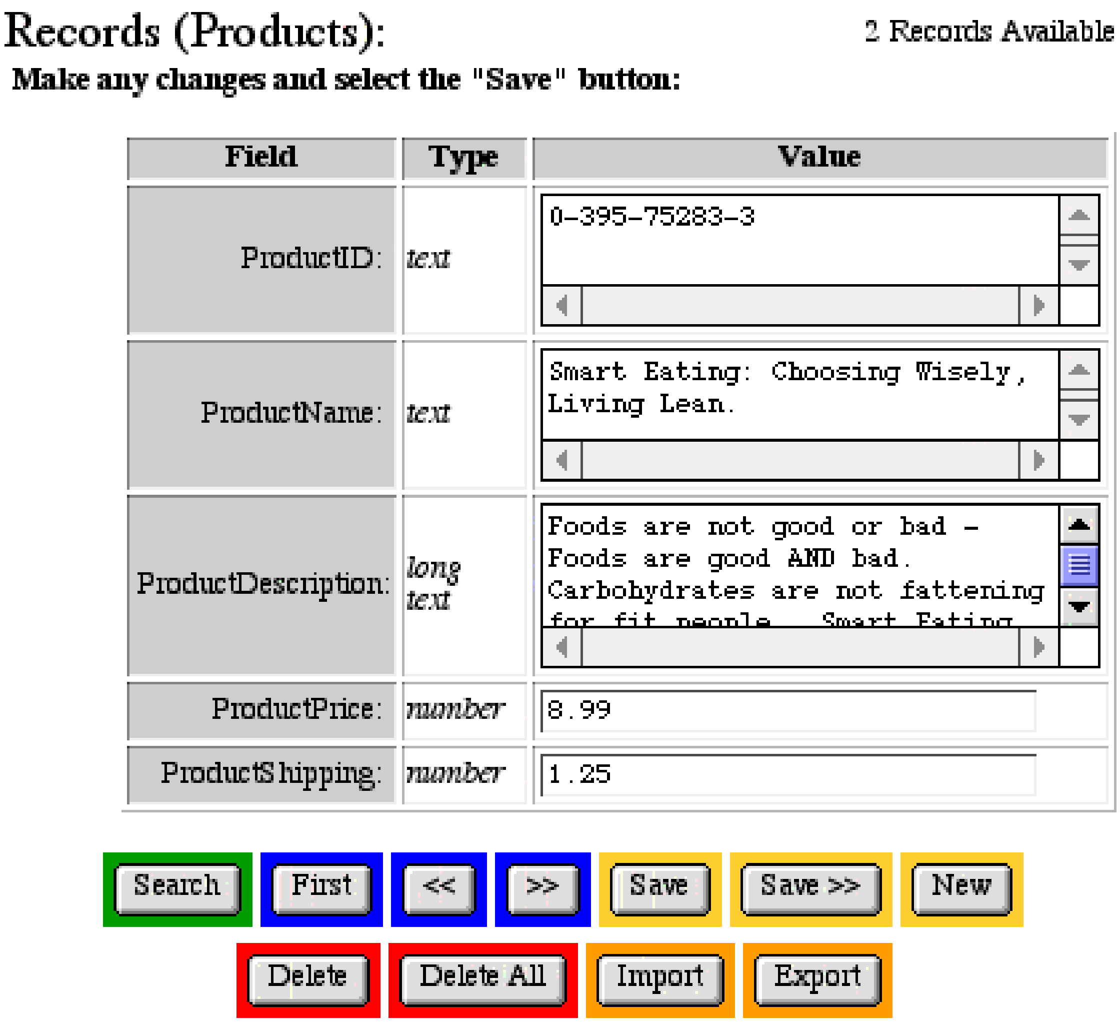


Records (Products): 0 Records Available

Enter the new record information and select the "Submit Record" button:

Field	Type	Value
ProductID:	*text*	0-312-11386-2
ProductName:	*text*	Total Nutrition: The Only Guide You'll Ever Need.
ProductDescription:	*long text*	Facts are healthier than fads. New myths and theories about nutrition splash across the headlines every day. Americans
ProductPrice:	*number*	16.95
ProductShipping:	*number*	2.00

After submitting a new record, the message "Record successfully added" displays and allows you to enter additional new records at the new record screen of the database administration interface. Enter new records following the directions above until all information has been added. To view the records, click the "First" button and then click the ">>" (Next) and "<<" (Previous) buttons to browse the records.

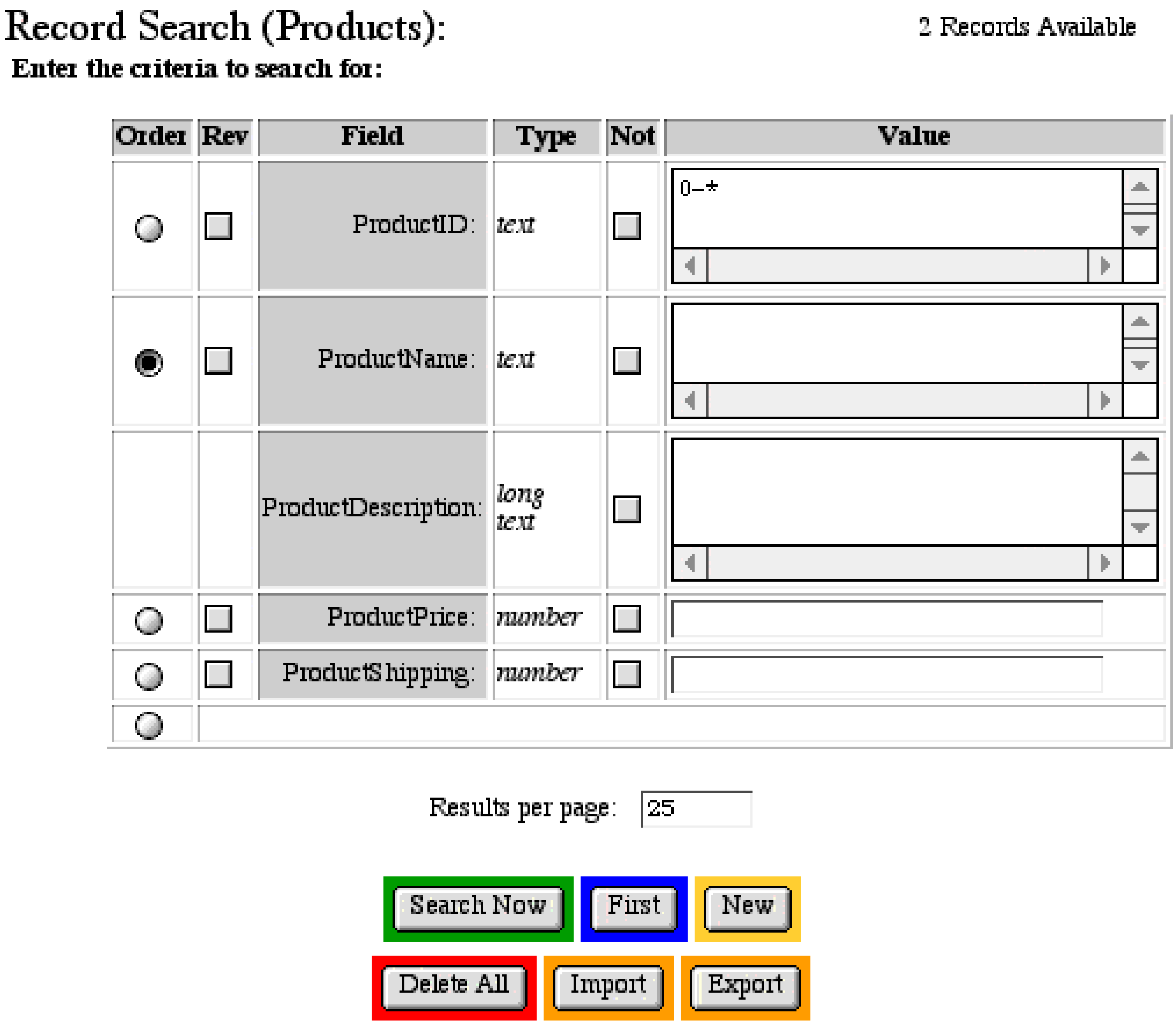


Records (Products): 2 Records Available

Make any changes and select the "Save" button:

Field	Type	Value
ProductID:	*text*	0-395-75283-3
ProductName:	*text*	Smart Eating: Choosing Wisely, Living Lean.
ProductDescription:	*long text*	Foods are not good or bad - Foods are good AND bad. Carbohydrates are not fattening for fit people. Smart Eating
ProductPrice:	*number*	8.99
ProductShipping:	*number*	1.25

To locate records, click the "Search" button, enter search criteria into any indexed field and click the "Search Now" button. Use an asterisk * for wildcard and partial searches (see example below). Select the radio button under the "Order" column to order search results by the selected field. Check the box under the "Rev" column beside the field you have selected in the "Order" column to reverse the order of search results. If no ordering is selected, search results are displayed in the ordered they were entered into the database. If a field is ordered, search results are ordered in ascending order (A to Z, smallest to largest) by default. If a field is ordered and reversed, search results are ordered in descending order (Z to A, largest to smallest). When results are displayed, select the radio button beside any search result and click the "View" button to view the full record.



Record Search (Products):

2 Records Available

Enter the criteria to search for:

Order	Rev	Field	Type	Not	Value
○	□	ProductID:	text	□	0-*
●	□	ProductName:	text	□	
		ProductDescription:	long text	□	
○	□	ProductPrice:	number	□	
○	□	ProductShipping:	number	□	
○					

Results per page: 25

To update a record, browse to view the record or locate and view the record via a search. While viewing the record, make changes in any field and click the "Save" button to save the changes and reload the record or click the "Save >>" button to save the changes to the existing record and advance to the next record.

When viewing a record, click the "Delete" button to delete that record. Click the "Delete All" button on any screen to delete all records in the database.

Comments and Notes

The mgiEditDatabase tag gives you and anyone who views the page in a web browser access to modify databases. Keep your databases secure by password-protecting the database administration page with an mgiAuthenticate or mgiAuthenticateDB tag.

Exporting Database Records

Introduction

MGI stores database information in a proprietary database format for the database that is built into the MGI code. However, you can use the export and import features of the mgiEditDatabase tag to move database records between different programs or to easily edit existing data or database structures. MGI exports records in a standard tab-delimited format.

MGI Tags

- mgiEditDatabase

Steps

1. Create a database administration page in a text editor.
2. Insert the mgiEditDatabase tag.
3. Save the database administration page.
4. FTP the database administration page to the web server running MGI.
5. View the database administration page in a web browser.
6. Select a database.
7. Export the database records.

Step 1: Create a database administration page in a text editor.

Create a new page in a text editing program to display the web-based, database administration interface.

Step 2: Insert the mgiEditDatabase tag.

On the database administration page, enter the mgiEditDatabase tag.

```
<mgiEditDatabase>
```

Step 3: Save the database administration page.

Save the database administration page and name it "dbadmin.mgi".

Step 4: FTP the database administration page to the web server running MGI.

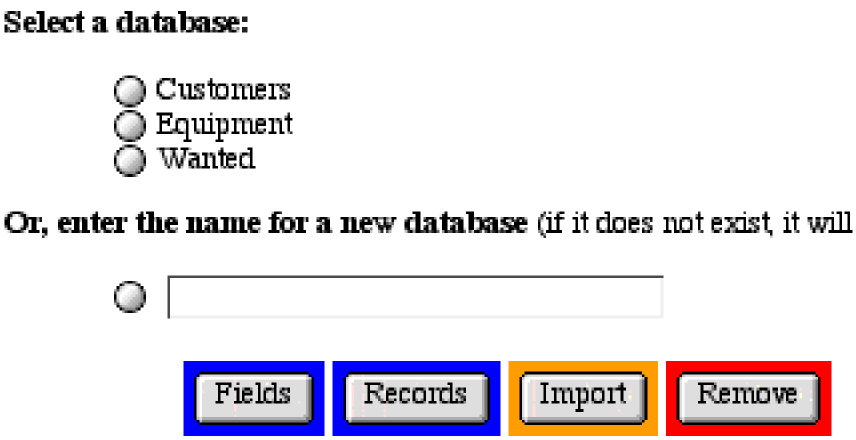
Upload the database administration page (dbadmin.mgi) from your local computer to the web server using an FTP program.

Step 5: View the database administration page in a web browser.

View the database administration page (dbadmin.mgi) in a browser. The first screen of the web-based, administration interface is displayed.

Step 6: Select a database.

Below "Select a database", click the radio button beside the database you wish to export and click the "Records" button. The record screen of the administration interface displays.



Select a database:

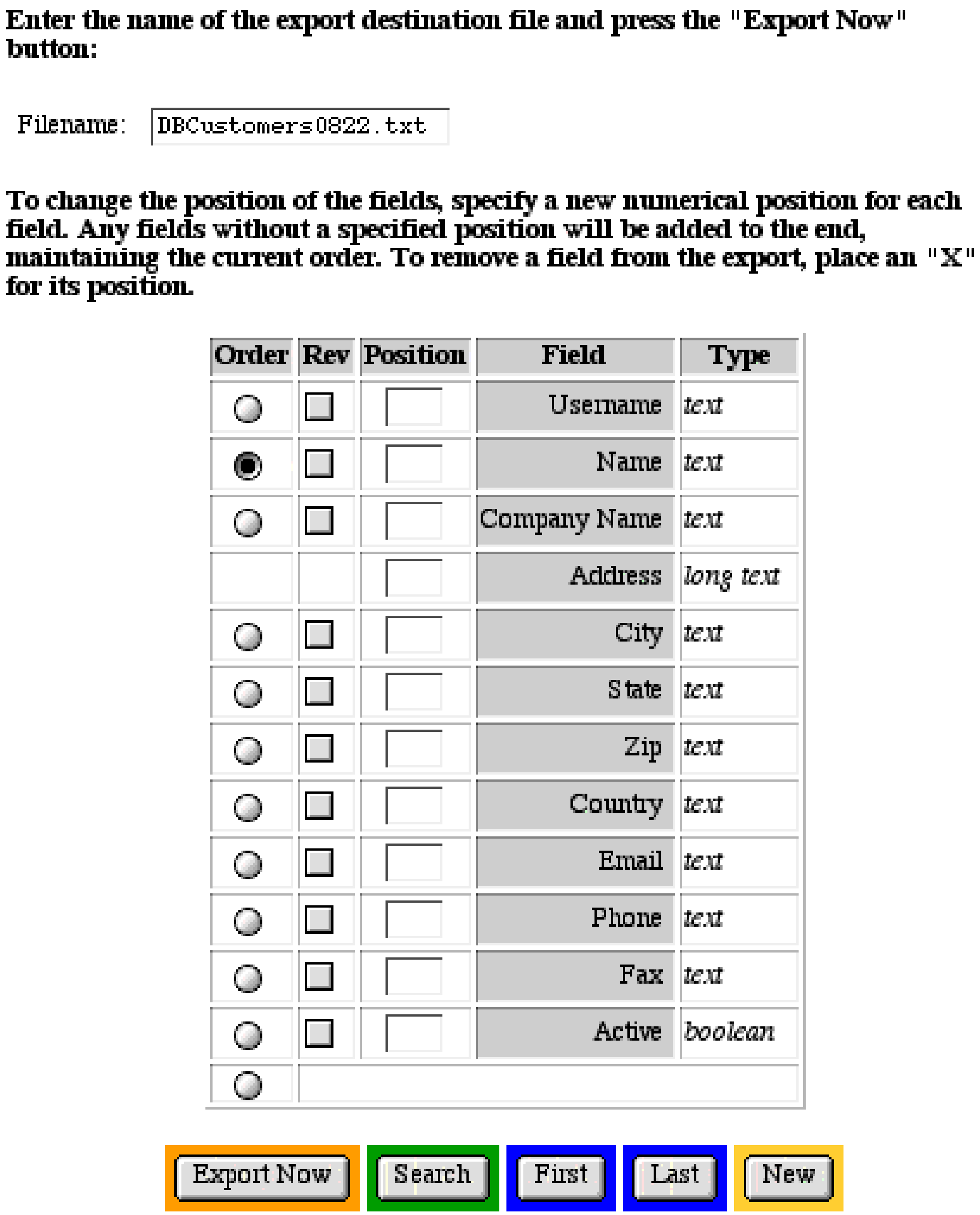
○ Customers
○ Equipment
○ Wanted

Or, enter the name for a new database (if it does not exist, it will be created):

○

Step 7: Export the database records.

In the record screen of the administration interface, click the "Export" button. The export interface displays:



Enter the name of the export destination file and press the "Export Now" button:

Filename: DBCustomers0822.txt

To change the position of the fields, specify a new numerical position for each field. Any fields without a specified position will be added to the end, maintaining the current order. To remove a field from the export, place an "X" for its position.

Order	Rev	Position	Field	Type
○	□		Username	*text*
●	□		Name	*text*
○	□		Company Name	*text*
			Address	*long text*
○	□		City	*text*
○	□		State	*text*
○	□		Zip	*text*
○	□		Country	*text*
○	□		Email	*text*
○	□		Phone	*text*
○	□		Fax	*text*
○	□		Active	*boolean*
○				

Beside "Filename", enter the name of the file to export. Exported files are saved to the web server in the same folder as the database administration page. Use a unique name if you do not wish to overwrite an existing file.

By default, database records are exported in the order of the original field creation and are sorted by the order entered.

To change the field order of the export, enter the numerical position for each field under "Position" (e.g., "1", "2". etc.). Any fields that are left blank will be added to the end of the specified field order and will be exported in the order they appear. To keep a field from being exported, place an "X" in the field under "Position".

To change the sorting of records for the export, select a field to use for the sort under "Order". To change the sort from ascending (A to Z, smallest to largest) to descending (Z to A, largest to smallest), check the box under "Rev" beside the field you have chosen for the sort.

Click the "Export Now" button to export the records. Records are exported to the specified file on the web server. When the export is complete the search interface of the database administration displays. The exported file is written in tab-delimited format to the same folder as the database administration page. You may use an FTP program to download the exported file.

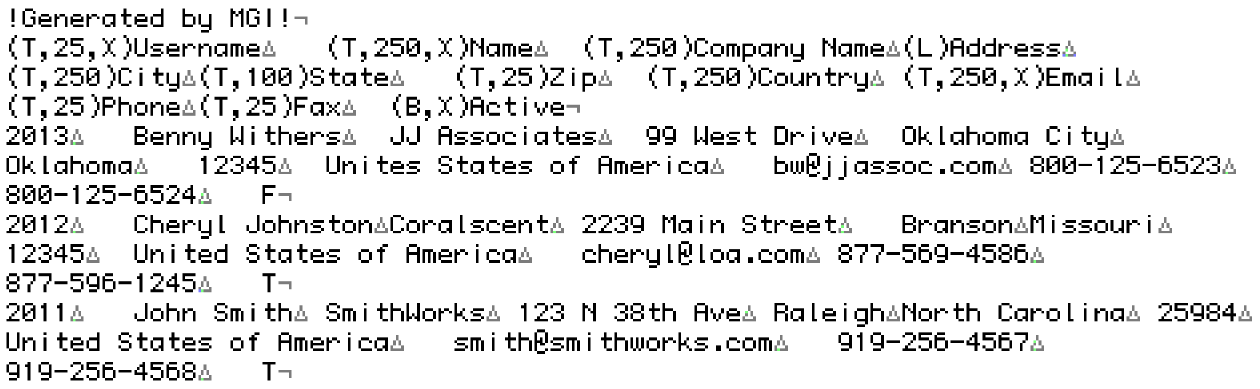
The first line of the exported file contains a comment "!Generated by MGI!". The second line of the exported file contains the field definitions (in parentheses) to the left of each field name. Each type of MGI field (text, long text, boolean, etc.) is represented by a letter:

- **B** - True/False (Boolean)
- **I** - Whole Number (Integer)
- **U** - Positive Number (Unsigned Integer)
- **N** - Decimal Number (Multi-Precision Float)
- **T** - Text
- **L** - Long Text

Text and decimal number fields are followed by the specified number of characters or decimal places. Field definitions with an "X" are indexed. Field definitions with a "Q" are unique.

The remaining lines in the exported file are the tab-delimited data of each record. All lines in the exported file end with a carriage return (Mac) or carriage return/line fee (NT).

The following is an example exported file. The text in this example is wrapped for viewing purposes, however each record is actually only one row of tab-delimited text. The triangles in this example represent tabs and the horizontal lines as the end of each row represent a carriage return.



```
!Generated by MGI!¬
(T,25,X)Username△   (T,250,X)Name△  (T,250)Company Name△(L)Address△
(T,250)City△(T,100)State△   (T,25)Zip△  (T,250)Country△ (T,250,X)Email△
(T,25)Phone△(T,25)Fax△  (B,X)Active¬
2013△   Benny Withers△  JJ Associates△  99 West Drive△  Oklahoma City△
Oklahoma△   12345△  Unites States of America△   bw@jjassoc.com△ 800-125-6523△
800-125-6524△   F¬
2012△   Cheryl Johnston△Coralscent△ 2239 Main Street△   Branson△Missouri△
12345△  United States of America△   cheryl@loa.com△ 877-569-4586△
877-596-1245△   T¬
2011△   John Smith△ SmithWorks△ 123 N 38th Ave△ Raleigh△North Carolina△ 25984△
United States of America△   smith@smithworks.com△   919-256-4567△
919-256-4568△   T¬
```

Comments and Notes

The mgiEditDatabase tag gives you and anyone who views the page in a web browser access to modify databases. Keep your databases secure by password-protecting the database administration page with an mgiAuthenticate or mgiAuthenticateDB tag.

[Return to the Creating and Populating Databases Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Importing Database Records into an Existing Database or to Create a New Database

Introduction

MGI stores database information in a proprietary database format for the database that is built into the MGI code. However, you can use the export and import features of the mgiEditDatabase tag to move database records between different programs or to easily edit existing data or database structures.

A file must be in a tab-delimited format in order to import. MGI cannot import any other formats, however tab-delimited is a common, standard format for database and spreadsheet programs (e.g., MS Excel, FileMaker, etc.).

You may import record data into an existing database structure (i.e., add records to a database) or you may have MGI create a new database structure (i.e., create fields and add records) during the import. Creating a new database with a simple import is ideal for minor adjustments in the database structure.

MGI Tags

- mgiEditDatabase

Steps

1. Create a database administration page in a text editor.
2. Insert the mgiEditDatabase tag.
3. Save the database administration page.
4. Create a tab-delimited import file.
5. Save the import file.
6. FTP the database administration page and import file to the web server running MGI.
7. View the database administration page in a web browser.
8. Select an existing database or create a new database.
9. Import the file.

Step 1: Create a database administration page in a text editor.

Create a new page in a text editing program to display the web-based, database administration interface.

Step 2: Insert the mgiEditDatabase tag.

On the database administration page, enter the mgiEditDatabase tag.

```
<mgiEditDatabase>
```

Step 3: Save the database administration page.

Save the database administration page and name it "dbadmin.mgi".

Step 4: Create a tab-delimited import file.

Depending on the current format of your data, you may need to "export" or "save as.." from an existing program in order to create a tab-delimited file. If you choose, you may also prepare your data directly in a text file. In any case, create a text file of tab-delimited data to import.

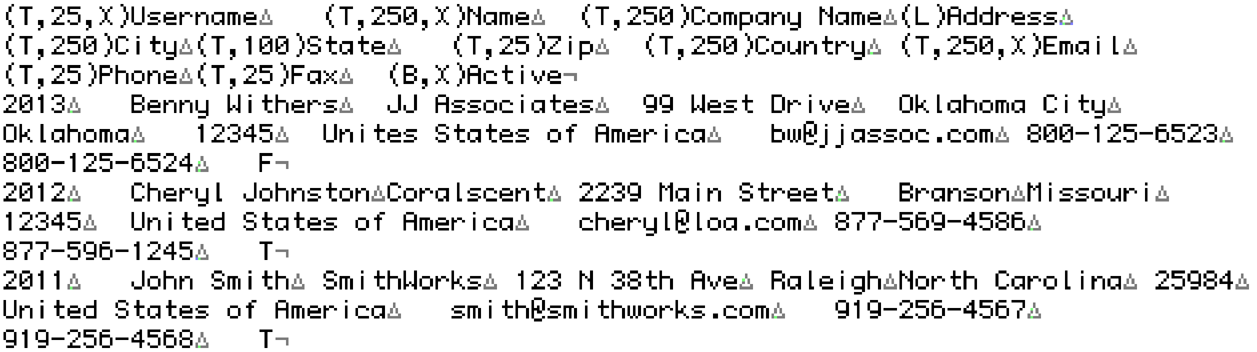
Next, enter the field definitions and field names as the first row in your import text file. Field definitions should be written within parentheses and the field name should be to the right of the parentheses (no space). Field names are **case-sensitive**. Each type of MGI field (text, long text, boolean, etc.) is represented by a letter:

- **B** - True/False (Boolean)
- **I** - Whole Number (Integer)
- **U** - Positive Number (Unsigned Integer)
- **N** - Decimal Number (Multi-Precision Float)
- **T** - Text
- **L** - Long Text

Text and decimal number fields are followed by a comma and the specified number of characters (text) or decimal places (decimal number). Field definitions followed by a comma and an "X" are indexed. Field definitions followed by a comma and a "Q" are unique. All lines in the file should end with a carriage return (Mac) or carriage return/line fee (NT).

The field definitions and names should also be tab-delimited and should match the order of your tab-delimited data. However, the fields are not required to match the order of an existing database structure - fields will be matched during import even if they are in a different order. **If your field order does match the field order in your existing database, the easiest way to get the field definitions and names is to export the existing database structure in MGI, then copy and paste the field definition line to the top of your import file**!

The following is an example import file. The text in this example is wrapped for viewing purposes, however each record is actually only one row of tab-delimited text. The triangles in this example represent tabs and the horizontal lines as the end of each row represent a carriage return.



```
(T,25,X)Username△   (T,250,X)Name△  (T,250)Company Name△(L)Address△
(T,250)City△(T,100)State△   (T,25)Zip△  (T,250)Country△ (T,250,X)Email△
(T,25)Phone△(T,25)Fax△  (B,X)Active¬
2013△   Benny Withers△  JJ Associates△  99 West Drive△  Oklahoma City△
Oklahoma△   12345△  Unites States of America△   bw@jjassoc.com△ 800-125-6523△
800-125-6524△   F¬
2012△   Cheryl Johnston△Coralscent△ 2239 Main Street△   Branson△Missouri△
12345△  United States of America△   cheryl@loa.com△ 877-569-4586△
877-596-1245△   T¬
2011△   John Smith△ SmithWorks△ 123 N 38th Ave△ Raleigh△North Carolina△ 25984△
United States of America△   smith@smithworks.com△   919-256-4567△
919-256-4568△   T¬
```

Step 5: Save the import file.

Save the import file. In this example the import file is named "import.txt", but you may choose any name.

Step 6: FTP the database administration page and import file to the web server running MGI.

Upload the database administration page (dbadmin.mgi) and import file (import.txt) from your local computer to the web server using an FTP program.

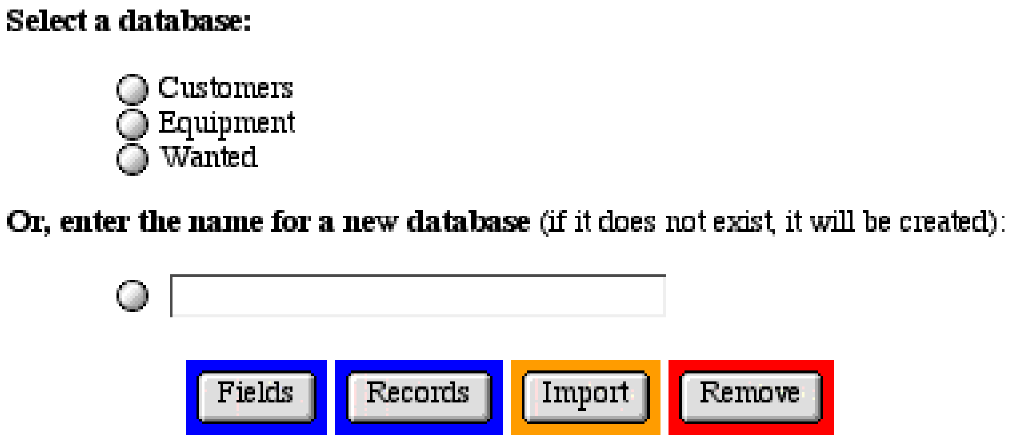
Step 7: View the database administration page in a web browser.

View the database administration page (dbadmin.mgi) in a browser. The first screen of the web-based, administration interface is displayed.

Step 8: Select an existing database or create a new database.

To import data and add records to an existing database, select a database and click the "Records" button, then click the "Import" button on the record screen of the administration interface. The import screen of the administration interface displays.

To create a new database structure (create fields and add records) during the import, create a new database by enter a database name under "Or, enter the name for a new database" and click the "Import" button. Database names are **case-sensitive**. The import screen of the administration interface displays.



Select a database:

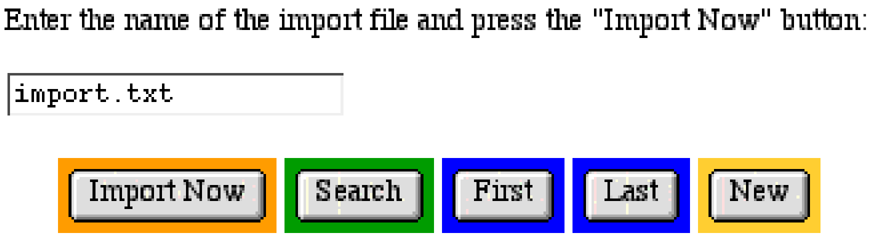
○ Customers
○ Equipment
○ Wanted

Or, enter the name for a new database (if it does not exist, it will be created):

○

Step 9: Import the file.

Whether you are adding to an existing database or creating a new database structure, the import screen of the administration interface is the same (except for addition buttons in the first version).



Enter the name of the import file and press the "Import Now" button:

import.txt

Enter the name of the import file in the text box and click the "Import Now" button to import. Any import errors will be displayed or you will return to the record screen of the administration interface when the import is complete. The total number of records in a database is located in the upper right corner of all record screens.

Comments and Notes

The mgiEditDatabase tag gives you and anyone who views the page in a web browser access to modify databases. Keep your databases secure by password-protecting the database administration page with an mgiAuthenticate or mgiAuthenticateDB tag.

[Return to the Creating and Populating Databases Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Locating Records in the Admin with an Advanced Search

Introduction

The search screen of the database administration interface allows you to search for data in specific fields and order results. You may use an advanced string search in the administration interface by adding the advancedSearch parameter to the mgiEditDatabase tag.

MGI Tags

- mgiEditDatabase

Steps

1. Create a database administration page in a text editor.
2. Insert the mgiEditDatabase tag and advancedSearch parameter.
3. Save the database administration page.
4. FTP the database administration page to the web server running MGI.
5. View the database administration page in a web browser.
6. Select a database.
7. Enter an advanced search.

Step 1: Create a database administration page in a text editor.

Create a new page in a text editing program to display the web-based, database administration interface.

Step 2: Insert the mgiEditDatabase tag and advancedSearch parameter.

On the database administration page, enter the mgiEditDatabase tag and set the advancedSearch parameter to "On".

```
<mgiEditDatabase advancedSearch="On">
```

Step 3: Save the database administration page.

Save the database administration page and name it "dbadmin.mgi".

Step 4: FTP the database administration page to the web server running MGI.

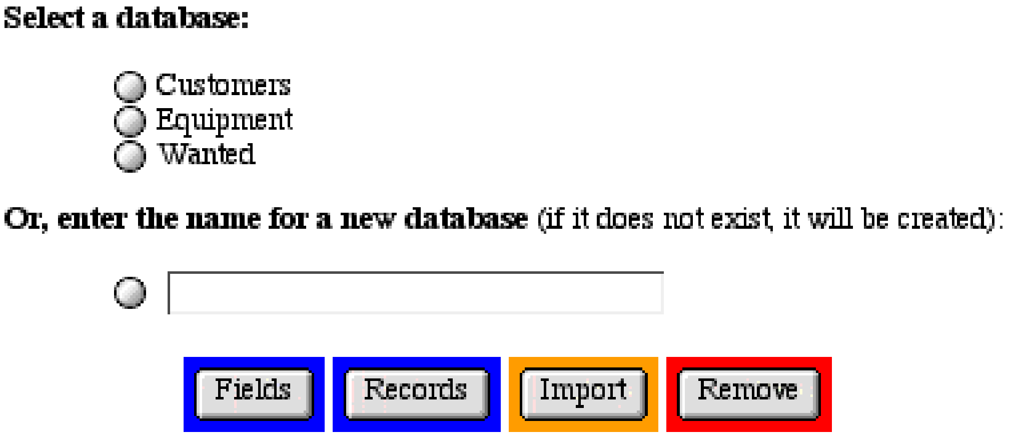
Upload the database administration page (dbadmin.mgi) from your local computer to the web server using an FTP program.

Step 5: View the database administration page in a web browser.

View the database administration page (dbadmin.mgi) in a browser. The first screen of the web-based, administration interface is displayed.

Step 6: Select a database.

Below "Select a database", click the radio button beside the database you wish to search and click the "Records" button. The search screen of the record administration interface displays.



Select a database:

○ Customers
○ Equipment
○ Wanted

Or, enter the name for a new database (if it does not exist, it will be created):

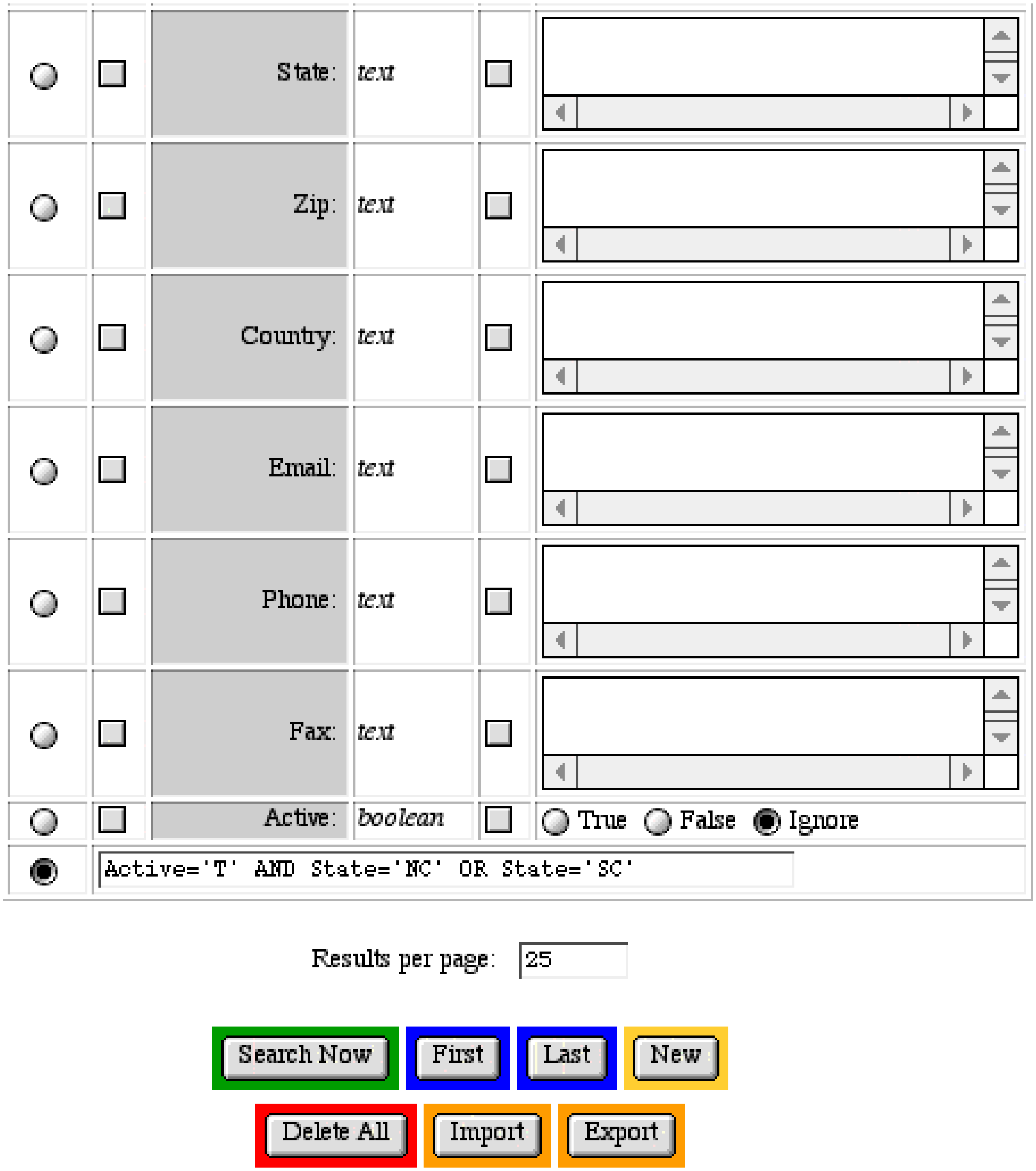
○

Step 7: Enter an advanced search.

In the search screen of the record administration interface, a blank field appears at the bottom of the field list. To perform advanced searches, select the radio button beside this field and enter advanced search strings, then click the "Search Now" button to view search results.

Advanced search strings should contain field name and search criteria in the format **keyFieldName='criteria'**. Valid operators are "=" (equals), ">" (greater than), "<" (less than), ">=" (greater than and equal to), and "<=" (less than and equal to). Field name and search criteria pairs should be separated by the search operator "AND", "OR", or "NOT". To order the results from a search string, include the order by field in the format **ORDER BY keyFieldName**. By default, search results are in asending order (ASC). To reverse the order of results in a search string, specify descending order by including DESC after the ORDER BY field. You may not enter quotes ("") in the value of the search string parameter. The following are example search strings:

```
searchString="LastName='J*' OR LastName='L'
AND Registration='Yes' ORDER BY CustID DESC"

searchString="FirstName='Ken' AND NOT LastName='Barker'
```



○	□	State:	*text*	□	
○	□	Zip:	*text*	□	
○	□	Country:	*text*	□	
○	□	Email:	*text*	□	
○	□	Phone:	*text*	□	
○	□	Fax:	*text*	□	
○	□	Active:	*boolean*	□	○ True ○ False ● Ignore
●	Active='T' AND State='NC' OR State='SC'				

Results per page: 25

Comments and Notes

The mgiEditDatabase tag gives you and anyone who views the page in a web browser access to modify databases. Keep your databases secure by password-protecting the database administration page with an mgiAuthenticate or mgiAuthenticateDB tag.

[Return to the Creating and Populating Databases Menu]

Searching Databases and Displaying Results

In the **Beginner Tutorial** section, learn how to search a database and display a list of results that match the search criteria. In the **Reference** section, view a complete technical reference for each tag used to search databases. In the **Downloads** section, download example code for each beginner and advanced tutorial. In the **FAQs** section, read answers to frequently asked questions.

Beginner Database Search Tutorials

- B1. Searching a Database and Displaying Search Results

Database Search MGI Tag Reference

- mgiSearchDatabase

Database Search Downloads

- B1. Download sample search and results pages (B1Search.zip - 8 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Searching Databases and Displaying Results

Introduction

Search criteria may be drawn from known field values, dynamic form submissions (post arguments), path argument values and/or from page variable values. In this step-by-step tutorial, a keyword search using a form is described. Search results are placed at the location of the mgiSearchDatabase tag on the page with the format that you define in the body of the mgiSearchDatabase tag

MGI Tags

- mgiSearchDatabase

Steps

1. Create a database with one or more indexed fields.
2. Create a search form in a text editor.
3. Save the search form.
4. Create a page to display search results and open the page in a text editor.
5. Insert the mgiSearchDatabase tag and parameters.
6. Save the search results page.
7. FTP the search form and search results page to the web server running MGI.
8. View the search form in a browser and perform a search.

Step 1: Create a database with one or more indexed fields.

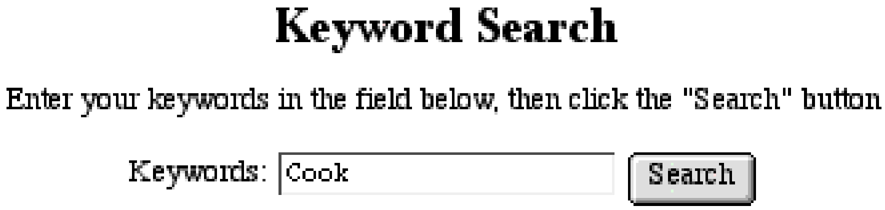
Create a database with one or more indexed fields. A field must be indexed in order to use the field for searching or for ordering search results. Get more information about Creating and Populating databases.

Step 2: Create a search form in a text editor.

Create a keyword search form on your page. Enter a label and a text field. Name the text field "keyword". The name of the text field is the name of the post argument that is created when the text field is submitted using the "post" method. Enter a submit button with the value "Search". Enter HTML <FORM> tags. The form action should be the name of the search results page (which you will create in step 3 - let's call it "results.mgi") and the form method should be "post".

```
<form action="results.mgi" method="post">
<center>
<h2>Keyword Search</h2>
<p>Enter your keywords in the field below, then click the
"Search" button</p>
<p>Keywords: <input type="text" name="keyword">
<input type="submit" name="submit" value="Search"></p>
</center>
</form>
```

This keyword search form will display on the page as:



Keyword Search

Enter your keywords in the field below, then click the "Search" button

Keywords: Cook

Step 3: Save the search form.

Save the search form and name it "search.mgi".

Step 4: Create a page to display search results and open the page in a text editor.

Create a new page to display search results named "results.mgi" and open the search results page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 5: Insert the mgiSearchDatabase tag and parameters.

Insert your cursor in the HTML of the search results page where you want the search results to display. Enter the beginning mgiSearchDatabase tag, databaseName parameter, keyFieldName parameter, pageVariableName parameter, orderByField parameter, resultsPerPage parameter and ending mgiSearchDatabase tag. Enter the page parameter and resultVariableName parameter if the number of search results may exceed the specified results per page. The page and resultVariableName parameters are used to construct dynamic "Previous" and "Next" buttons for browsing search results.

1. In the **databaseName** parameter, enter the case-sensitive name of the database that you wish to search.
2. In the **keyFieldName** parameter, enter the name of the case-sensitive database field name to search for the value of the post argument that follows.
3. In the **pageVariableName** parameter, enter the name of the page variable that contains the search criteria for the preceeding key field name search. In this example, a page

variable is used to add wildcards to the beginning and end of a keyword search.

4. In the **orderByField** parameter, enter the case-sensitive name of the database field to use for ordering search results. If this parameter is not included, search results are dispalyed in the order in which they were entered in the database.
5. In the **resultsPerPage** parameter, enter the number of search results to display on each screen. If this parameter is not included, 25 results are displayed per screen.
6. In the **page** parameter, enter the number for the set of search results to display. In this example, the page number is dynamically entered by embedding a page variable value.
7. In the **resultVariableName** parameter, enter the name to prepend to all result page variables that are used to construct dynamic "Previous" and "Next" buttons. In this examples, all result page variables are prepended with "ProductResults" (e.g., "ProductResults_Page).

In the body of the mgiSearchDatabase tag, enter text, HTML, and MGI database field placeholders to create a template for the search results. The template is repeated for each search result. To retrieve a database field value, use the placeholder format of &mgiDBFieldEnterFieldNameHere; in the body of the mgiSearchDatabase tag. Each placeholder will be replaced with the specific information in that database field for each result. For example, to include values from the ProductName field, you would enter the following in the body of the mgiSearchDatabase tag:

```
&mgiDBFieldProductName;
```

To display the unique internal database ID for a result, use the field name "MGIIDFIELD".

Any text, HTML or MGI tags that are static (i.e., not repeated for each result) should be kept outside of the mgiSearchDatabase tags.

In this example, visitors enter a keyword into a text field to search the "ProductName" field of the database. Since visitors will not enter the exact full product name as it appears in the database, you must append their search criteria with wildcard searches. To accomplish this, enter an mgiSet tag named "SearchCriteria" above the mgiSearchDatabase tag with the value of the keyword and wildcard searches on both sides of the keyword. Use this page variable as the search criteria in the mgiSearchDatabase tag.

If it is possible for the number of search results to exceed the value listed in the resultsPerPage parameter, you must construct dynamic "Previous" and "Next" buttons that allow visitors to browse through different sets of search results. Below the mgiSearchDatabase tags, enter a hidden input for the value of the previous page of results and the value of the next page of results using the result variables (ResultVariableName_PrevPage and ResultVariableName_NextPage). Below the hidden inputs, display a "Previous" button if you are not currently displaying the first set of search results and display a "Next" button if there is more than one page of results available. In addition, enter a hidden input that contains the original search criteria. You must provide this information on each screen of the search results so that mgiSearchDatabase tag can perform a search.

Above the mgiSearchDatabase tags, perform a conditional comparison to determine which set of results should be displayed. If the visitor selected the "Previous" button during a search, then display the previous set of results. If the visitor selected the "Next" button during a search, then display the next set of results. Otherwise, display the first set of results. Set the page value in a variable and embed that variable in the page parameter of the mgiSearchDatabase tag.

In order for the "Previous" and "Next" buttons to function, enter HTML <FORM> tags. The action of the form tag should be the name of the search results page (results.mgi) and the method should be "post".

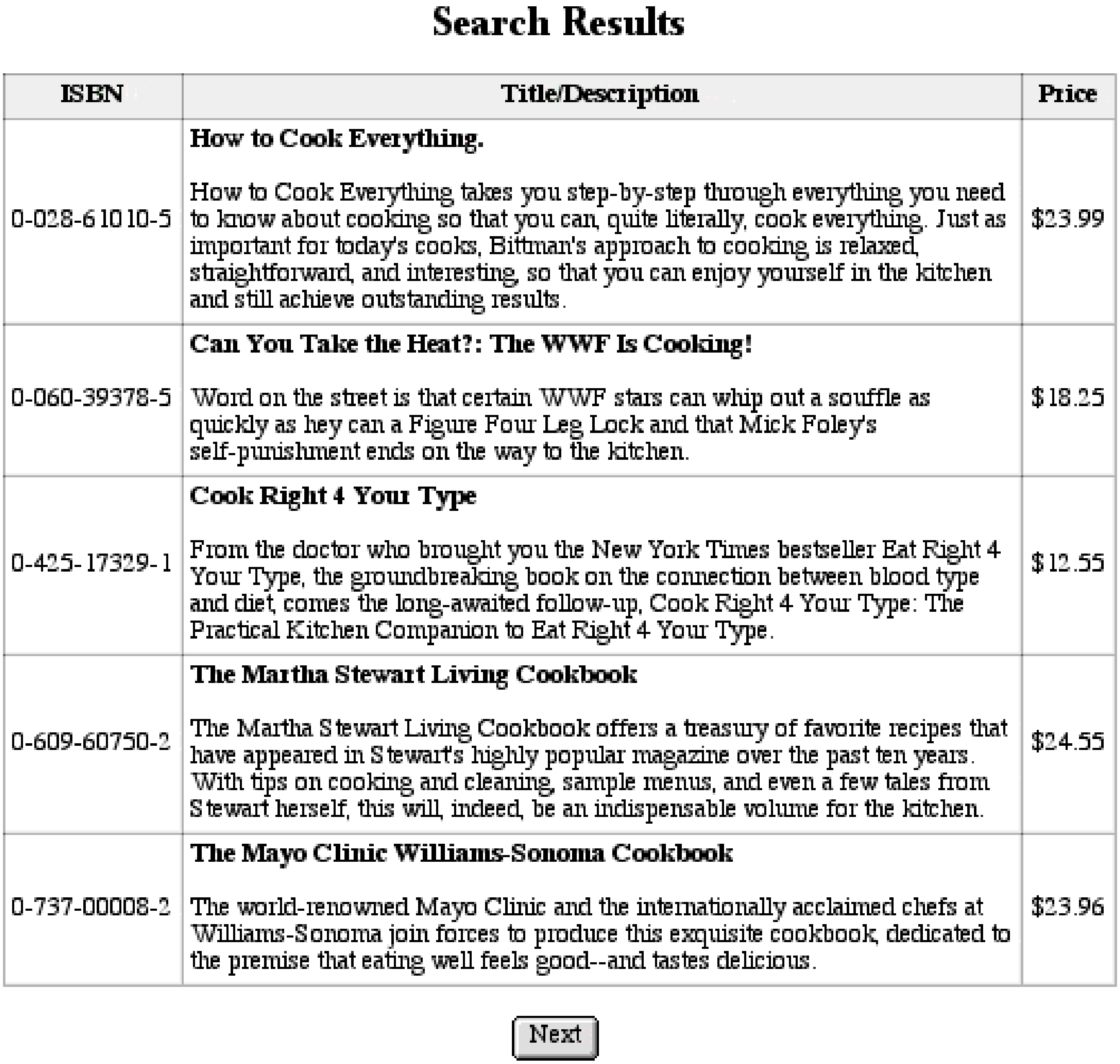
The following is an example search results page including dynamic "Previous" and "Next" buttons. An mgiSet tag is used before the mgiSearchDatabase tag to add wildcard values to the beginning and end of the keyword search in the example.

```
<form action="results.mgi" method="post">

<mgiComment>
Determine which page of results to display
</mgiComment>

<mgiSet name="PageToDisplay" defaultValue="1">
  <mgiInlineIf lhs={mgiPostArgument name="ResultsSet"}
  relationship="equals" rhs="Prev"
  then={mgiPostArgument name="PrevPage"}>

  <mgiInlineIf lhs={mgiPostArgument name="ResultsSet"}
  relationship="equals" rhs="Next"
  then={mgiPostArgument name="NextPage"}>
</mgiSet>

<mgiComment>
Perform the search and display the search results
</mgiComment>

<mgiSet name="SearchCriteria">
  <mgiIf lhs={mgiPostArgument name="keyword"}
relationship="isNotEmpty">
  *<mgiPostArgument name="keyword">*
  </mgiIf>
```

```
</mgiSet>

<center>
<h2>Search Results</h2>
<p>
<table width="500" cellspacing="0"
cellpadding="3" border="1">
<tr bgcolor="#eeeeee">
  <th>ISBN</th>
  <th>Title/Description</th>
  <th>Price</th>
</tr>

<mgiSearchDatabase databaseName="Products"
keyFieldName="ProductName"
pageVariableName="SearchCriteria"
orderByField="ProductID" resultsPerPage="5"
page={mgiGet name="PageToDisplay"}
resultVariableName="KeywordResults">
<tr>
  <td>&mgiDBFieldProductID;</td>
  <td><b>&mgiDBFieldProductName;</b>
  <p>&mgiDBFieldProductDescription;</td>
  <td align="right">$&mgiDBFieldProductPrice;</td>
</tr>
</mgiSearchDatabase>

</table>
</p>

<mgiComment>
Pass search criteria and display
dynamic Previous and Next Buttons
</mgiComment>

<p><input type="hidden" name="keyword"
value={mgiPostArgument name="keyword"}>

<input type="hidden" name="PrevPage"
value={mgiGet name="KeywordResults_PrevPage"}>

<input type="hidden" name="NextPage"
value={mgiGet name="KeywordResults_NextPage"}>

<mgiIf lhs={mgiGet name="KeywordResults_Page"}
```

```
relationship="greaterThan" rhs="1">
<input type="submit" name="ResultsSet" value="Prev">
</mgiIf>

<mgiIf lhs={mgiGet name="KeywordResults_NextPage"}
relationship="greaterThan" rhs="0">
<input type="submit" name="ResultsSet" value="Next">
</mgiIf>

</center>

</form>
```

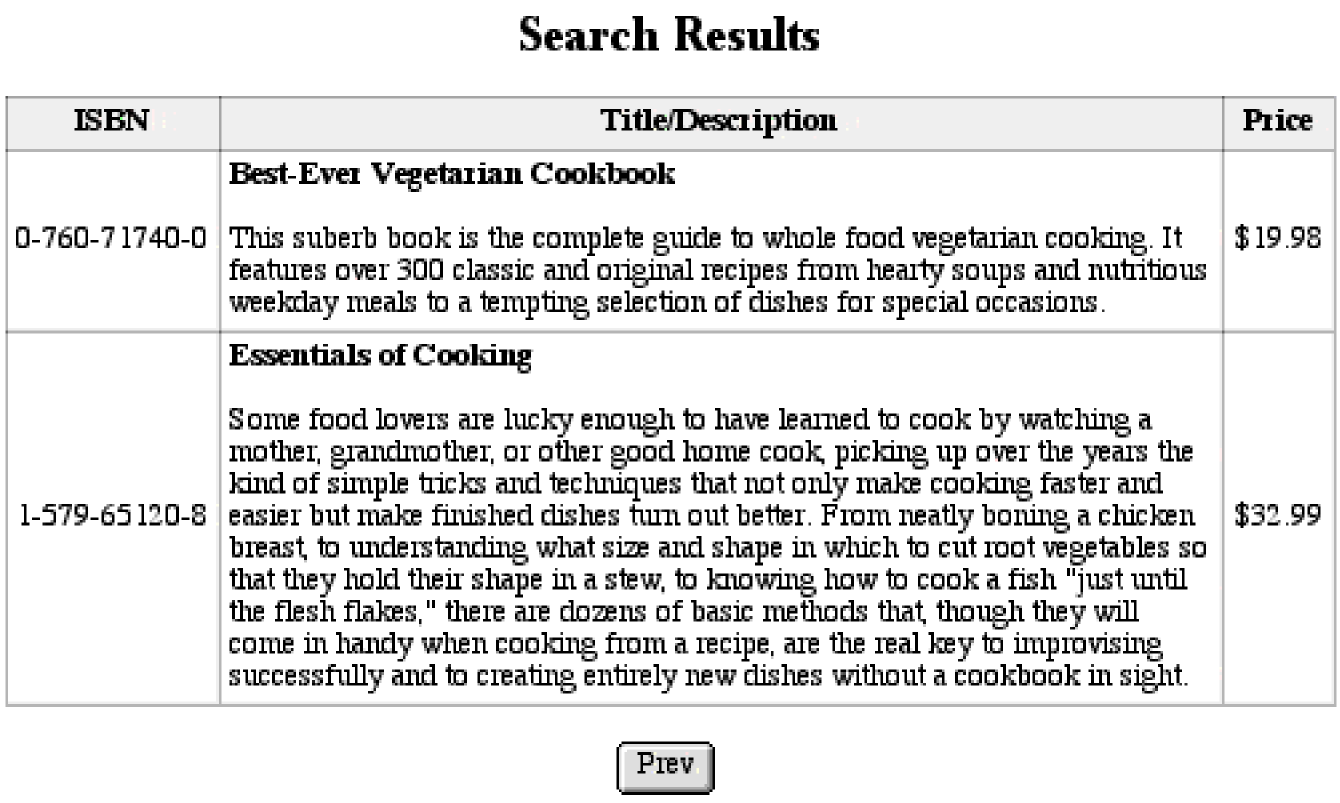
The first screen of results using the keyword "Cook" would display as:



Search Results

ISBN	Title/Description	Price
0-028-61010-5	**How to Cook Everything.** How to Cook Everything takes you step-by-step through everything you need to know about cooking so that you can, quite literally, cook everything. Just as important for today's cooks, Bittman's approach to cooking is relaxed, straightforward, and interesting, so that you can enjoy yourself in the kitchen and still achieve outstanding results.	$23.99
0-060-39378-5	**Can You Take the Heat?: The WWF Is Cooking!** Word on the street is that certain WWF stars can whip out a souffle as quickly as hey can a Figure Four Leg Lock and that Mick Foley's self-punishment ends on the way to the kitchen.	$18.25
0-425-17329-1	**Cook Right 4 Your Type** From the doctor who brought you the New York Times bestseller Eat Right 4 Your Type, the groundbreaking book on the connection between blood type and diet, comes the long-awaited follow-up, Cook Right 4 Your Type: The Practical Kitchen Companion to Eat Right 4 Your Type.	$12.55
0-609-60750-2	**The Martha Stewart Living Cookbook** The Martha Stewart Living Cookbook offers a treasury of favorite recipes that have appeared in Stewart's highly popular magazine over the past ten years. With tips on cooking and cleaning, sample menus, and even a few tales from Stewart herself, this will, indeed, be an indispensable volume for the kitchen.	$24.55
0-737-00008-2	**The Mayo Clinic Williams-Sonoma Cookbook** The world-renowned Mayo Clinic and the internationally acclaimed chefs at Williams-Sonoma join forces to produce this exquisite cookbook, dedicated to the premise that eating well feels good--and tastes delicious.	$23.96

The second set of search results using the keyword "Cook" would display as:



Search Results

ISBN	Title/Description	Price
0-760-71740-0	**Best-Ever Vegetarian Cookbook** This suberb book is the complete guide to whole food vegetarian cooking. It features over 300 classic and original recipes from hearty soups and nutritious weekday meals to a tempting selection of dishes for special occasions.	$19.98
1-579-65120-8	**Essentials of Cooking** Some food lovers are lucky enough to have learned to cook by watching a mother, grandmother, or other good home cook, picking up over the years the kind of simple tricks and techniques that not only make cooking faster and easier but make finished dishes turn out better. From neatly boning a chicken breast, to understanding what size and shape in which to cut root vegetables so that they hold their shape in a stew, to knowing how to cook a fish "just until the flesh flakes," there are dozens of basic methods that, though they will come in handy when cooking from a recipe, are the real key to improvising successfully and to creating entirely new dishes without a cookbook in sight.	$32.99

Step 6: Save the search results page.

Save the changes you have made to the search results page.

Step 7: FTP the search form and search results page to the web server running MGI.

Upload the search form and search results page to the web server using an FTP program.

Step 8: View the search form in a browser and perform a search.

View the search form in a web browser. Enter a keyword in the field and click the "Search" button. The results matching your criteria are displayed. If more than 5 results are found, a "Next" button is also displayed. Clicking the "Next" button will display the next set of results.

[Return to the Searching Databases and Displaying Results]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

Shopping Online

In the **Beginner Tutorial** section, learn how to integrate a shopping basket system with hard-coded products and products displayed from a database search. In the **Advanced Tutorial** section, learn online shopping techniques including price, tax and shipping rules, customizing the shopping basket display, encrypting orders, implementing inventory control, and authorizing credit cards via Accesspoint and other credit card gateways. In the **Reference** section, view a complete technical reference for each tag used for online shopping. In the **Downloads** section, download example code for each beginner and advanced tutorial. In the **FAQs** section, read answers to frequently asked questions.

Beginner Shopping Online Tutorials

- B1. Default Shopping Basket, Checkout, Confirmation and Order Processing

---

Advanced Shopping Online Tutorials

- A1. Price, Shipping and Tax Item Rules
- A2. Price, Shipping and Tax Basket Rules
- A3. Customize the Shopping Basket Display
- A4. Customize the Payment, Billing and Shipping Information Display
- A5. Customize the Order Confirmation Display
- A6. Customize the Order Email
- A7. Encrypting Orders
- A8. Using Inventory Control
- A9. Authorizing Credit Cards via a Gateway

Shopping Online MGI Tag Reference

- mgiButton
- mgiBuyMe
- mgiCollectUserInfo
- mgiConfirmOrder
- mgiSendOrder
- mgiShoppingBasket

Shopping Online Downloads

- B1. Download example hard-coded and database-driven product, shopping basket, checkout, confirmation, and order processing pages (B1Shop.zip - 8 KB)
- A1. Download an example hard-coded shopping basket with price, shipping, and tax item rules (A1ItemRules.zip - 300 KB)
- A2. Download an example hard-coded shopping basket with price, shipping, and tax basket rules (A2BasketRules.zip - 300 KB)

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Default Shopping Basket, Checkout, Confirmation and Order Processing

Introduction

The default shopping basket is integrated with hard-coded products or database products using the mgiBuyMe tag. Products are added to the shopping basket page where the contents of the shopping basket displays and the visitor may modify the shopping basket or proceed to the secure check out page. On the check out page, the visitor is prompted for payment, billing, and shipping information, then proceeds to the order confirmation page. On the order confirmation page, the contents of the shopping basket (including any tax and shipping) are displayed with the visitor's payment, billing and shipping entries. Upon proceeding to the final send order page, the customer's order is converted to an email and sent to the specified address.

MGI Tags

- mgiButton
- mgiBuyMe
- mgiCollectUserInfo
- mgiConfirmOrder
- mgiSendOrder
- mgiShoppingBasket

Steps

1. Create a shopping basket administration page in a text editor.
2. Insert the mgiShoppingBasket tag in admin mode.
3. Save the shopping basket administration page.
4. FTP the shopping basket administration page to the web server running MGI.
5. View the shopping basket administration page in a web browser.
6. Create a shopping basket handle.
7. Create product pages and open them in a text editor.
8. Insert an mgiBuyMe tag and a submit button for each product.

9. Save the product pages.
10. Create a shopping basket page and open it in a text editor.
11. Insert the mgiShoppingBasket and mgiButton tags.
12. Save the shopping basket page.
13. Create a check out page and open it in a text editor.
14. Insert the mgiCollectUserInfo and mgiButton tags.
15. Save the check out page.
16. Create a confirm order page and open it in a text editor.
17. Insert the mgiConfirmOrder and mgiButton tags.
18. Save the confirm order page.
19. Create an order processing page and open it in a text editor.
20. Insert the mgiSendOrder tag.
21. Save the order processing page.
22. Tokenize all other pages of the web site.
23. Save the web site pages.
24. FTP all pages to the web server running MGI.
25. View a product page in a web browser and purchase a product.

Step 1: Create a shopping basket administration page in a text editor.

Create a new page in a text editing program to display the web-based, shopping basket administration interface.

Step 2: Insert the mgiShoppingBasket tag in admin mode.

On the shopping basket administration page, enter the mgiShoppingBasket tag and mode parameter. In the mode parameter, enter "admin".

```
<mgiShoppingBasket mode="admin">
</mgiShoppingBasket>
```

Step 3: Save the shopping basket administration page.

Save the shopping basket administration page and name it "sbadmin.mgi".

Step 4: FTP the shopping basket administration page to the web server running MGI.

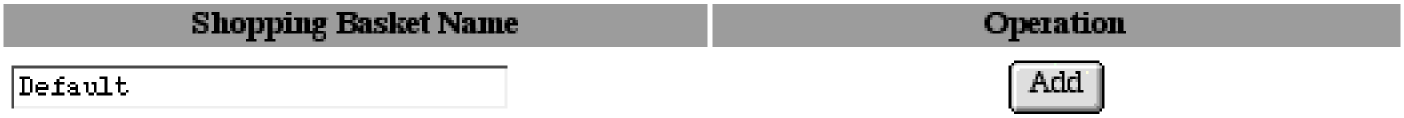
Upload the shopping basket administration page (sbadmin.mgi) from your local computer to the web server using an FTP program.

Step 5: View the shopping basket administration page in a web browser.

View the shopping basket administration page (sbadmin.mgi) in a web browser. The first screen of the web-based, administration interface is displayed.

Step 6: Create a shopping basket handle.

Below the column "Shopping Basket Name", enter the name of a shopping basket handle and click the "Add" button. In this example, the shopping basket handle is named "Default". The shopping basket handle determines the configuration of a shopping basket (you may have and use multiple shopping basket handles in one region). For a default shopping basket configuration, the handle does not require customization and merely needs to be created. Once the handle is created, note its name and close the shopping basket administration page.



Shopping Basket Name	Operation
Default	Add

Step 7: Create product pages and open them in a text editor.

Create pages to display products for sale. You may choose any design and layout for products including pictures, descriptions, pricing information, etc. Insert placeholders for a quantity box for each product and a submit button. The default quantity box is 3 characters wide. Finally, open your product pages in a text editing program that allows you to view and modify the HTML and code of the page.

Step 8: Insert an mgiBuyMe tag and a submit button for each product.

Replace the quantity box placeholder for each product with an mgiBuyMe tag, productID parameter, name parameter, and price parameter. In the productID parameter, enter a unique code for the product or embed the product's unique ID from a database search. Product information is tracked through the shopping basket by the productID. In the name parameter, enter a short description of the product or embed the product's short description from a database search. In the price parameter, enter the per item price of each product as a decimal number or embed the product's price from a database search. Do not enter characters such as dollar signs in the price parameter.

Replace the submit button placeholder with an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button (e.g., "Add to Shopping Basket").

A form action adds the quantity entered in the quantity box(s) to the shopping basket page for processing when the submit button is clicked. Enter the name of the shopping baket page (shoppingbasket.mgi) in the action parameter of the <FORM> tag. Enter "post" in the method parameter of the <FORM> tag. Enclose the mgiBuyMe tag and "Add to Shopping Basket" submit button with HTML form tags. If you want to allow customers to add multiple items to a shopping basket at one time, enclose all mgiBuyMe tags and a single submit button or multiple submit buttons with HTML form tags.

The default shopping basket uses tokens to track the purchases of individual customers. Enter one beginning mgiToken tag before all products and submit buttons, and enter one ending mgiToken tag after all products and submit buttons. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example hard-coded product page. See an example database search page below.

```
<mgiToken>

<form action="shoppingbasket.mgi" method="post">

<h2>Products</h2>

<p>
<table width="500" cellspacing="0"
cellpadding="3" border="1">
<tr bgcolor="#eeeeee">
  <th>ISBN</th>
  <th>Title/Description</th>
  <th>Price</th>
  <th>Quantity</th>
</tr>
<tr>
  <td>0-394-53264-3</td>
  <td><b>The Way to Cook</b> <p>The magnum opus:
half a million copies sold and climbing. More than
800 recipes and over 600 color photographs. Photographs
by Brian Leatart and Jim Scherer.</td>
  <td align="right">$52.00</td>
  <td align="center"><mgiBuyMe productID="0-394-53264-3"
name="The Way to Cook" price="52.00"></td>
</tr>
<tr>
  <td>0-737-02041-5</td>
  <td><b>The Kid's Cookbook : A Great Book for Kids
Who Love to Cook</b> <p>Designed especially for
kids ages 9 and up, The Kids Cookbook is packed with
```

```
kid-friendly recipes-and great photos to match. Each
recipe includes an equipment list and all instruction
young cooks need to have a great time whipping up
something delicious. (Super-easy recipes are flagged,
so kids can choose their challenges. </td>
  <td align="right">$15.96</td>
  <td align="center"><mgiBuyMe productID="0-737-02041-5"
name="The Kid's Cookbook : A Great Book for Kids Who
Love to Cook" price="15.96"></td>
</tr>
<tr>
  <td colspan="4" align="right">
  <mgiButton value="Add to Shopping Basket"></td>
</tr>
</table>
</p>

</form>

</mgiToken>
```

The following is an example database search results page that embeds database information in the mgiBuyMe tags. Notice there are 2 sets of HTML form tags. The first set adds products to the database and the second set posts "Next" and "Previous" results.

```
<mgiToken>

<form action="shoppingbasket.mgi" method="post">

<mgiComment>
Determine which page of results to display
</mgiComment>

<mgiSet name="PageToDisplay" defaultValue="1">
  <mgiInlineIf lhs={mgiPostArgument name="ResultsSet"}
  relationship="equals" rhs="Prev"
  then={mgiPostArgument name="PrevPage"}>

  <mgiInlineIf lhs={mgiPostArgument name="ResultsSet"}
  relationship="equals" rhs="Next"
  then={mgiPostArgument name="NextPage"}>
</mgiSet>

<mgiComment>
Perform the search and display the search results
```

```
</mgiComment>

<mgiSet name="SearchCriteria">
  <mgiIf lhs={mgiPostArgument name="keyword"}
relationship="isNotEmpty">
  *<mgiPostArgument name="keyword">*
  </mgiIf>
</mgiSet>

<center>
<h2>Search Results</h2>
<p>
<table width="500" cellspacing="0"
cellpadding="3" border="1">
<tr bgcolor="#eeeeee">
  <th>ISBN</th>
  <th>Title/Description</th>
  <th>Price</th>
  <th>Quantity</th>
</tr>

<mgiSearchDatabase databaseName="Products"
keyFieldName="ProductName"
pageVariableName="SearchCriteria"
orderByField="ProductID" resultsPerPage="5"
page={mgiGet name="PageToDisplay"}
resultVariableName="KeywordResults">
<tr>
  <td>&mgiDBFieldProductID;</td>
  <td><b>&mgiDBFieldProductName;</b>
      <p>&mgiDBFieldProductDescription;</td>
  <td align="right">$&mgiDBFieldProductPrice;</td>
  <td align="center"><mgiBuyMe
productID="&mgiDBFieldProductID;"
name="&mgiDBFieldProductName;"
price="&mgiDBFieldProductPrice;"></td>
</tr>
</mgiSearchDatabase>

<tr>
  <td colspan="4" align="right">
<mgiButton value="Add to Shopping Basket"></td>
</tr>
</table>
</p>
```

```
</form>

<mgiComment>
Pass search criteria and display
dynamic Previous and Next Buttons
</mgiComment>

<form action="results.mgi" method="post">

<p><input type="hidden" name="keyword"
value={mgiPostArgument name="keyword"}>

<input type="hidden" name="PrevPage"
value={mgiGet name="KeywordResults_PrevPage"}>

<input type="hidden" name="NextPage"
value={mgiGet name="KeywordResults_NextPage"}>

<mgiIf lhs={mgiGet name="KeywordResults_Page"}
relationship="greaterThan" rhs="1">
<input type="submit" name="ResultsSet" value="Prev">
</mgiIf>

<mgiIf lhs={mgiGet name="KeywordResults_NextPage"}
relationship="greaterThan" rhs="0">
<input type="submit" name="ResultsSet" value="Next">
</mgiIf>

</center>

</form>
```

Step 9: Save the product pages.

Save the changes you have made to the product pages.

Step 10: Create a shopping basket page and open it in a text editor.

Create a page named "shoppingbasket.mgi" to display the contents of a customer's shopping basket and open the page in a text editing program that allows you to view and modify the HTML and code.

Step 11: Insert the mgiShoppingBasket and mgiButton tags.

Insert your cursor in the HTML of the shopping basket page where you want the shopping basket to display and enter the beginning mgiShoppingBasket tag, handle parameter and ending mgiShoppingBasket tag. In the handle parameter, enter the name of the default handle you created in the shopping basket administration (i.e., "Default").

Below the mgiShopping basket tag, enter an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to modify shopping basket quantities (e.g., "Modify Quantity").

Below the mgiShoppingBasket tag and modify quantity button, enter a second mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to proceed to the check out page (e.g., "Check Out").

On the shopping basket page, form actions perform two functions. One form action modifies the contents of the shopping basket (i.e., changes quantities). Enclose the mgiShoppingBasket tag and mgiButton tag used to modify the shopping basket with HTML form tags. Enter the name of the shopping basket page (shoppingbasket.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag.

Another form action links customers to the check out page to enter payment, billing and shipping information. Enclose the mgiButton tag used to check out with HTML form tags. Enter the name of the check out page (checkout.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag. Note: To use a secure server to collect payment information, enter the URL to a secure server running MGI in the action parameter of the <FORM> tag (e.g., https://secure.domain.com/checkout.mgi). You may substitute a text link for the check out button and form tags.

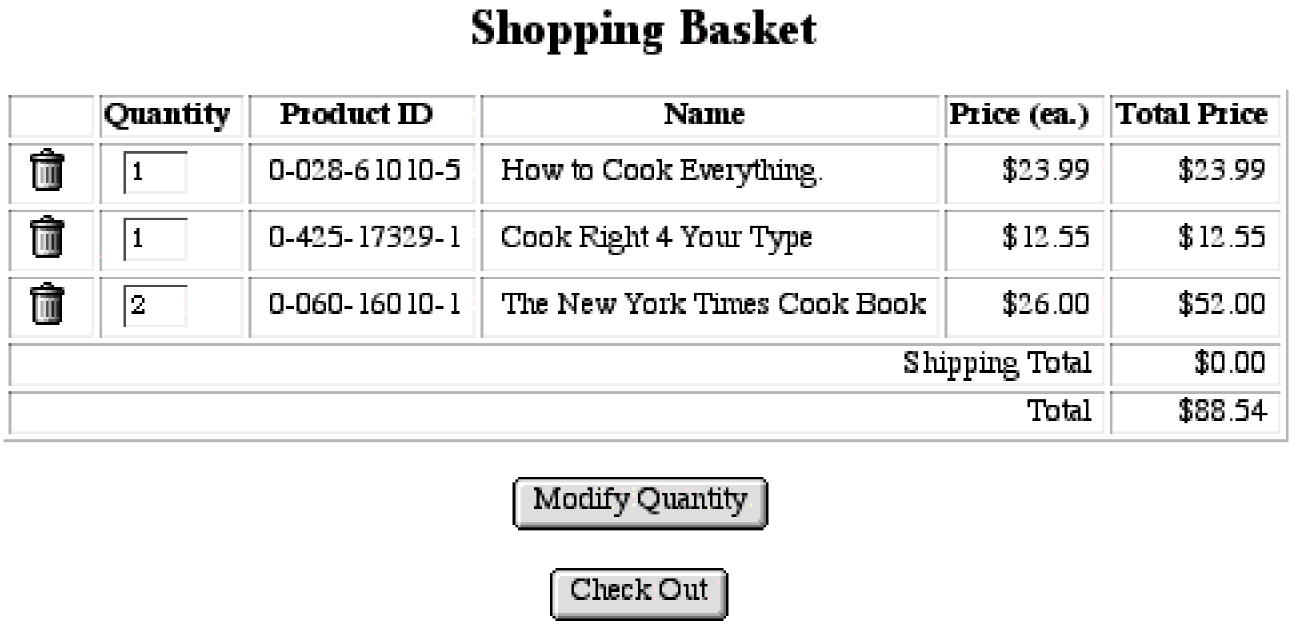
The default shopping basket uses tokens to track the purchases of individual customers. Enter a beginning mgiToken tag before the first HTML form tag and enter an ending mgiToken tag after the last closing HTML form tag. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example shopping basket page.

```
<mgiToken>

<center>

<h2>Shopping Basket</h2>
```

```
<form action="shoppingbasket.mgi" method="post">

<p><mgiShoppingBasket handle="Default">
</mgiShoppingBasket>

<p><mgiButton value="Modify Quantity">

</form>

<form action="checkout.mgi" method="post">

<mgiButton value="Check Out">

</form>

</center>

</mgiToken>
```

The default shopping basket contents display in a table with columns for product removal, product quantity, product ID, product name, product price, multiplied product subtotals and order total:



Shopping Basket

	Quantity	Product ID	Name	Price (ea.)	Total Price
🗑	1	0-028-61010-5	How to Cook Everything.	$23.99	$23.99
🗑	1	0-425-17329-1	Cook Right 4 Your Type	$12.55	$12.55
🗑	2	0-060-16010-1	The New York Times Cook Book	$26.00	$52.00
				Shipping Total	$0.00
				Total	$88.54

Step 12: Save the shopping basket page.

Save the changes you have made to the shopping basket page (shoppingbasket.mgi).

Step 13: Create a check out page and open it in a text editor.

Create a page named "checkout.mgi" to collect payment, billing and shipping information from the customer and open the page in a text editing program that allows you to modify the HTML and code of the page.

Step 14: Insert the mgiCollectUserInfo and mgiButton tags.

Insert your cursor in the HTML of the check out page where you want the payment, billing and shipping information tables to display and enter the mgiCollectUserInfo tag, handle parameter and shoppingBasketURL parameter. In the handle parameter, enter the name of the default handle you created in the shopping basket administration (i.e., "Default"). In the shoppingBasketURL parameter, enter the full URL to the region where items were added to the shopping basket (and thus where the internal MGI shopping basket database was created and populated).

Below the mgiCollectUserInfo tag, enter an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to proceed to the order confirmation page (e.g., "Confirm Order").

Form actions on the check out page link customers to the order confirmation page to review their order and customer information. Enclose the mgiCollectUserInfo tag, mgiButton tag, and any custom form elements with HTML form tags. Enter the name of the order confirmation page (confirmorder.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag.

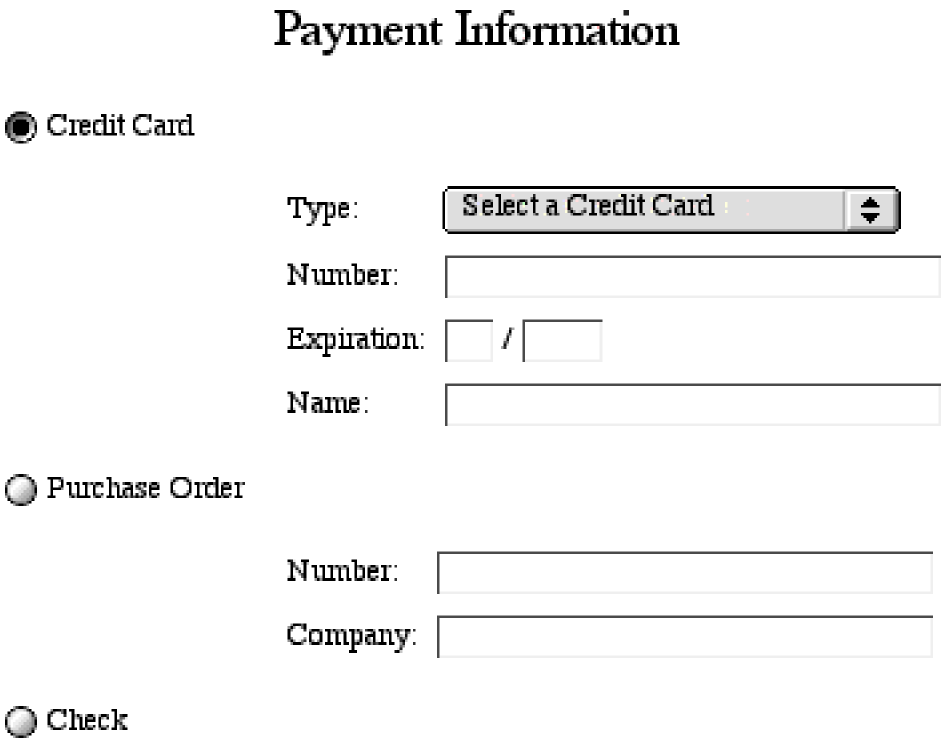
The default shopping basket uses tokens to track the purchases of individual customers. Enter a beginning mgiToken tag before the beginning HTML form tag and enter an ending mgiToken tag after the ending HTML form tag. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example check out page.

```
<mgiToken>

<form action="confirmorder.mgi" method="post">

<center>

<mgiCollectUserInfo handle="Default"
shoppingBasketURL="http://www.domain.com/shop/">

<p><mgiButton value="Confirm Order">

</center>

</form>
```

```
</mgiToken>
```

The default check out page display payment, billing and shipping information in tables. Custom form elements appear as you format them.



Payment Information

● Credit Card

Type: Select a Credit Card

Number:

Expiration: /

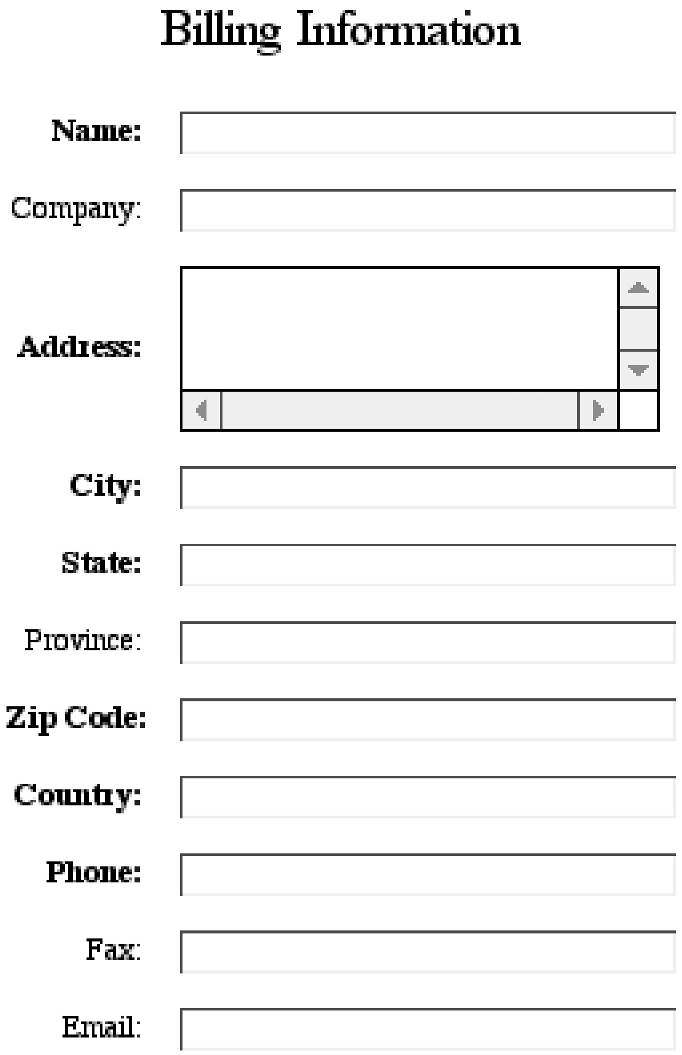
Name:

○ Purchase Order

Number:

Company:

○ Check



Billing Information

Name:

Company:

Address:

City:

State:

Province:

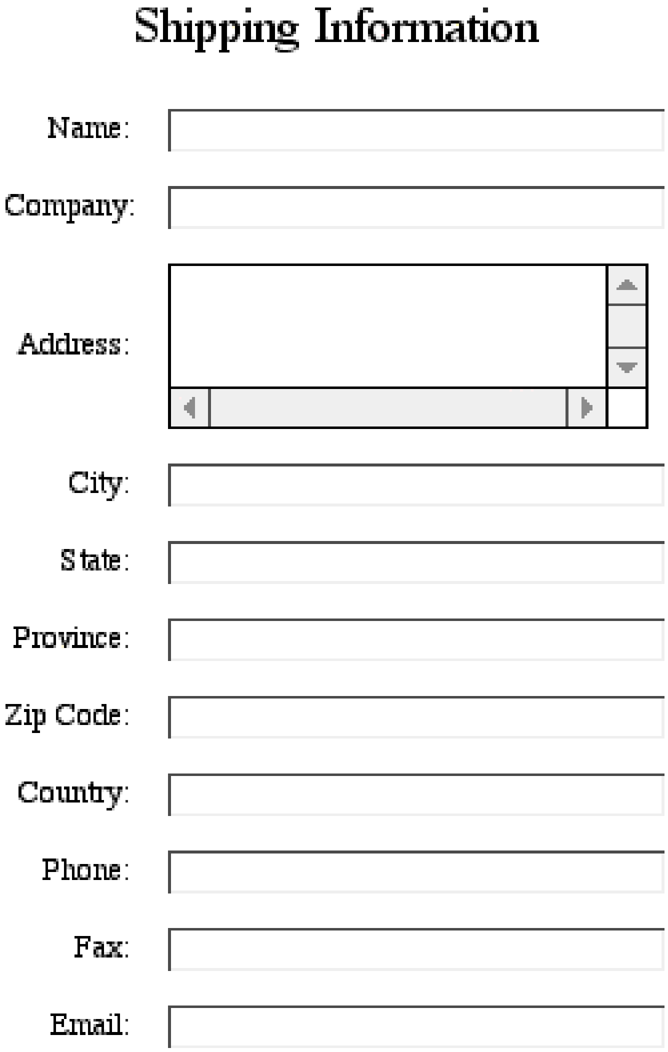
Zip Code:

Country:

Phone:

Fax:

Email:



Shipping Information

Name:

Company:

Address:

City:

State:

Province:

Zip Code:

Country:

Phone:

Fax:

Email:

Step 15: Save the check out page.

Save the changes you have made to the check out page.

Step 16: Create a confirm order page and open it in a text editor.

Create a page named "confirmorder.mgi" to present the final shopping basket, payment information, billing information, and shipping information for review. Open the confirm order page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 17: Insert the mgiConfirmOrder and mgiButton tags.

Insert your cursor in the HTML of the confirm order page where you want the shopping basket, payment, billing, shipping, and additional information tables to display and enter the beginning mgiConfirmOrder tag, handle parameter, shoppingBasketURL parameter, and ending mgiConfirmOrder tag. In the handle parameter, enter the name of the default handle you created in the shopping basket administration (i.e., "Default"). In the shoppingBasketURL parameter, enter the full

URL to the region where items were added to the shopping basket (and thus where the internal MGI shopping basket database was created and populated).

Below the mgiConfirmOrder tag, enter an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to proceed to the final order processing page (e.g., "Complete Order").

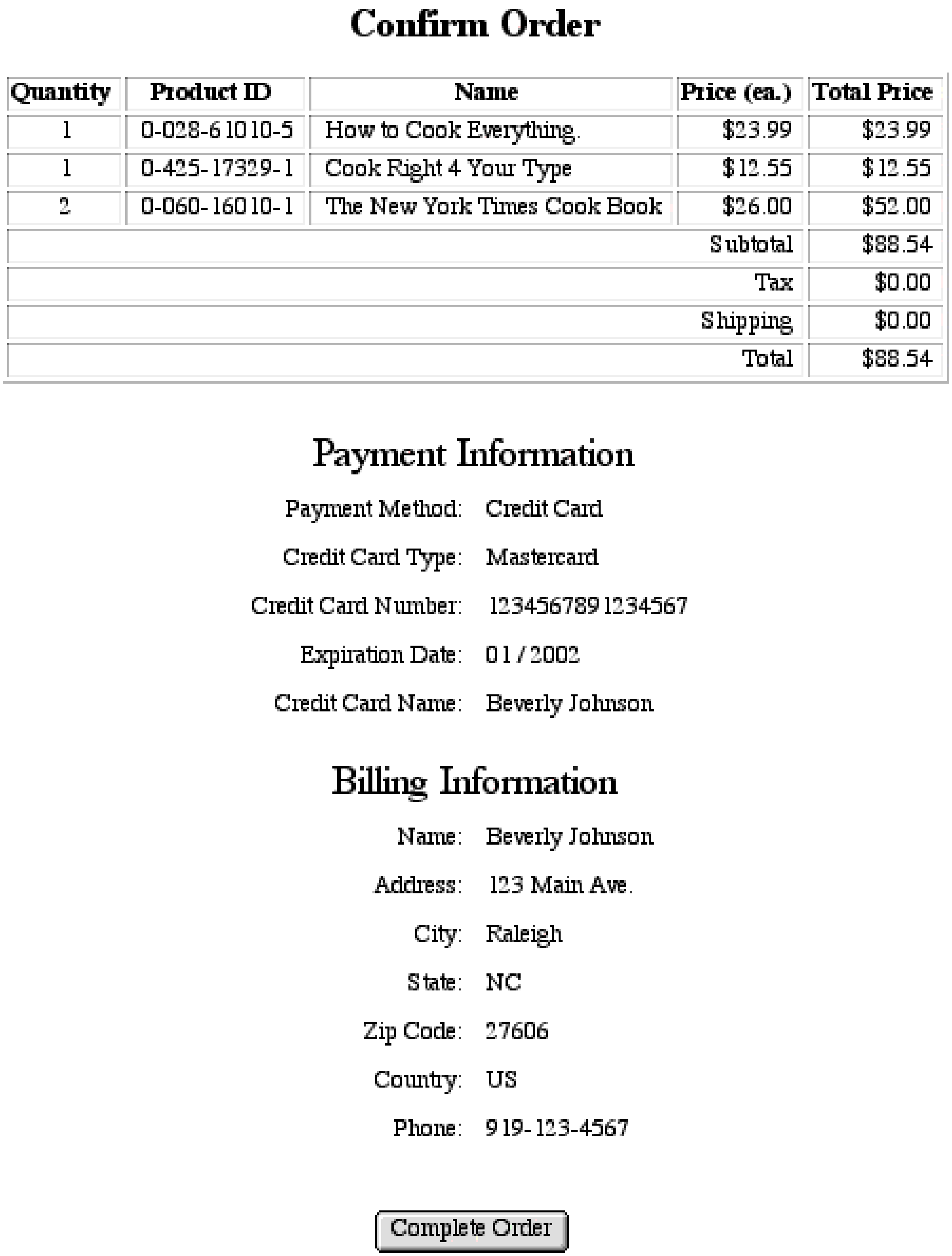
Form actions on the confirm order page link customers to the order processing page to complete and send their order. Enclose the mgiConfirmOrder tag and mgiButton tag with HTML form tags. Enter the name of the order processing page (completeorder.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag.

The default shopping basket uses tokens to track the purchases of individual customers. Enter a beginning mgiToken tag before the beginning HTML form tag and enter an ending mgiToken tag after the ending HTML form tag. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example confirm order page.

```
<mgiToken>

<form action="completeorder.mgi" method="post">

<center>

<mgiConfirmOrder handle="Default"
shoppingBasketURL="http://www.domain.com/shop/">
</mgiConfirmOrder>

<p><mgiButton value="Complete Order">

</center>

</form>

</mgiToken>
```

The default confirm order page displays completed elements of the payment, billing and shipping information.



Confirm Order

Quantity	Product ID	Name	Price (ea.)	Total Price
1	0-028-61010-5	How to Cook Everything.	$23.99	$23.99
1	0-425-17329-1	Cook Right 4 Your Type	$12.55	$12.55
2	0-060-16010-1	The New York Times Cook Book	$26.00	$52.00
			Subtotal	$88.54
			Tax	$0.00
			Shipping	$0.00
			Total	$88.54

Payment Information

Payment Method: Credit Card

Credit Card Type: Mastercard

Credit Card Number: 1234567891234567

Expiration Date: 01 / 2002

Credit Card Name: Beverly Johnson

Billing Information

Name: Beverly Johnson

Address: 123 Main Ave.

City: Raleigh

State: NC

Zip Code: 27606

Country: US

Phone: 919-123-4567

Step 18: Save the confirm order page.

Save the changes you have made to the confirm order page.

Step 19: Create an order processing page and open it in a text editor.

Create an page named "completeorder.mgi" to present a "thank you for purchasing" message to customers and to send a formatted email of the order to a specified address. Open the order processing page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 20: Insert the mgiSendOrder tag.

Enter a "thank you" message to display to customers when they complete their order.

At any place on the order processing page, enter a beginning mgiSendOrder tag, handle parameter, shoppingBasketURL parameter, to parameter, from parameter, mailServer parameter, subject parameter, and ending mgiSendOrder tag.

In the handle parameter, enter the name of the default handle you created in the shopping basket administration (i.e., "Default"). In the shoppingBasketURL parameter, enter the full URL to the region where items were added to the shopping basket (and thus where the internal MGI shopping basket database was created and populated).

In the "to" parameter, enter the email address to receive shopping basket orders. In the "from" parameter, enter the email address that appears in the "from" line of shopping basket orders. In the mailServer parameter, enter the address for the outgoing SMTP mail server of your domain (e.g., mail.domain.com). In the subject parameter, enter the subject of the order email. The mgiSendOrder tag does not display information to the customer.

The following is an example order processing page.

```
<center>

<h2>Order Complete</h2>

<p>Thank you for ordering.  Your order has been
processed and you should receive the merchandise
within 2 to 4 weeks.

<mgiSendOrder handle="Default"
shoppingBasketURL="http://www.domain.com/shop/"
to="sales@domain.com" from="webmaster@domain.com"
mailServer="mail.domain.com"
subject="Online Order">
</mgiSendOrder>

</center>
```

The default order email is formatted with payment, billing, shipping, and order information.

```
Payment Information
-------------------
 Payment Method:  Credit Card
           Type:  Mastercard
         Number:  1234567891234567
Expiration Date:  01/2002
           Name:  Beverly Johnson
Billing Information
-------------------
           Name:  Beverly Johnson
        Address:  123 Main Ave.
           City:  Raleigh
          State:  NC
       Zip Code:  27606
        Country:  US
          Phone:  919-123-4567
Product Information
--------------------
       Quantity:  1
     Product ID:  0-028-61010-5
           Name:  How to Cook Everything.
    Total Price:  $23.99

       Quantity:  1
     Product ID:  0-425-17329-1
           Name:  Cook Right 4 Your Type
    Total Price:  $12.55

       Quantity:  2
     Product ID:  0-060-16010-1
           Name:  The New York Times Cook Book
    Total Price:  $52.00

       Subtotal:  $88.54
            Tax:  $0.00
       Shipping:  $0.00
          Total:  $88.54
```

Step 21: Save the order processing page.

Save the changes you have made to the order processing page.

Step 22: Tokenize all other pages of the web site.

The default shopping basket uses tokens to track the purchases of individual customers. Using a text editor, enter a beginning mgiToken tag and ending mgiToken tag on all additional pages of the web site. The mgiToken tags should enclose all links on the pages including HREFs and FORM actions.

```
<mgiToken>
All tags and links appear here.
</mgiToken>
```

Step 23: Save the web site pages.

Save the changes that you make as you tokenize each page.

Step 24: FTP all pages to the web server running MGI.

Upload all pages of the web site to the web server using an FTP program. If you are using a secure server, upload **only** the check out, order confirmation and order processing pages to the secure portion of your web site. Do not attempt to use the shopping basket page on a secure server.

Step 25: View a product page in a web browser and purchase a product.

View a hard-code or database-driven product page in a web browser. Enter a quantity to purchase in the quantity text field and click the "Add to Shopping Basket" button. The item(s) appears in the shopping basket display on the shopping basket page. Click the "Check Out" button to proceed to the check out page. On the check out page, enter a payment choice, billing information and shipping information (if it is different). Click the "Confirm Order" button to proceed to the confirm order page. On the confirm order page, review your order, payment information, billing information and shipping information, then click the "Complete Order" button to complete the ordering process. The message from the order processing page is displayed and the shopping basket order is formatted and emailed.

[Return to the Shopping Online Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Price, Shipping and Tax Item Rules

Introduction

Rules are algorithms that MGI uses to calculate product price specials, shipping and tax. Rules can be applied on a per item basis (item rules) or across all items in the shopping basket (basket rules). Item rules apply to individual items and are entered in the mgiBuyMe tag for the specific item. Item tax rules also require parameters in the mgiConfirmOrder and mgiSendOrder tags. Basket rules apply across all items in a shopping basket and are entered in the mgiShoppingBasket, mgiConfirmOrder, and mgiSendOrder tags.

Item rules and basket rules are mutually exclusive for each type of rule. For example, you may not have an item price rule and a basket price rule, however you may have an item price rule and a basket shipping rule. All rules are configured in the web-based administration interface of the mgiShoppingBasket tag.

Price rules specific to items include specials such as "buy one get one free" and percentage discounts. Using the mgiBuyMe tag, a shipping price can be specified for each product, but shipping rules allow you to add base shipping costs for each product and shipping charges or for specific items (e.g., "$1 S&H" or "free shipping when you buy 2 or more of this item"). Tax rules help you calculate local, state and federal taxes or any other tax on a specific item based on a customer's location. Tax rules can also be chained in a specific order for multiple taxes.

The following example is a shopping site for art posters that illustrates price, shipping and tax rules for individual items (i.e. "item rules"). This example is hard-coded products. The same principles would apply to mgiBuyMe tags with embedded product information from a database.

MGI Tags

- mgiButton
- mgiBuyMe
- mgiCollectUserInfo
- mgiConfirmOrder
- mgiSendOrder
- mgiShoppingBasket

Steps

1. Create a shopping basket administration page in a text editor.
2. Insert the mgiShoppingBasket tag in admin mode.

3. Save the shopping basket administration page.
4. FTP the shopping basket administration page to the web server running MGI.
5. View the shopping basket administration page in a web browser.
6. Create a shopping basket handle.
7. Configure item price rules.
8. Configure item shipping rules.
9. Configure item tax rules.
10. Create product pages and open them in a text editor.
11. Insert an mgiBuyMe tag, rule parameters and a submit button for each product.
12. Save the product pages.
13. Create a shopping basket page and open it in a text editor.
14. Insert the mgiShoppingBasket and mgiButton tags.
15. Save the shopping basket page.
16. Create a check out page and open it in a text editor.
17. Insert the mgiCollectUserInfo and mgiButton tags.
18. Save the check out page.
19. Create a confirm order page and open it in a text editor.
20. Insert the mgiConfirmOrder tag, customerLocation parameter and mgiButton tag.
21. Save the confirm order page.
22. Create an order processing page and open it in a text editor.
23. Insert the mgiSendOrder tag and customerLocation parameter.
24. Save the order processing page.
25. Tokenize all other pages of the web site.
26. Save the web site pages.
27. FTP all pages to the web server running MGI.
28. View a product page in a web browser and purchase a product.

Step 1: Create a shopping basket administration page in a text editor.

Create a new page in a text editing program to display the web-based, shopping basket administration interface.

Step 2: Insert the mgiShoppingBasket tag in admin mode.

On the shopping basket administration page, enter the mgiShoppingBasket tag and mode parameter. In the mode parameter, enter "admin".

```
<mgiShoppingBasket mode="admin"> </mgiShoppingBasket>
```

Step 3: Save the shopping basket administration page.

Save the shopping basket administration page and name it "sbadmin.mgi".

Step 4: FTP the shopping basket administration page to the web server running MGI.

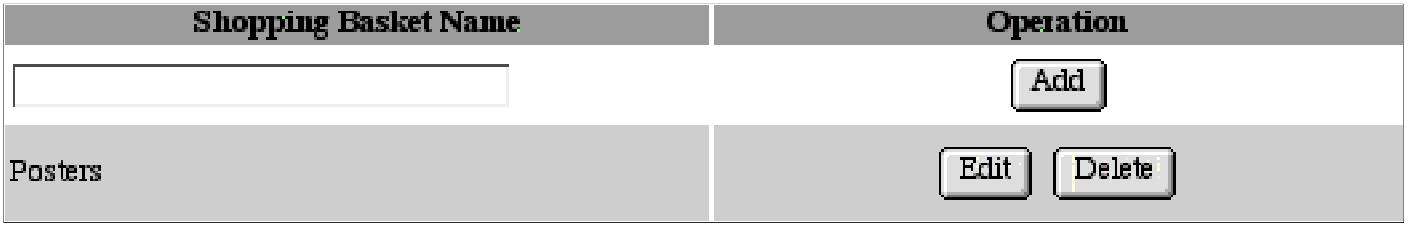
Upload the shopping basket administration page (sbadmin.mgi) from your local computer to the web server using an FTP program.

Step 5: View the shopping basket administration page in a web browser.

View the shopping basket administration page (sbadmin.mgi) in a web browser. The first screen of the web-based, administration interface is displayed.

Step 6: Create a shopping basket handle.

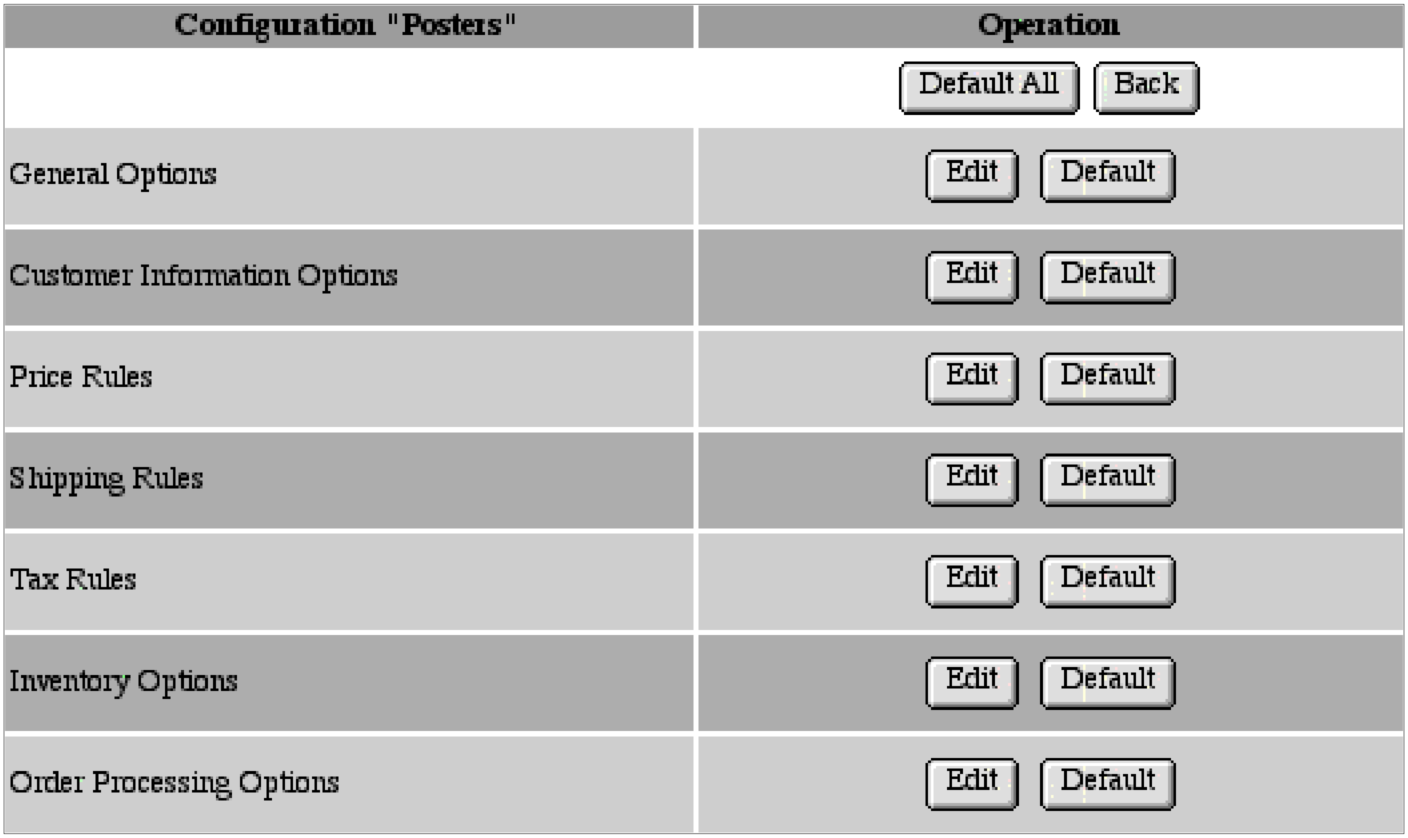
Below the column "Shopping Basket Name", enter the name of a shopping basket handle and click the "Add" button. In this example, the shopping basket handle is named "Posters". The shopping basket handle determines the configuration of a shopping basket (you may have and use multiple shopping basket handles in one region). For a default shopping basket configuration, the handle does not require customization and merely needs to be created. Price, shipping, and tax rules require configuration of the shopping basket handle.



Shopping Basket Name	Operation
	Add
Posters	Edit Delete

Step 7: Configure item price rules.

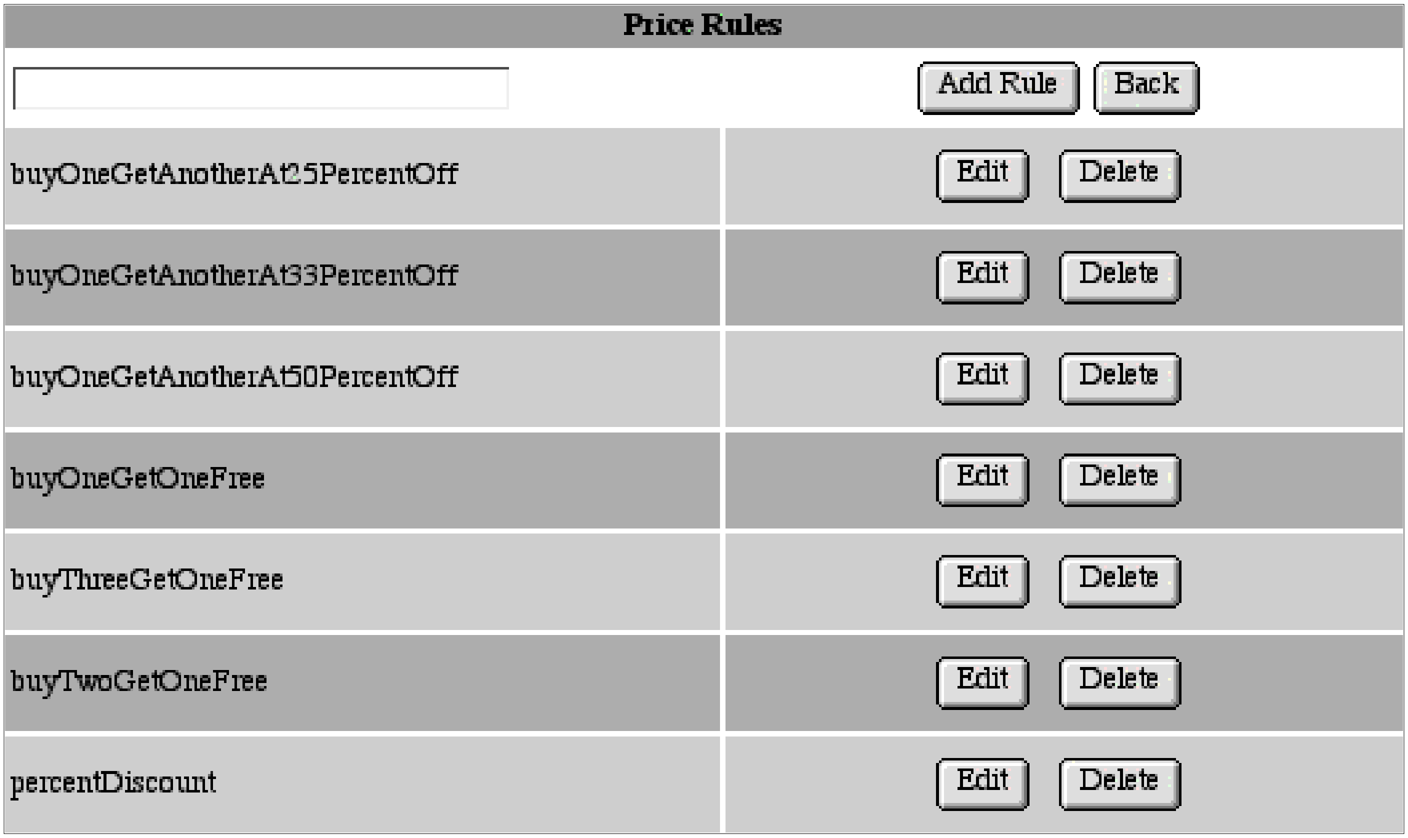
To configure the shopping basket handle (name), click the "Edit" button beside the shopping basket name to display the configuration interface:



Configuration "Posters"	Operation
	Default All Back
General Options	Edit Default
Customer Information Options	Edit Default
Price Rules	Edit Default
Shipping Rules	Edit Default
Tax Rules	Edit Default
Inventory Options	Edit Default
Order Processing Options	Edit Default

Click the "Edit" button beside "Price Rules" to display the price rules interface. Notice there are six pre-configured price rules. All pre-configured price rules are **item** rules and can be entered in the mgiBuyMe tag. In this example, one item is discounted with the pre-configured "buyOneGetOneFree" price rule.

To create a custom price rule, enter a name for the price rule and click the "Add Rule" button. For this example, create a price rule named "percentDiscount" to use for specific item discounts.

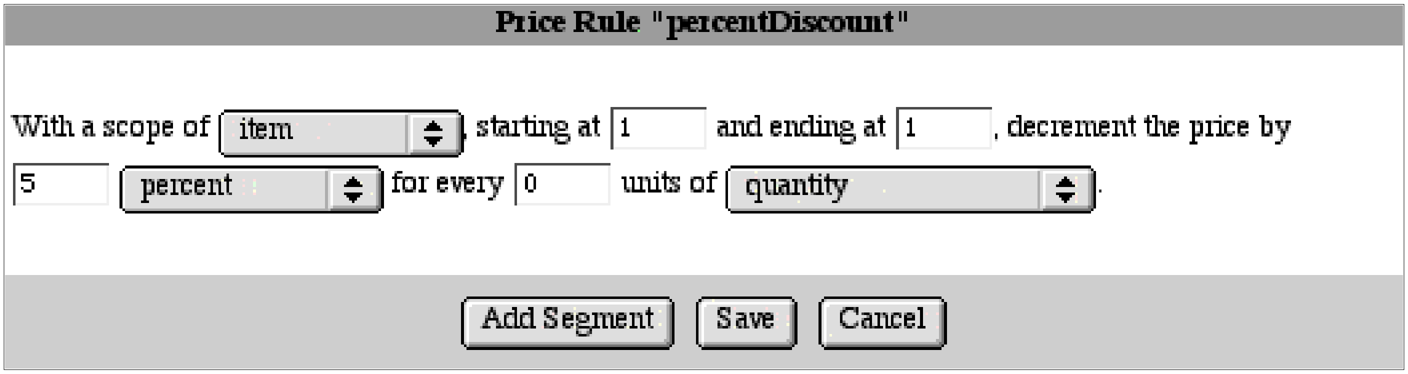


Price Rules	
buyOneGetAnotherAt25PercentOff	Edit Delete
buyOneGetAnotherAt33PercentOff	Edit Delete
buyOneGetAnotherAt50PercentOff	Edit Delete
buyOneGetOneFree	Edit Delete
buyThreeGetOneFree	Edit Delete
buyTwoGetOneFree	Edit Delete
percentDiscount	Edit Delete

To configure the price rule, click the "Edit" button beside the price rule to display the price rule configuration interface. The price rule configuration provides selections for almost any price rule algorithm. The options for a price rule segment are the scope of the price rule (item or basket), the beginning and ending break points of the price rule, the dollar or percentage discount, the units required for the discount, and the unit of measurement for the discount (item quantity, item price or item weight). Multiple segments of a price rule allow you to configure multiple break points or multiple conditions. Price rules are performed as they are entered from top to bottom.

In this example, the "percentDiscount" price rule will discount an item 5% for the first purchase and discount the item 15% percent for all additional purchases (i.e., "5% for 1 and 15% for 2 or more"). The separate discounts and associated conditions are configured in two separate segments of the "percentDiscount" price rule.

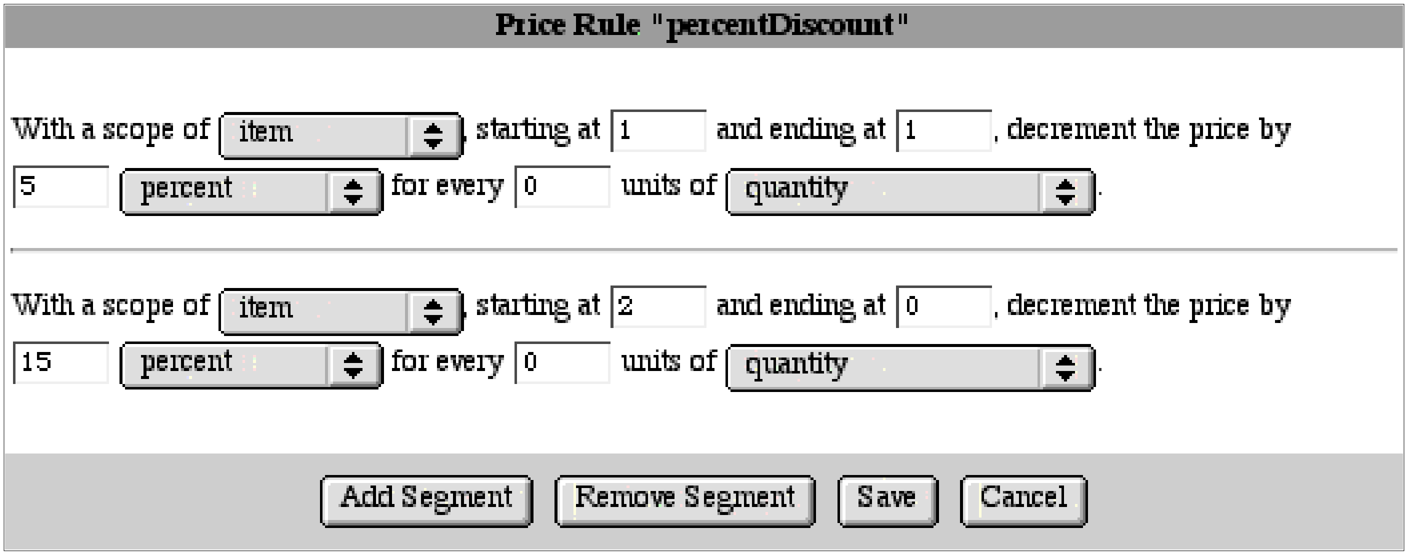
First, configure the 5% discount segment of the price rule. The scope of the discount is a specific item. The beginning and ending break points of the discount are a purchase of 1 item (i.e., beginning at 1 and ending at 1). The discount itself is 5 percent. The discount is not applied to each item from the beginning break point to the ending break point (the discount only applies once when the beginning break point is reached), therefore the discount applies to zero units of quantity.



Price Rule "percentDiscount"

With a scope of item, starting at 1 and ending at 1, decrement the price by 5 percent for every 0 units of quantity.

Next, configure the 15% discount segment of the price rule. To add a new segment to the price rule, click the "Add Segment" button. The scope of the additional discount is a specific item. The beginning break point of the discount is the second item purchased (i.e., "2") and there is not an ending break point (i.e., "0"). The discount itself is 15 percent. The discount is not applied to each item from the beginning break point to the ending break point (the discount only applies once when the beginning break point is reached), therefore the discount applies to zero units of quantity.



Price Rule "percentDiscount"

With a scope of item, starting at 1 and ending at 1, decrement the price by 5 percent for every 0 units of quantity.

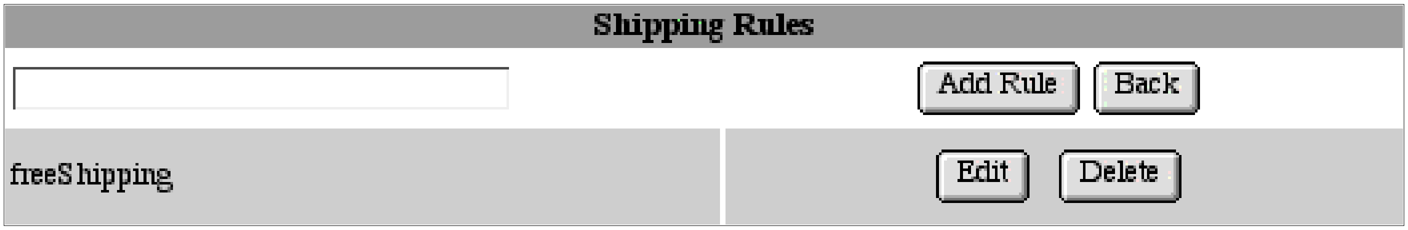
With a scope of item, starting at 2 and ending at 0, decrement the price by 15 percent for every 0 units of quantity.

Click the "Save" button to save the price rule configuration and return to the price rules interface.

Step 8: Configure item shipping rules.

Return to the main configuration interface for the "Posters" shopping basket handle by clicking the "Back" button under the "Operation" column. Click the "Edit" button beside "Shipping Rules" to display the shipping rules interface.

To create a custom shipping rule, enter a name for the shipping rule and click the "Add Rule" button. For this example, create a shipping rule named "freeShipping" to offer free shipping for multiple quantity purchases.



Shipping Rules

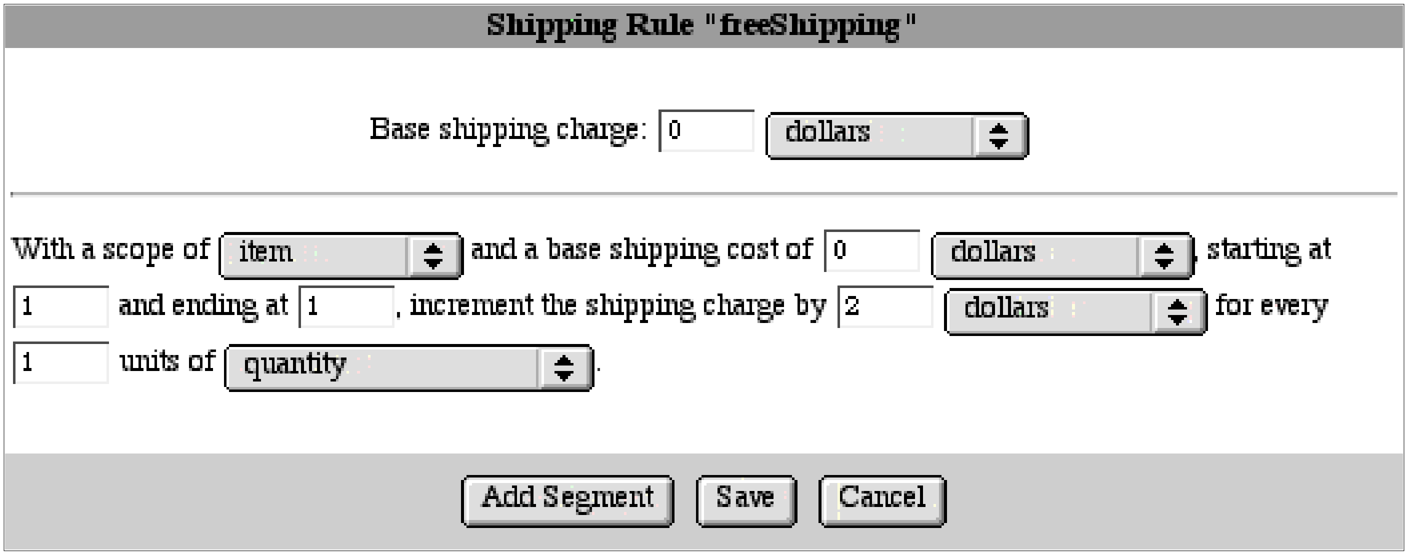
freeShipping

To configure the shipping rule, click the "Edit" button beside the shipping rule to display the shipping rule configuration interface.

The shipping rule configuration provides a selection for an overall base shipping cost (in addition to other item or basket shipping charges). The options for a shipping rule segment are the scope of the shipping rule (item or basket), the base shipping cost, the beginning and ending break points of the shipping rule, the dollar or percentage shipping charge, the units required to incur the shipping charge and the unit of measurement basis of the shipping charge. Multiple segments of a shipping rule allow you to configure multiple break points or multiple conditions. Shipping rules are performed as they are entered from top to bottom.

For this example, the shipping charge for some items is "free shipping for purchasing 2 or more of an item". To configure this shipping charge, add 2 segments, the first to configure the shipping charges for purchasing only 1 of an item and the second to configure the shipping charges for purchasing 2 or more of an item.

In the first segment, configure the shipping charges for a single purchase of an item. The scope of the shipping charge is an item. The base shipping cost is zero. The beginning and ending break points of the shipping charge are a purchase of 1 item (i.e., beginning at 1 and ending at 1). The shipping charge is 2 dollars for the first item and the shipping charge applies to each quantity unit (1) that is purchased.

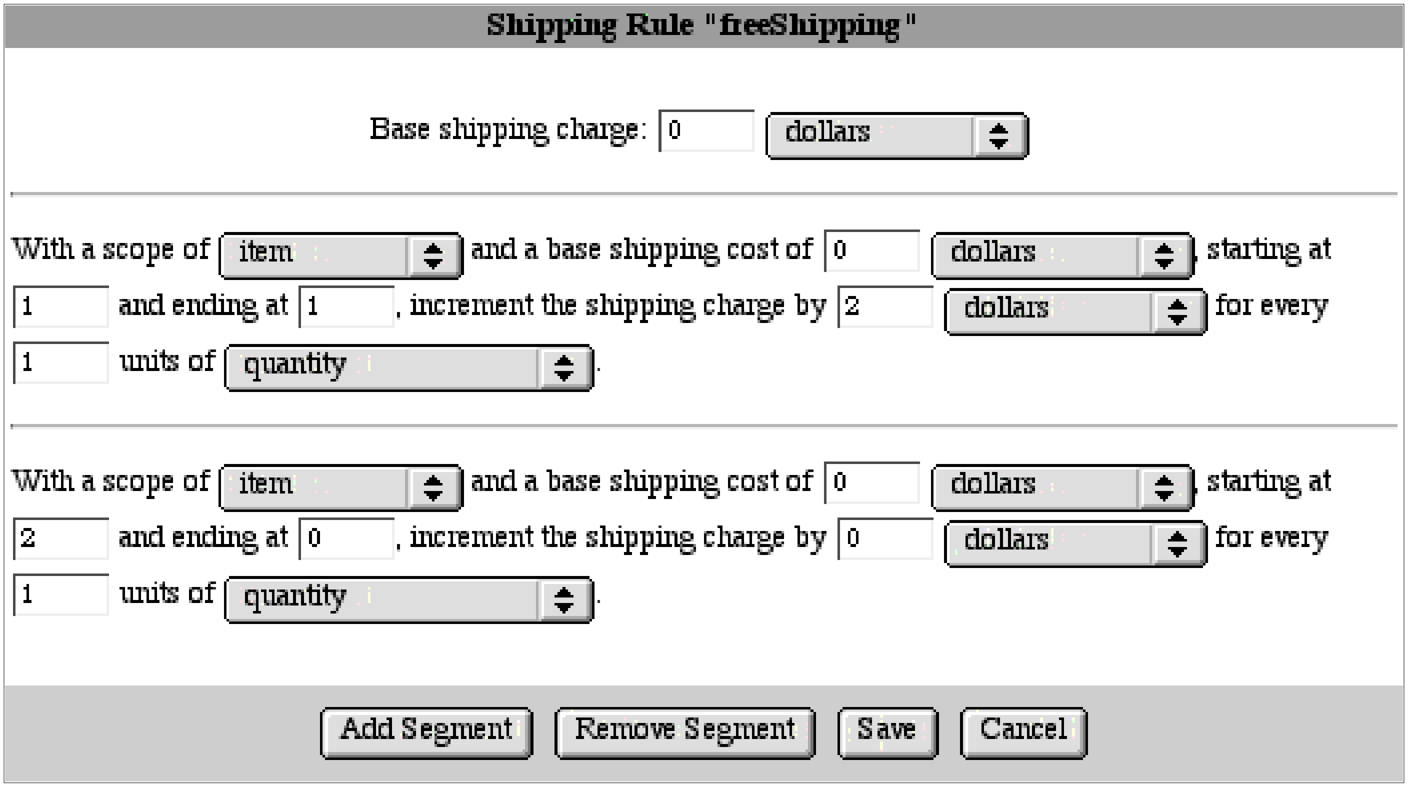


Shipping Rule "freeShipping"

Base shipping charge: 0 dollars

With a scope of item and a base shipping cost of 0 dollars, starting at 1 and ending at 1, increment the shipping charge by 2 dollars for every 1 units of quantity.

In the second segment, configure the free shipping charges for multiple quantities purchased. To add a new segment to the shipping rule, click the "Add Segment" button. The scope of the free shipping is an item. The base shipping cost is zero. The beginning break point of the free shipping is the second item purchased (i.e., "2") and there is not an ending break point (i.e., "0"). The shipping charge is zero dollars and that free shipping applies to each quantity unit (1) that is purchased.



Shipping Rule "freeShipping"

Base shipping charge: 0 dollars

With a scope of item and a base shipping cost of 0 dollars, starting at 1 and ending at 1, increment the shipping charge by 2 dollars for every 1 units of quantity.

With a scope of item and a base shipping cost of 0 dollars, starting at 2 and ending at 0, increment the shipping charge by 0 dollars for every 1 units of quantity.

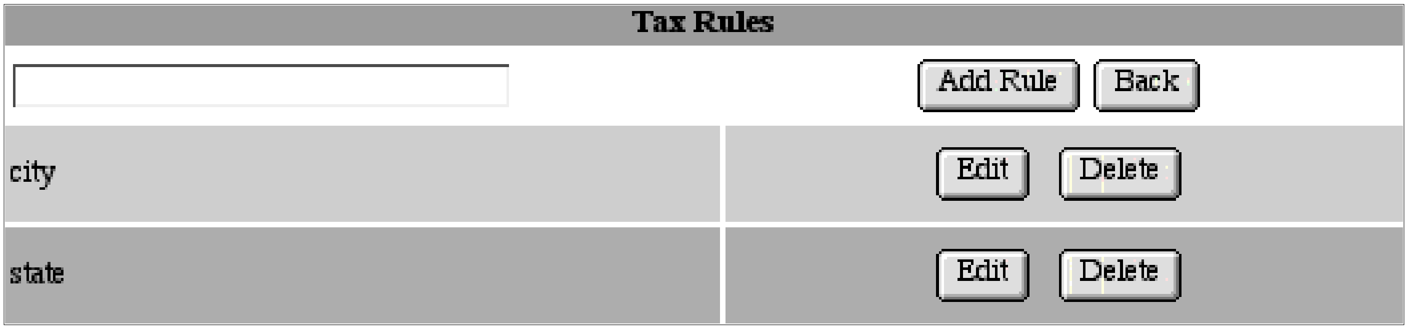
Click the "Save" button to save the shipping rule configuration and return to the shipping rules interface.

Step 9: Configure item tax rules.

Return to the main configuration interface for the "Posters" shopping basket handle by clicking the "Back" button under the "Operation" column. Click the "Edit" button beside "Tax Rules" to display the tax rules interface.

For this example, the tax charge is 1% for city sales tax and 3% for state sales tax on selected items. Customers pay no tax charges on other items. Each tax charge is entered as a separate tax rule, but the tax rules can be "chained" together to charge multiple taxes. Multiple segments of a tax rule can be added to charge taxes for multiple locations. For example, if you are charging a different state tax for North Carolina and South Carolina, you would enter two segments to the "state" tax rule.

Create two new tax rules. Name the first tax rule, "city" and the second tax rule "state". To create a custom tax rule, enter a name for the tax rule and click the "Add Rule" button.

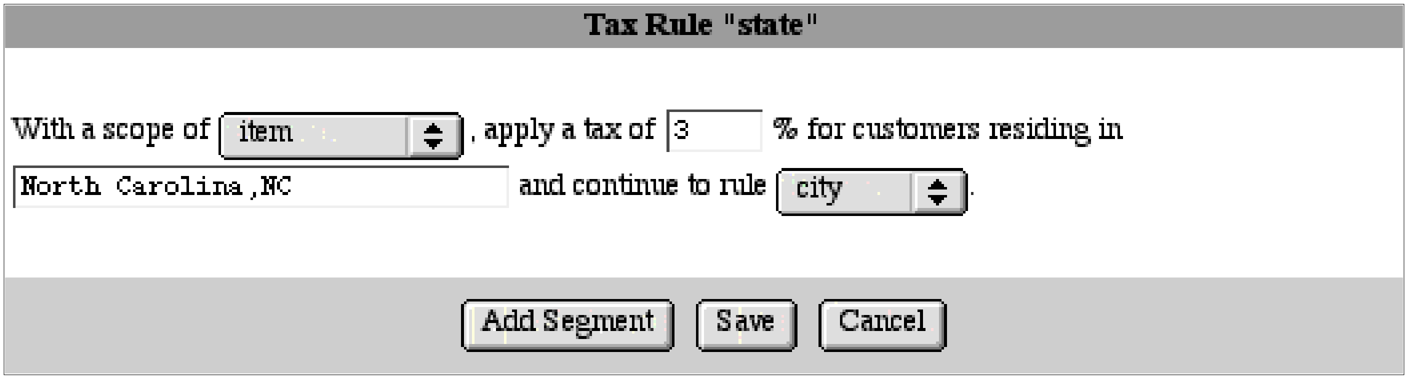


Tax Rules

city	Edit Delete
state	Edit Delete

To configure a tax rule, click the "Edit" button beside the tax rule. The options for a tax rule are the scope of the tax rule (item or basket), the percentage tax charge, the location(s) receiving the tax charge and the next tax rule to apply in the chain.

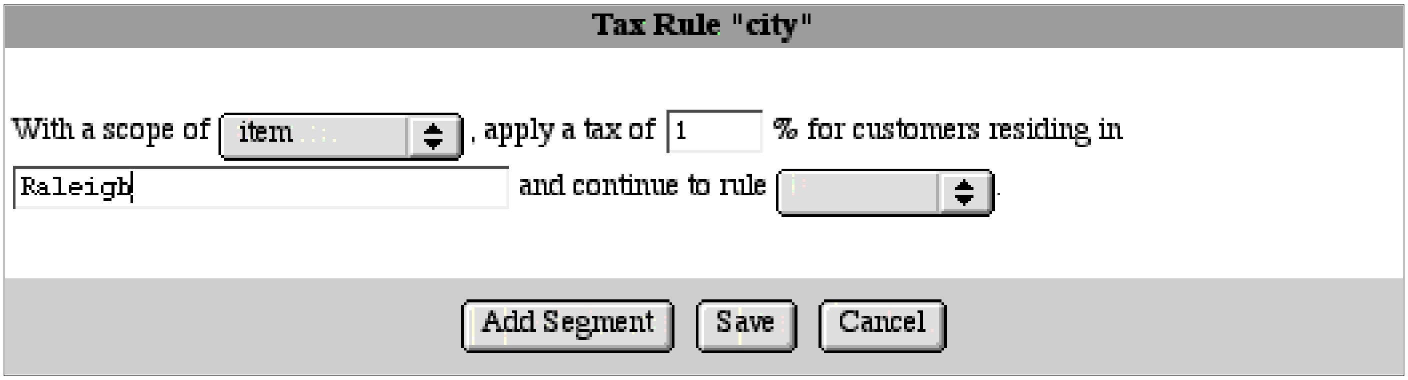
Configure the "state" tax rule as a 3% tax for residents of North Carolina (NC). The scope of the tax charge is an item. The tax charge is 3 percent. The location of the tax charge is North Carolina. Multiple locations or multiple values should be entered as a comma-delimited list (e.g., "North Carolina, NC"). After the "state" tax is applied for North Carolina, continue to the "city" tax rule. Click "Save" to save the tax rule and return to the tax rules interface.



Tax Rule "state"

With a scope of item, apply a tax of 3 % for customers residing in North Carolina ,NC and continue to rule city.

Configure the "city" tax rule as a 1% tax for residents of Raleigh. The scope of the tax charge is an item. The tax charge is 1 percent. The location of the tax charge is Raleigh. Multiple locations or multiple values should be entered as a comma-delimited list. After the "city" tax is applied for North Carolina, no additional taxes are charged. Click "Save" to save the tax rule and return to the tax rules interface.



Tax Rule "city"

With a scope of item, apply a tax of 1 % for customers residing in Raleigh and continue to rule .

Close the shopping basket admin page.

Step 10: Create product pages and open them in a text editor.

Create pages to display products for sale. You may choose any design and layout for products including pictures, descriptions, pricing information, etc. Insert placeholders for a quantity box for each product and a submit button. The default quantity box is 3 characters wide. Finally, open your product pages in a text editing program that allows you to view and modify the HTML and code of the page.

Step 11: Insert an mgiBuyMe tag, rule parameters and a submit button for each product.

Replace the quantity box placeholder for each product with an mgiBuyMe tag, productID parameter, name parameter, price parameter, and shipping parameter. For products with special pricing, shipping charges, or tax charges, enter the priceRule parameter, shippingRule parameter, and taxRule parameter, respectively.

In the productID parameter, enter a unique code for the product or embed the product's unique ID from a database search. Product information is tracked through the shopping basket by the productID. In the name parameter, enter a short description of the product or embed the product's short description from a database search. In the price parameter, enter the per item price of each product as a decimal number or embed the product's price from a database search. Do not enter characters such as dollar signs in the price parameter. In the shipping parameter, enter the per item shipping cost of each product as a decimal number or embed the product's shipping cost from a database search. Do not enter characters such as dollar signs in the shipping parameter.

If the product receives special pricing (e.g., "Buy one get one free"), include the priceRule parameter in the mgiBuyMe tag. In the priceRule parameter, enter the name of the price rule to apply to the item.

If the product incurs a shipping charge, include the shippingRule parameter in the mgiBuyMe tag. In the shippingRule parameter, enter the name of the shipping rule to apply to the item. With the specified shipping rule, the "shipping" parameter does not need to be entered.

If the product incurs a tax charge (or multiple tax charges), include the taxRule parameter in the mgiBuyMe tag. In the taxRule parameter, enter the name of the tax rule to apply. If tax rules are "chained" together, only enter the top-level tax rule. For example, if a tax rule is chained from "state" to "county" to "city", you would enter only the "state" tax rule.

Replace the submit button placeholder for each item (or for all items) with an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button (e.g., "Add to Shopping Basket").

A form action adds the quantity entered in the quantity box(s) to the shopping basket page for processing when the submit button is clicked. Enter the name of the shopping basket page (basket.mgi) in the action parameter of the <FORM> tag. Enter "post" in the method parameter of the <FORM> tag. Enclose the mgiBuyMe tag and "Add to Shopping Basket" submit button for each item with HTML form tags. If you want to allow customers to add multiple items to a shopping basket at one time, enclose all mgiBuyMe tags and a single submit button or multiple submit buttons with HTML form tags.

The default shopping basket uses tokens to track the purchases of individual customers. Enter one beginning mgiToken tag before all products and submit buttons, and enter one ending mgiToken tag after all products and submit buttons. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following are several examples of products with special item pricing, item shipping charges and item tax charges. One product may have pricing, shipping and/or a tax rule applied. You may not apply more than one of a single rule type to a product. For example, you may not apply three pricing rules to an individual product.

```
<mgiToken>

<mgiComment>Product with no price,
shipping or tax rules</mgiComment>

<FORM ACTION="basket.mgi" METHOD="post">
<P><CENTER>
<B><FONT SIZE="+1">Waterlilies</FONT></B>
</CENTER></P>
<P><CENTER><IMG SRC="images/GCL522S.JPG" BORDER="0"
WIDTH="150" HEIGHT="100" ALIGN="BOTTOM">
<BR>Item: GCL522
<BR>Artist: Monet
<BR>24&quot; x 36&quot;
<BR><MGIBUYME PRODUCTID="GCL522" NAME="Waterlilies"
PRICE="20.00" SHIPPING="1.00"> @ $20.00 ea.
<BR><MGIBUTTON VALUE="Buy Now!">
</CENTER>
</FORM>

<mgiComment>Product with the pre-configured buy
one get one free price rule</mgiComment>
```

```
<FORM ACTION="basket.mgi" METHOD="post">
<P><CENTER>
<B><FONT SIZE="+1">Bassin aux Nympheas</FONT></B>
</CENTER></P>
<P><CENTER>
<B><FONT COLOR="#ff0000" SIZE="+1">
Special: Buy 1 Get 1 FREE</FONT></B>
</CENTER></P>
<P><CENTER><IMG SRC="images/GCL593S.JPG" BORDER="0"
WIDTH="96" HEIGHT="144" ALIGN="BOTTOM">
<BR>Item: GCL593
<BR>Artist: Monet
<BR>24&quot; x 36&quot;
<BR><MGIBUYME PRODUCTID="GCL593"
NAME="Bassin aux Nympheas" PRICE="20.00"
SHIPPING="1.00" PRICERULE="buyOneGetOneFree">
@ $20.00 ea.
<BR><MGIBUTTON VALUE="Buy Now!">
</CENTER>
</FORM>

<mgiComment>Product with price rule of 5%
discount for purchasing 1 of th item and 15%
discount for purchasing 2 or more of the item
</mgiComment>

<FORM ACTION="basket.mgi" METHOD="POST">
<P><CENTER>
<B><FONT SIZE="+1">The Starry Night</FONT></B>
</CENTER></P>
<P><CENTER>
<B><FONT COLOR="#ff0000" SIZE="+1">
Special: 5% Discount on 1 &amp;
<BR>15% discount on 2 or more</FONT></B>
</CENTER></P>
<P><CENTER><IMG SRC="images/GCM550S.JPG" BORDER="0"
WIDTH="124" HEIGHT="100" ALIGN="BOTTOM">
<BR>Item: GCM550
<BR>Artist: Van Gogh
<BR>16 x 20
<BR><MGIBUYME PRODUCTID="GCM550"
NAME="The Starry Night" PRICE="14.00"
SHIPPING="1.00" PRICERULE="percentDiscount">
@ $14.00 ea.
<BR><MGIBUTTON VALUE="Buy Now!">
```

```
</CENTER>
</FORM>

<mgiComment>Product with free shipping rule
for purchasing 2 or more of the item</mgiComment>

<FORM ACTION="basket.mgi" METHOD="post">
<P><CENTER>
<B><FONT SIZE="+1">Sailboats at Argenteuil</FONT></B>
</CENTER></P>
<P><CENTER>
<B><FONT COLOR="#ff0000" SIZE="+1">
Special: Purchase 2 or more and receive FREE
shipping on this item</FONT></B>
</CENTER></P>
<P><CENTER><IMG SRC="images/ARV759S.JPG" BORDER="0"
WIDTH="215" HEIGHT="70" ALIGN="BOTTOM">
<BR>Item: ARV759
<BR>Artist: Monet
<BR>11-3/4&quot; x 36&quot;
<BR><MGIBUYME PRODUCTID="ARV759"
NAME="Sailboats at Argenteuil" PRICE="12.00"
SHIPPINGRULE="freeShipping">
@ $12.00 ea.
<BR><MGIBUTTON VALUE="Buy Now!">
</CENTER>
</FORM>

<mgiComment>Product with chained tax rule
for state taxes and city taxes</mgiComment>

<FORM ACTION="basket.mgi" METHOD="POST">
<P><CENTER>
<B><FONT SIZE="+1">Composition No. 6</FONT></B>
</CENTER></P>
<P><CENTER><IMG SRC="images/GCL641S.JPG" BORDER="0"
WIDTH="150" HEIGHT="100" ALIGN="BOTTOM">
<BR>Item: GCL641
<BR>Artist: Kandinsky
<BR>24&quot; x 36&quot;
<BR><MGIBUYME PRODUCTID="GCL641"
NAME="Composition No. 6" PRICE="16.00"
SHIPPING="1.00" TAXRULE="state">
@ $16.00 ea. + <B><FONT COLOR="#ff0000">Tax</FONT></B>
<BR><MGIBUTTON VALUE="Buy Now!">
```

```
</CENTER>
</FORM>

</mgiToken>
```

Step 12: Save the product pages.

Save the changes you have made to the product pages.

Step 13: Create a shopping basket page and open it in a text editor.

Create a page named "basket.mgi" to display the contents of a customer's shopping basket and open the page in a text editing program that allows you to view and modify the HTML and code.

Step 14: Insert the mgiShoppingBasket and mgiButton tags.

Insert your cursor in the HTML of the shopping basket page where you want the shopping basket to display and enter the beginning mgiShoppingBasket tag, handle parameter and ending mgiShoppingBasket tag. In the handle parameter, enter the name of the shopping basket handle you created in the shopping basket administration (i.e., "Posters").

Below the mgiShopping basket tag, enter an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to modify shopping basket quantities (e.g., "Modify Quantity").

Below the mgiShoppingBasket tag and modify quantity button, enter a second mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to proceed to the check out page (e.g., "Check Out").

On the shopping basket page, form actions perform two functions. One form action modifies the contents of the shopping basket (i.e., changes quantities). Enclose the mgiShoppingBasket tag and mgiButton tag used to modify the shopping basket with HTML form tags. Enter the name of the shopping basket page (basket.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag.

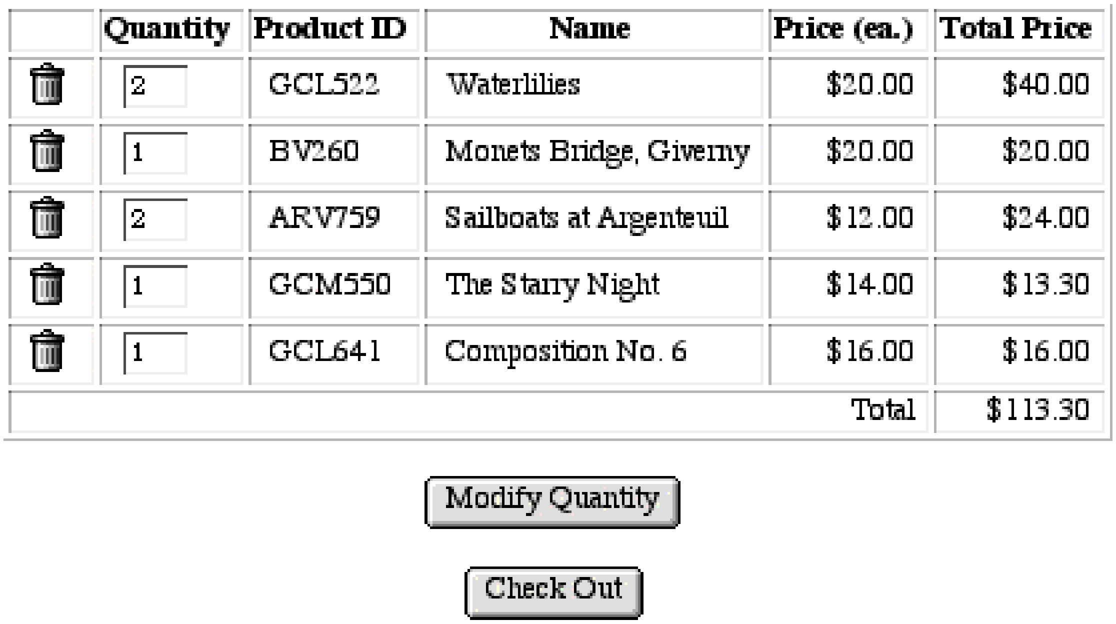
Another form action links customers to the check out page to enter payment, billing and shipping information. Enclose the mgiButton tag used to check out with HTML form tags. Enter the name of the check out page (checkout.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag. Note: To use a secure server to collect payment information, enter the URL to a secure server running MGI in the action parameter of the <FORM> tag (e.g., https://secure.domain.com/checkout.mgi). You may substitute a text link for the check out button and form tags.

The default shopping basket uses tokens to track the purchases of individual customers. Enter a beginning mgiToken tag before the first HTML form tag and enter an ending mgiToken tag after the last closing HTML form tag. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example shopping basket page.

```
<mgiToken>

<center>

<h2>Shopping Basket</h2>

<form action="basket.mgi" method="post">

<p><mgiShoppingBasket handle="Posters">
</mgiShoppingBasket>

<p><mgiButton value="Modify Quantity">

</form>

<form action="checkout.mgi" method="post">

<mgiButton value="Check Out">

</form>

</center>

</mgiToken>
```

The default shopping basket contents display in a table with columns for product removal, product quantity, product ID, product name, product price, multiplied product subtotals and order total. Price discounts are deducted and displayed in the "Total Price" column.



	Quantity	Product ID	Name	Price (ea.)	Total Price
🗑	2	GCL522	Waterlilies	$20.00	$40.00
🗑	1	BV260	Monets Bridge, Giverny	$20.00	$20.00
🗑	2	ARV759	Sailboats at Argenteuil	$12.00	$24.00
🗑	1	GCM550	The Starry Night	$14.00	$13.30
🗑	1	GCL641	Composition No. 6	$16.00	$16.00
				Total	$113.30

Step 15: Save the shopping basket page.

Save the changes you have made to the shopping basket page (basket.mgi).

Step 16: Create a check out page and open it in a text editor.

Create a page named "checkout.mgi" to collect payment, billing and shipping information from the customer and open the page in a text editing program that allows you to modify the HTML and code of the page.

Step 17: Insert the mgiCollectUserInfo and mgiButton tags.

Insert your cursor in the HTML of the check out page where you want the payment, billing and shipping information tables to display and enter the mgiCollectUserInfo tag, handle parameter and shoppingBasketURL parameter. In the handle parameter, enter the name of the shopping basket handle you created in the shopping basket administration (i.e., "Posters"). In the shoppingBasketURL parameter, enter the full URL to the region where items were added to the shopping basket (and thus where the internal MGI shopping basket database was created and populated).

Below the mgiCollectUserInfo tag, enter an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to proceed to the order confirmation page (e.g., "Confirm Order").

Form actions on the check out page link customers to the order confirmation page to review their order and customer information. Enclose the mgiCollectUserInfo tag, mgiButton tag, and any custom form elements with HTML form tags. Enter the name of the order confirmation page (confirm.mgi) in the action parameter of the <FORM>

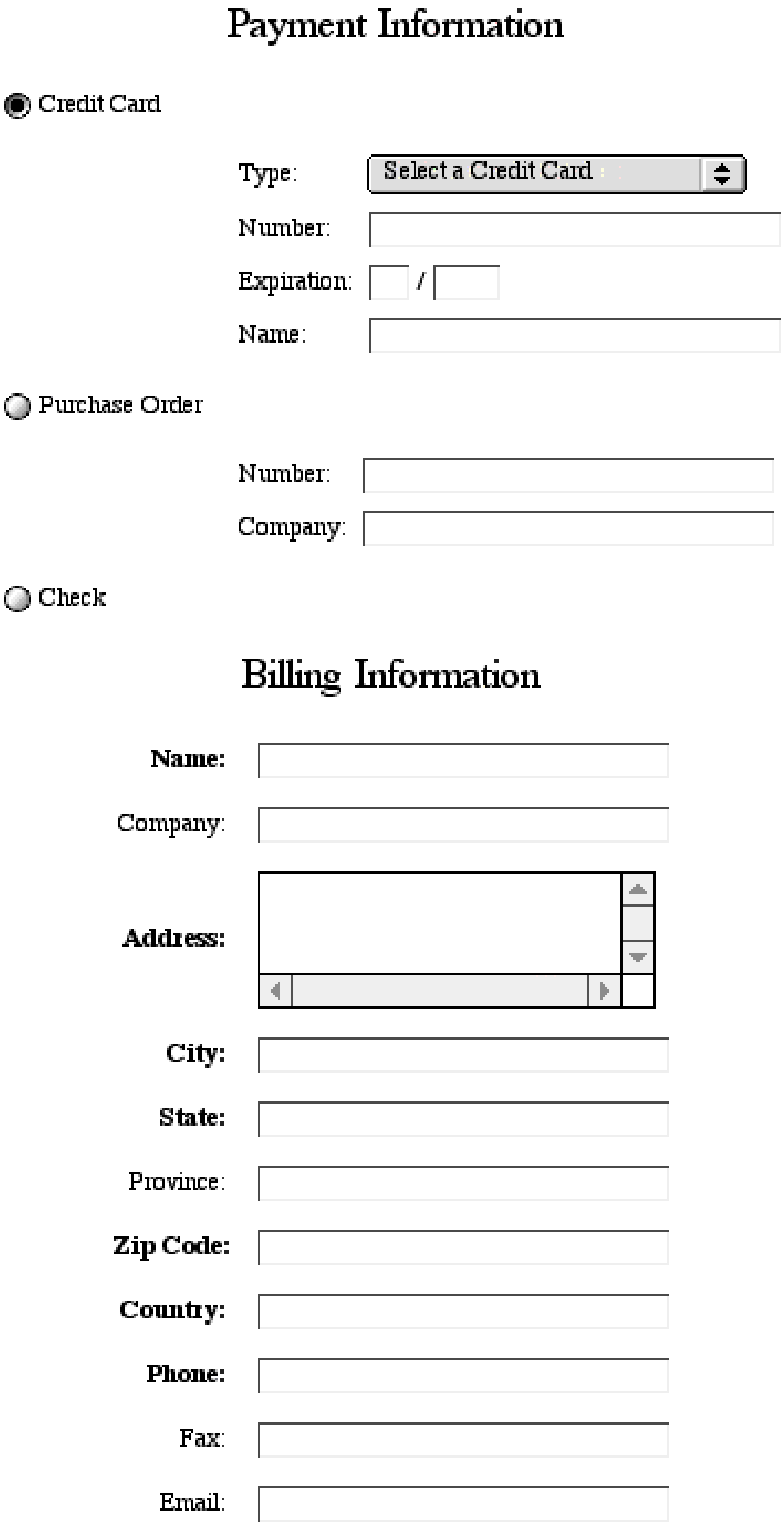
tag and enter "post" in the method parameter of the <FORM> tag.

The default shopping basket uses tokens to track the purchases of individual customers. Enter a beginning mgiToken tag before the beginning HTML form tag and enter an ending mgiToken tag after the ending HTML form tag. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example check out page.

```
<mgiToken>

<form action="confirm.mgi" method="post">

<center>

<mgiCollectUserInfo handle="Posters"
shoppingBasketURL="http://www.domain.com/shop/">

<p><mgiButton value="Confirm Order">

</center>

</form>

</mgiToken>
```

The default check out page display payment, billing and shipping information in tables. Custom form elements appear as you format them.



Payment Information

● Credit Card

Type: Select a Credit Card

Number:

Expiration: /

Name:

○ Purchase Order

Number:

Company:

○ Check

Billing Information

Name:

Company:

Address:

City:

State:

Province:

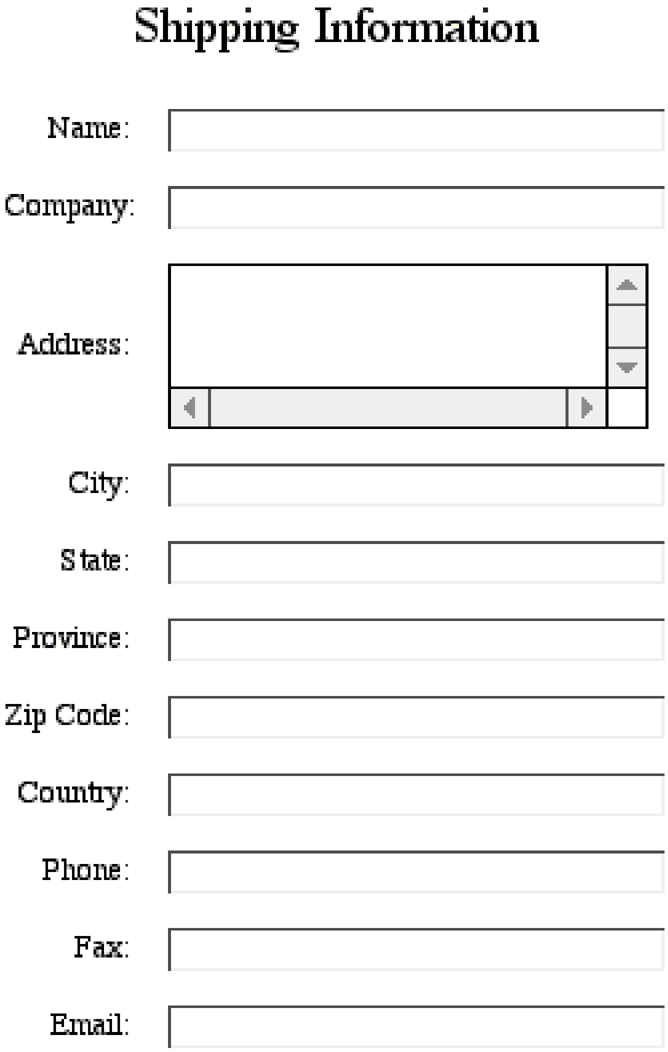
Zip Code:

Country:

Phone:

Fax:

Email:



Step 18: Save the check out page.

Save the changes you have made to the check out page.

Step 19: Create a confirm order page and open it in a text editor.

Create a page named "confirm.mgi" to present the final shopping basket, payment information, billing information, and shipping information for review. Open the confirm order page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 20: Insert the mgiConfirmOrder tag, customerLocation parameter and mgiButton tag.

Insert your cursor in the HTML of the confirm order page where you want the shopping basket, payment, billing, shipping, and additional information tables to display and enter the beginning mgiConfirmOrder tag, handle parameter, shoppingBasketURL parameter, customerLocation parameter and ending mgiConfirmOrder tag.

In the handle parameter, enter the name of the shopping basket handle you created in the shopping basket administration (i.e., "Posters"). In the shoppingBasketURL parameter, enter the full URL to the region where items were added to the shopping basket (and thus where the internal MGI shopping basket database was created and populated).

When implementing a tax rule, you must collect the customer's location(s) from their billing information and embed it in the customerLocation parameter of the mgiConfirmOrder parameter. For "chained" tax rules, embed a comma-delimited list of the customer's location from the top-level tax rule to the bottom-level tax rule. The location list must be in the same order as the tax rule chain. Create the comma-delimited list in a variable (using mgiSet) before the mgiConfirmOrder tag and embed the variable in the customerLocation parameter of the mgiConfirmOrder tag (using mgiGet). In this example, the customer's state and city are compiled in a variable and embedded in the customerLocation parameter.

Below the mgiConfirmOrder tag, enter an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to proceed to the final order processing page (e.g., "Complete Order").

Form actions on the confirm order page link customers to the order processing page to complete and send their order. Enclose the mgiConfirmOrder tag and mgiButton tag with HTML form tags. Enter the name of the order processing page (process.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag.

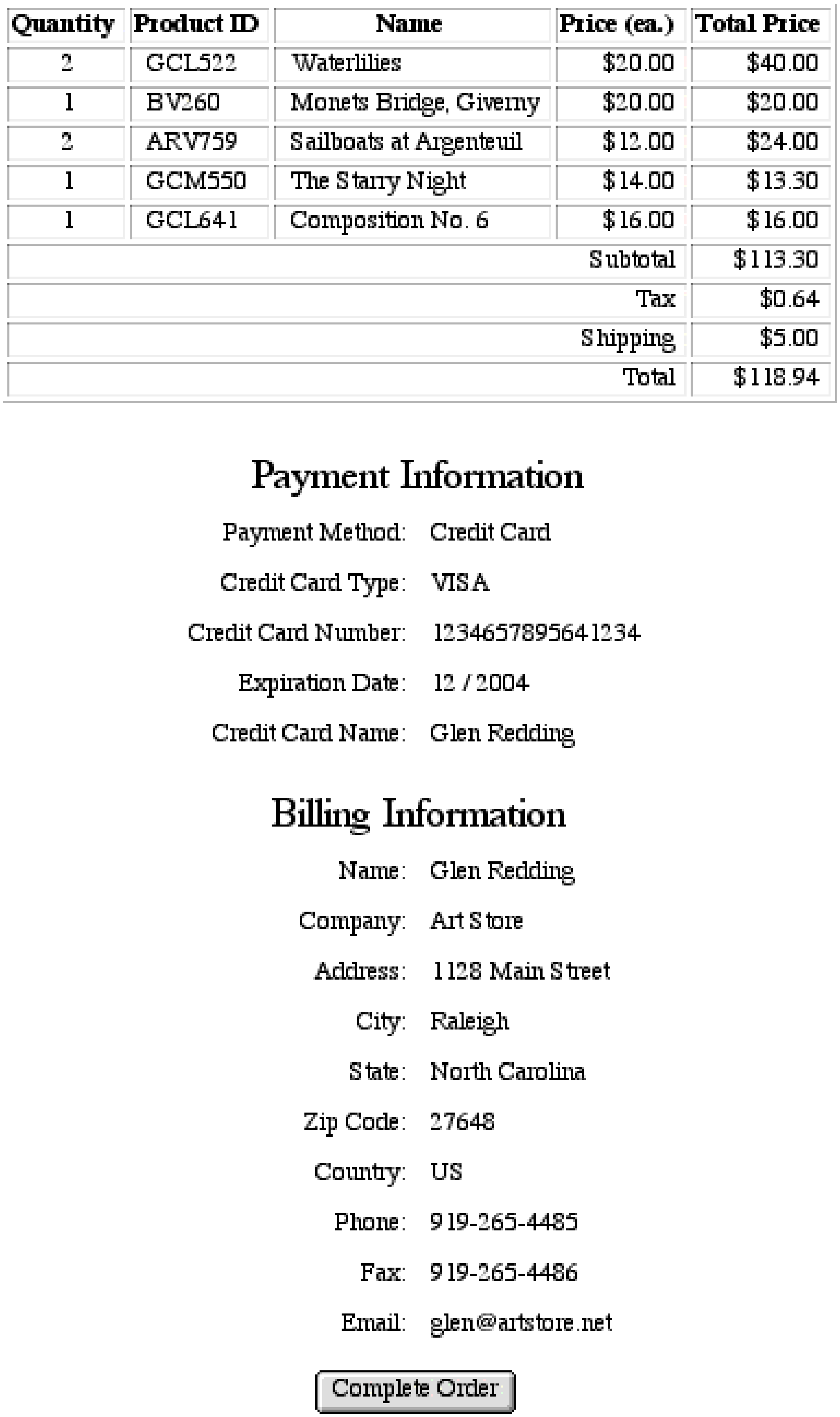
The default shopping basket uses tokens to track the purchases of individual customers. Enter a beginning mgiToken tag before the beginning HTML form tag and enter an ending mgiToken tag after the ending HTML form tag. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example confirm order page.

```
<mgiToken>

<form action="process.mgi" method="post">

<center>

<mgiSet name="locations">
<mgiPostArgument name="bSTATE">,<mgiPostArgument
name="bCITY">
```

```
</mgiSet>

<mgiConfirmOrder handle="Posters"
shoppingBasketURL="http://www.domain.com/shop/"
customerLocation={mgiGet name="locations"}>
</mgiConfirmOrder>

<p><mgiButton value="Complete Order">

</center>

</form>

</mgiToken>
```

The default confirm order page displays completed elements of the payment, billing and shipping information. The confirm order page also displays the calculated shipping and total item tax charges.



Quantity	Product ID	Name	Price (ea.)	Total Price
2	GCL522	Waterlilies	$20.00	$40.00
1	BV260	Monets Bridge, Giverny	$20.00	$20.00
2	ARV759	Sailboats at Argenteuil	$12.00	$24.00
1	GCM550	The Starry Night	$14.00	$13.30
1	GCL641	Composition No. 6	$16.00	$16.00
			Subtotal	$113.30
			Tax	$0.64
			Shipping	$5.00
			Total	$118.94

Payment Information

Payment Method:	Credit Card
Credit Card Type:	VISA
Credit Card Number:	1234657895641234
Expiration Date:	12 / 2004
Credit Card Name:	Glen Redding

Billing Information

Name:	Glen Redding
Company:	Art Store
Address:	1128 Main Street
City:	Raleigh
State:	North Carolina
Zip Code:	27648
Country:	US
Phone:	919-265-4485
Fax:	919-265-4486
Email:	glen@artstore.net

Step 21: Save the confirm order page.

Save the changes you have made to the confirm order page.

Step 22: Create an order processing page and open it in a text editor.

Create an page named "process.mgi" to present a "thank you for purchasing" message to customers and to send a formatted email of the order to a specified address. Open the order processing page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 23: Insert the mgiSendOrder tag and customerLocation parameter.

Enter a "thank you" message to display to customers when they complete their order.

At any place on the order processing page, enter a beginning mgiSendOrder tag, handle parameter, shoppingBasketURL parameter, customerLocation parameter, to parameter, from parameter, mailServer parameter, subject parameter, and ending mgiSendOrder tag.

In the handle parameter, enter the name of the default handle you created in the shopping basket administration (i.e., "Posters"). In the shoppingBasketURL parameter, enter the full URL to the region where items were added to the shopping basket (and thus where the internal MGI shopping basket database was created and populated).

When implementing a tax rule, you must collect the customer's location(s) from their billing information and embed it in the customerLocation parameter of the mgiSendOrder parameter. For "chained" tax rules, embed a comma-delimited list of the customer's location from the top-level tax rule to the bottom-level tax rule. The location list must be in the same order as the tax rule chain. Create the comma-delimited list in a variable (using mgiSet) before the mgiSendOrder tag and embed the variable in the customerLocation parameter of the mgiSendOrder tag (using mgiGet). In this example, the customer's state and city are compiled in a variable and embedded in the customerLocation parameter.

In the "to" parameter, enter the email address to receive shopping basket orders. In the "from" parameter, enter the email address that appears in the "from" line of shopping basket orders. In the mailServer parameter, enter the address for the outgoing SMTP mail server of your domain (e.g., mail.domain.com). In the subject parameter, enter the subject of the order email. The mgiSendOrder tag does not display information to the customer.

The following is an example order processing page.

```
<center>

<h2>Order Complete</h2>
```

```
<p>Thank you for ordering.  Your order has been
processed and you should receive the merchandise
within 2 to 4 weeks.

<mgiSet name="locations">
<mgiPostArgument name="bSTATE">,<mgiPostArgument
name="bCITY">
</mgiSet>

<mgiSendOrder handle="Posters"
shoppingBasketURL="http://www.domain.com/shop/"
to="sales@domain.com" from="webmaster@domain.com"
mailServer="mail.domain.com"
subject="Online Order"
customerLocation={mgiGet name="locations"}>
</mgiSendOrder>

</center>
```

The default order email is formatted with payment, billing, shipping, and order information.

```
Payment Information
-------------------
 Payment Method:  Credit Card
           Type:  VISA
         Number:  1234657895641234
Expiration Date:  12/2004
           Name:  Glen Redding

Billing Information
-------------------
           Name:  Glen Redding
        Company:  Art Store
        Address:  1128 Main Street
           City:  Raleigh
          State:  North Carolina
       Zip Code:  27648
        Country:  US
          Phone:  919-265-4485
            Fax:  919-265-4486
          Email:  glen@artstore.net

Product Information
--------------------
```

```
       Quantity:  2
     Product ID:  GCL522
           Name:  Waterlilies
    Total Price:  $40.00

       Quantity:  1
     Product ID:  BV260
           Name:  Monets Bridge, Giverny
    Total Price:  $20.00

       Quantity:  2
     Product ID:  ARV759
           Name:  Sailboats at Argenteuil
    Total Price:  $24.00

       Quantity:  1
     Product ID:  GCM550
           Name:  The Starry Night
    Total Price:  $13.30

       Quantity:  1
     Product ID:  GCL641
           Name:  Composition No. 6
    Total Price:  $16.00

       Subtotal:  $113.30
            Tax:  $0.64
       Shipping:  $5.00
          Total:  $118.94
```

Step 24: Save the order processing page.

Save the changes you have made to the order processing page.

Step 25: Tokenize all pages of the web site.

The default shopping basket uses tokens to track the purchases of individual customers. Using a text editor, enter a beginning mgiToken tag and ending mgiToken tag on all additional pages of the web site. The mgiToken tags should enclose all links on the pages including HREFs and FORM actions.

```
<mgiToken>
All tags and links appear here.
</mgiToken>
```

Step 26: Save the web site pages.

Save the changes that you make as you tokenize each page.

Step 27: FTP all pages to the web server running MGI.

Upload all pages of the web site to the web server using an FTP program. If you are using a secure server, upload **only** the check out, order confirmation and order processing pages to the secure portion of your web site. Do not attempt to use the shopping basket page on a secure server.

Step 28: View a product page in a web browser and purchase a product.

View a product page in a web browser. Enter a quantity to purchase in the quantity text field and click the "Add to Shopping Basket" button. The item(s) appears in the shopping basket display on the shopping basket page. Click the "Check Out" button to proceed to the check out page. On the check out page, enter a payment choice, billing information and shipping information (if it is different). Click the "Confirm Order" button to proceed to the confirm order page. On the confirm order page, review your order, payment information, billing information and shipping information, then click the "Complete Order" button to complete the ordering process. The message from the order processing page is displayed and the shopping basket order is formatted and emailed.

[Return to the Shopping Online Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Price, Shipping and Tax Basket Rules

Introduction

Rules are algorithms that MGI uses to calculate product price specials, shipping and tax. Rules can be applied on a per item basis (item rules) or across all items in the shopping basket (basket rules). Item rules apply to individual items and are entered in the mgiBuyMe tag for the specific item. Item tax rules also require parameters in the mgiConfirmOrder and mgiSendOrder tags. Basket rules apply across all items in a shopping basket and are entered in the mgiShoppingBasket, mgiConfirmOrder, and mgiSendOrder tags.

Item rules and basket rules are mutually exclusive for each type of rule. For example, you may not have an item price rule and a basket price rule, however you may have an item price rule and a basket shipping rule. All rules are configured in the web-based administration interface of the mgiShoppingBasket tag.

Price rules for an entire order include specials such as "buy $50 and get $5 off" and percentage discounts. A base shipping cost for an entire order can be entered in addition to shipping charges such as "$3 for 1 to 5 items, $5 for 6 to 10 items, etc.". Tax rules help you calculate local, state and federal taxes or any other tax on an order based on a customer's location. Tax rules can also be chained in a specific order for multiple taxes.

The following example is a shopping site for art posters that illustrates price, shipping and tax rules for entire orders (i.e. "basket rules"). This example is hard-coded products. The same principles would apply to mgiBuyMe tags with embedded product information from a database.

MGI Tags

- mgiButton
- mgiBuyMe
- mgiCollectUserInfo
- mgiConfirmOrder
- mgiSendOrder
- mgiShoppingBasket

Steps

1. Create a shopping basket administration page in a text editor.
2. Insert the mgiShoppingBasket tag in admin mode.
3. Save the shopping basket administration page.
4. FTP the shopping basket administration page to the web server running MGI.
5. View the shopping basket administration page in a web browser.
6. Create a shopping basket handle.
7. Configure basket price rules.
8. Configure basket shipping rules.
9. Configure basket tax rules.
10. Create product pages and open them in a text editor.
11. Insert an mgiBuyMe tag and a submit button for each product.
12. Save the product pages.

13. Create a shopping basket page and open it in a text editor.
14. Insert the mgiShoppingBasket tag, rule parameters and mgiButton tag.
15. Save the shopping basket page.
16. Create a check out page and open it in a text editor.
17. Insert the mgiCollectUserInfo tag and mgiButton tags.
18. Save the check out page.
19. Create a confirm order page and open it in a text editor.
20. Insert the mgiConfirmOrder tag, rule parameters, customerLocation parameter and mgiButton tag.
21. Save the confirm order page.
22. Create an order processing page and open it in a text editor.
23. Insert the mgiSendOrder tag, rule parameters and customerLocation parameter.
24. Save the order processing page.
25. Tokenize all other pages of the web site.
26. Save the web site pages.
27. FTP all pages to the web server running MGI.
28. View a product page in a web browser and purchase a product.

Step 1: Create a shopping basket administration page in a text editor.

Create a new page in a text editing program to display the web-based, shopping basket administration interface.

Step 2: Insert the mgiShoppingBasket tag in admin mode.

On the shopping basket administration page, enter the mgiShoppingBasket tag and mode parameter. In the mode parameter, enter "admin".

```
<mgiShoppingBasket mode="admin">
```

Step 3: Save the shopping basket administration page.

Save the shopping basket administration page and name it "sbadmin.mgi".

Step 4: FTP the shopping basket administration page to the web server running MGI.

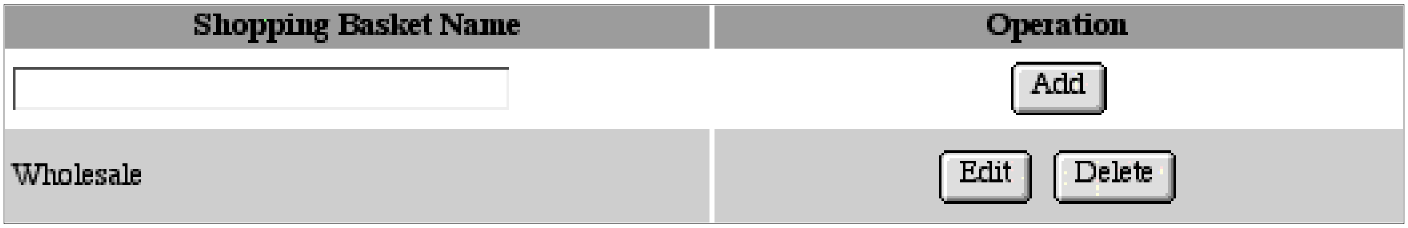
Upload the shopping basket administration page (sbadmin.mgi) from your local computer to the web server using an FTP program.

Step 5: View the shopping basket administration page in a web browser.

View the shopping basket administration page (sbadmin.mgi) in a web browser. The first screen of the web-based, administration interface is displayed.

Step 6: Create a shopping basket handle.

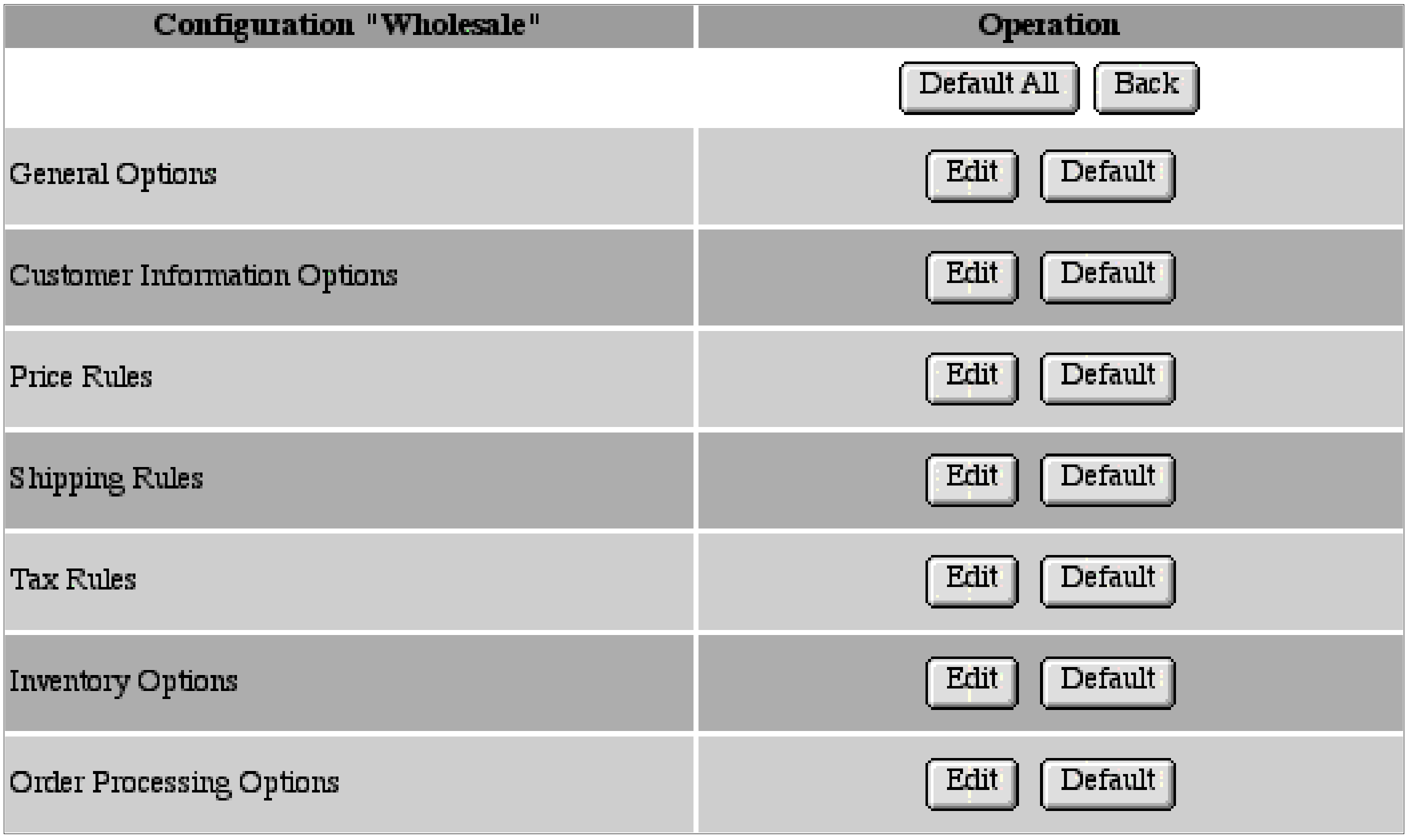
Below the column "Shopping Basket Name", enter the name of a shopping basket handle and click the "Add" button. In this example, the shopping basket handle is named "Wholesale". The shopping basket handle determines the configuration of a shopping basket (you may have and use multiple shopping basket handles in one region). For a default shopping basket configuration, the handle does not require customization and merely needs to be created. Price, shipping, and tax rules require configuration of the shopping basket handle.



Shopping Basket Name	Operation
	Add
Wholesale	Edit Delete

Step 7: Configure basket price rules.

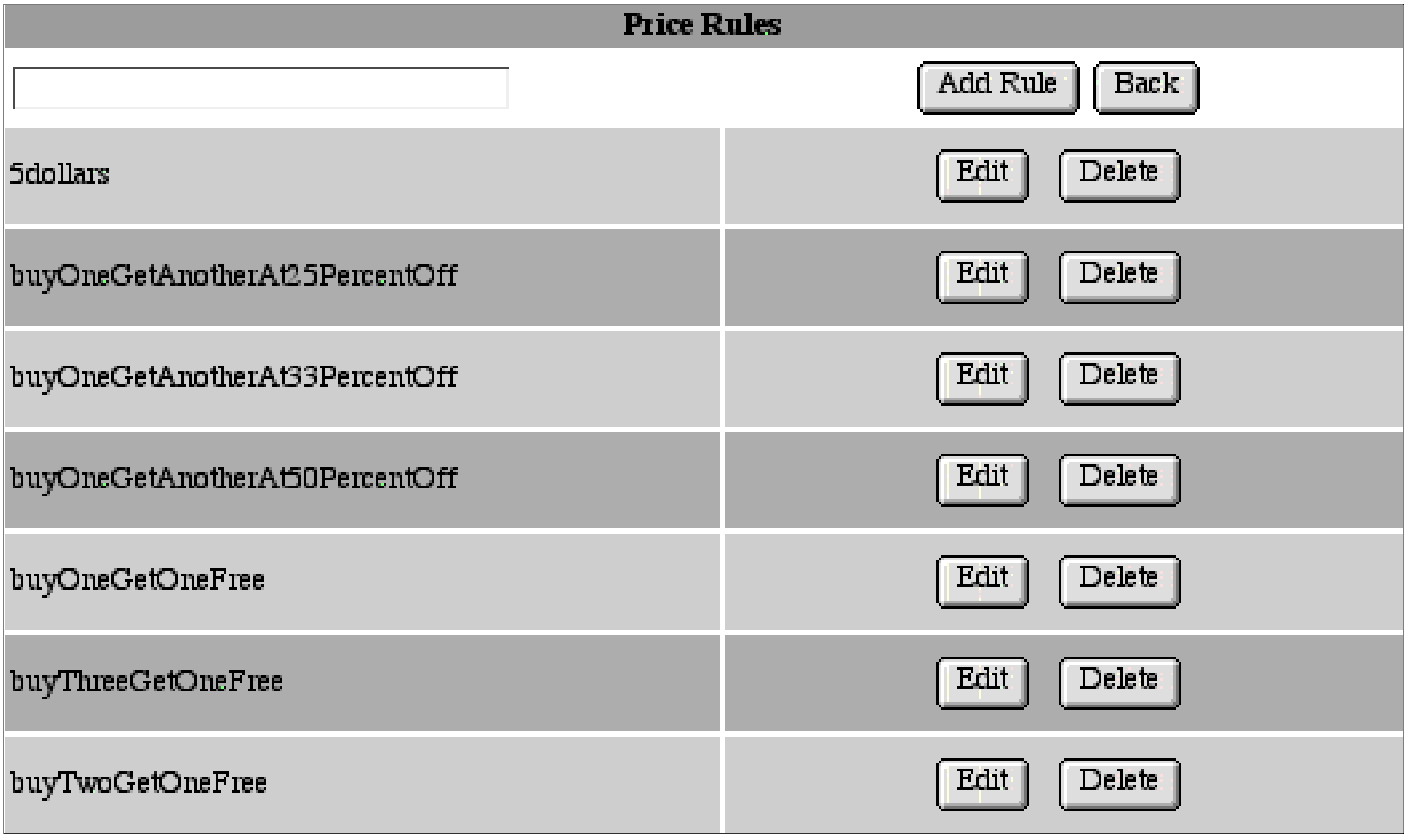
To configure the shopping basket handle (name), click the "Edit" button beside the shopping basket name to display the configuration interface:



Configuration "Wholesale"	Operation
	Default All Back
General Options	Edit Default
Customer Information Options	Edit Default
Price Rules	Edit Default
Shipping Rules	Edit Default
Tax Rules	Edit Default
Inventory Options	Edit Default
Order Processing Options	Edit Default

Click the "Edit" button beside "Price Rules" to display the price rules interface. Notice there are six pre-configured price rules. All pre-configured price rules are **item** rules and therefore cannot be used for an order discount (basket rules).

To create a custom price rule, enter a name for the price rule and click the "Add Rule" button. For this example, create a price rule named "5dollars" to use for an order discount.

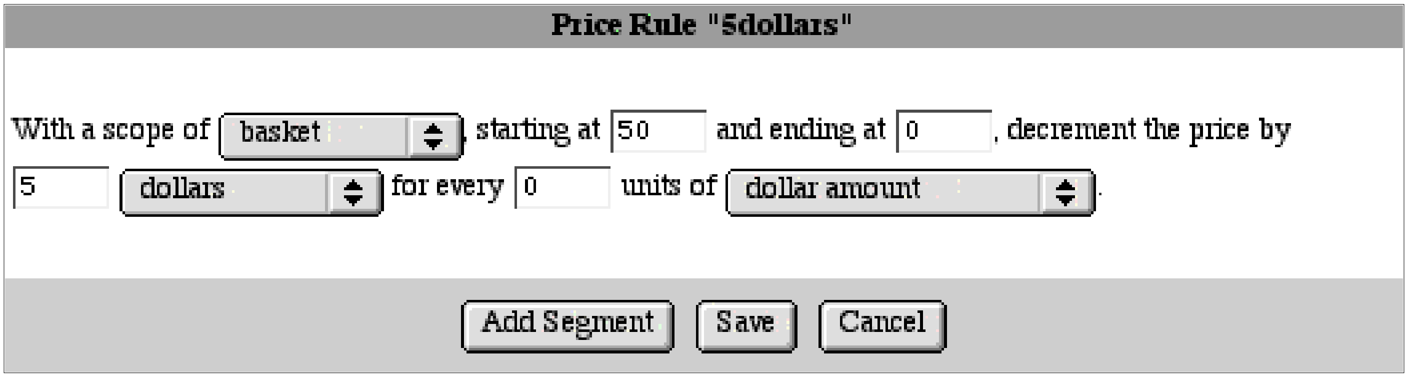


Price Rules	
	Add Rule Back
5dollars	Edit Delete
buyOneGetAnotherAt25PercentOff	Edit Delete
buyOneGetAnotherAt33PercentOff	Edit Delete
buyOneGetAnotherAt50PercentOff	Edit Delete
buyOneGetOneFree	Edit Delete
buyThreeGetOneFree	Edit Delete
buyTwoGetOneFree	Edit Delete

To configure the price rule, click the "Edit" button beside the price rule to display the price rule configuration interface. The price rule configuration provides selections for almost any price rule algorithm. The options for a price rule segment are the scope of the price rule (item or basket), the beginning and ending break points of the price rule, the dollar or percentage discount, the units required for the discount, and the unit of measurement for the discount (item quantity, item price or item weight). Multiple segments of a price rule allow you to configure multiple break points or multiple conditions. Price rules are performed as they are entered from top to bottom.

In this example, the "5dollars" price rule will discount a basket $5 when the customer purchases at least $50 of merchandise.

To configure the price rule. The scope of the discount is a basket. The beginning break point is 50 dollars (when the discount is applied) and there is not an ending break point (i.e., "0"). The discount itself is 5 dollars. The discount is not applied to each item from the beginning break point to the ending break point (the discount only applies once when the beginning break point is reached), therefore the discount applies to zero units of dollar amount.


Price Rule "5dollars"

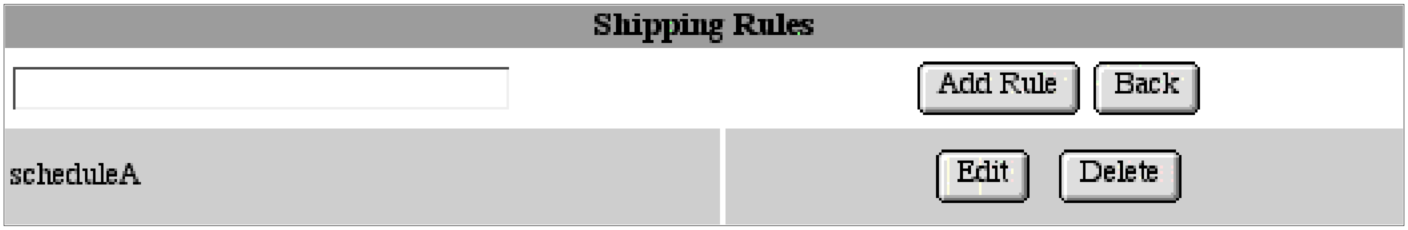
With a scope of basket, starting at 50 and ending at 0, decrement the price by 5 dollars for every 0 units of dollar amount.

Click the "Save" button to save the price rule configuration and return to the price rules interface.

Step 8: Configure basket shipping rules.

Return to the main configuration interface for the "Wholesale" shopping basket handle by clicking the "Back" button under the "Operation" column. Click the "Edit" button beside "Shipping Rules" to display the shipping rules interface.

To create a custom shipping rule, enter a name for the shipping rule and click the "Add Rule" button. For this example, create a shipping rule named "scheduleA" to offer a shipping schedule based on the total number of items purchased.


Shipping Rules

scheduleA

To configure the shipping rule, click the "Edit" button beside the shipping rule to display the shipping rule configuration interface.

The shipping rule configuration provides a selection for an overall base shipping cost (in addition to other item or basket shipping charges). The options for a shipping rule segment are the scope of the shipping rule (item or basket), the base shipping cost, the beginning and ending break points of the shipping rule, the dollar or percentage shipping charge, the units required to incur the shipping charge and the unit of measurement basis of the shipping charge. Multiple segments of a shipping rule allow you to configure multiple break points or multiple conditions. Shipping rules are performed as they are entered from top to bottom.

For this example, the shipping schedule is based on the total number of items purchased:

- $7 for 1 to 5 items.
- $5 for 6 to 10 items
- $3 for 11 to 15 items
- Free for 16 or more items

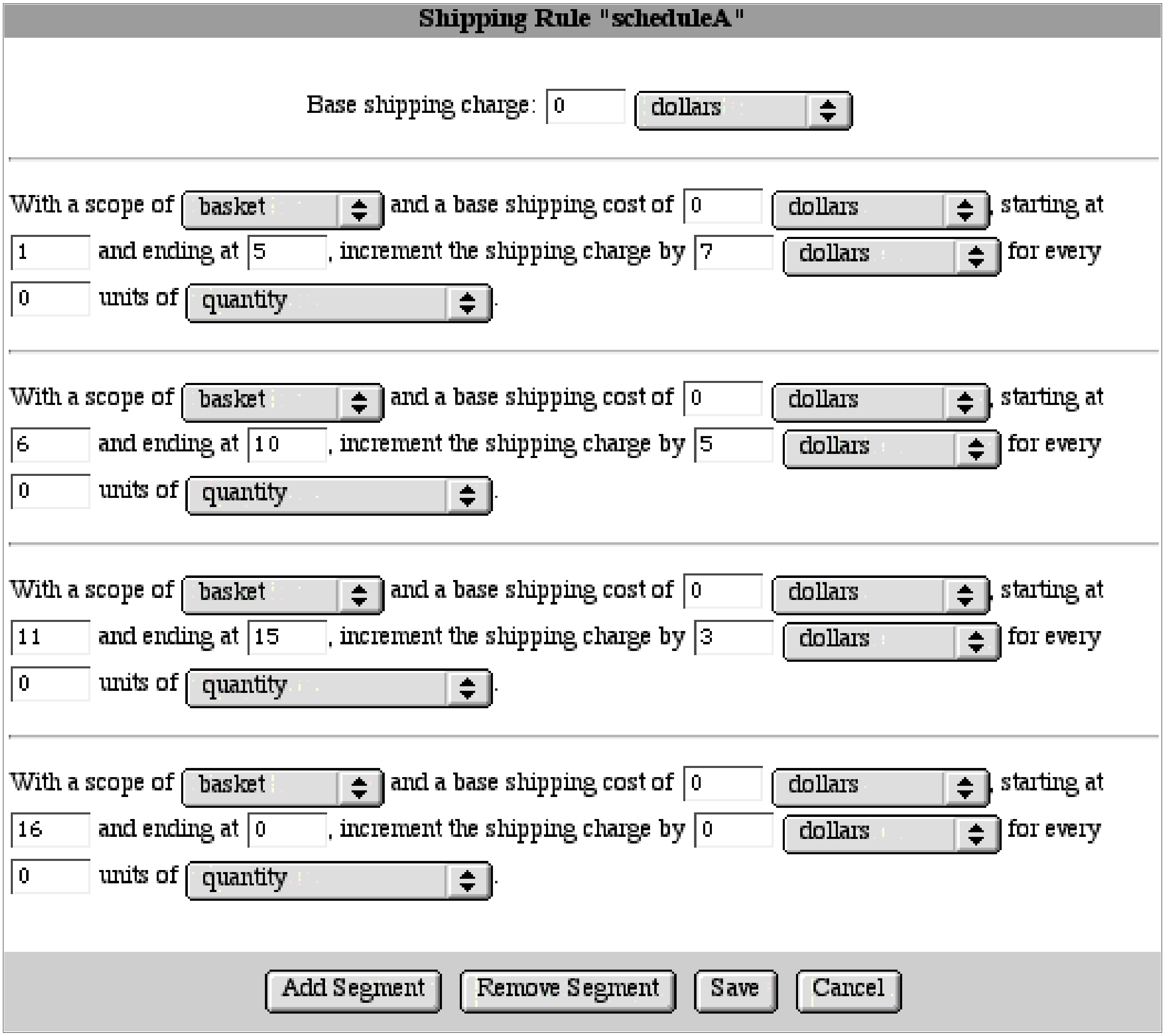
To create this shipping schedule, each shipping "condition" will be configured as a segment of the "scheduleA" shipping rule. Add 3 additional segements (for a total of 4). To add a segment, click the "Add Segment" button.

In the first segment, configure the shipping charges for 1 to 5 total items. The scope of the shipping charge is a basket. The base shipping cost is zero. The beginning break point is 1 and the ending break point is 5. The shipping charge is 7 dollars. The shipping charge is not applied to each item from the beginning break point to the ending break point (the charge only applies once when the beginning break point is reached), therefore the charge applies to zero units of quantity.

In the second segment, configure the shipping charges for 6 to 10 total items. The scope of the shipping charge is a basket. The base shipping cost is zero. The beginning break point is 6 and the ending break point is 10. The shipping charge is 5 dollars. The shipping charge is not applied to each item from the beginning break point to the ending break point (the charge only applies once when the beginning break point is reached), therefore the charge applies to zero units of quantity.

In the third segment, configure the shipping charges for 11 to 15 total items. The scope of the shipping charge is a basket. The base shipping cost is zero. The beginning break point is 11 and the ending break point is 15. The shipping charge is 3 dollars. The shipping charge is not applied to each item from the beginning break point to the ending break point (the charge only applies once when the beginning break point is reached), therefore the charge applies to zero units of quantity.

In the fourth segment, configure the shipping charges for 15 or more total items. The scope of the shipping charge is a basket. The base shipping cost is zero. The beginning break point is 15 and there is no ending point (i.e., "0"). The shipping charge is 0 dollars (it's free!). The shipping charge is not applied to each item from the beginning break point to the ending break point (the charge only applies once when the beginning break point is reached), therefore the charge applies to zero units of quantity.



Shipping Rule "scheduleA"

Base shipping charge: 0 dollars

With a scope of basket and a base shipping cost of 0 dollars, starting at 1 and ending at 5, increment the shipping charge by 7 dollars for every 0 units of quantity.

With a scope of basket and a base shipping cost of 0 dollars, starting at 6 and ending at 10, increment the shipping charge by 5 dollars for every 0 units of quantity.

With a scope of basket and a base shipping cost of 0 dollars, starting at 11 and ending at 15, increment the shipping charge by 3 dollars for every 0 units of quantity.

With a scope of basket and a base shipping cost of 0 dollars, starting at 16 and ending at 0, increment the shipping charge by 0 dollars for every 0 units of quantity.

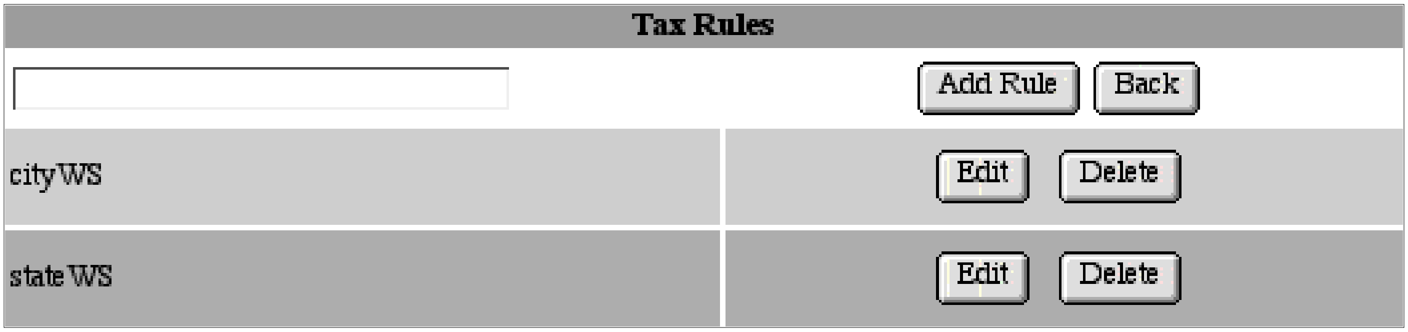
Click the "Save" button to save the shipping rule configuration and return to the shipping rules interface.

Step 9: Configure basket tax rules.

Return to the main configuration interface for the "Wholesale" shopping basket handle by clicking the "Back" button under the "Operation" column. Click the "Edit" button beside "Tax Rules" to display the tax rules interface.

For this example, the tax charge is 2% for city sales tax and 4% for state sales tax the total dollar amount of the basket. Each tax charge is entered as a separate tax rule, but the tax rules can be "chained" together to charge multiple taxes. Multiple segments of a tax rule can be added to charge taxes for multiple locations. For example, if you are charging a different state tax for North Carolina and South Carolina, you would enter two segments to the "stateWS" tax rule.

Create two new tax rules. Name the first tax rule, "cityWS" and the second tax rule "stateWS". To create a custom tax rule, enter a name for the tax rule and click the "Add Rule" button.

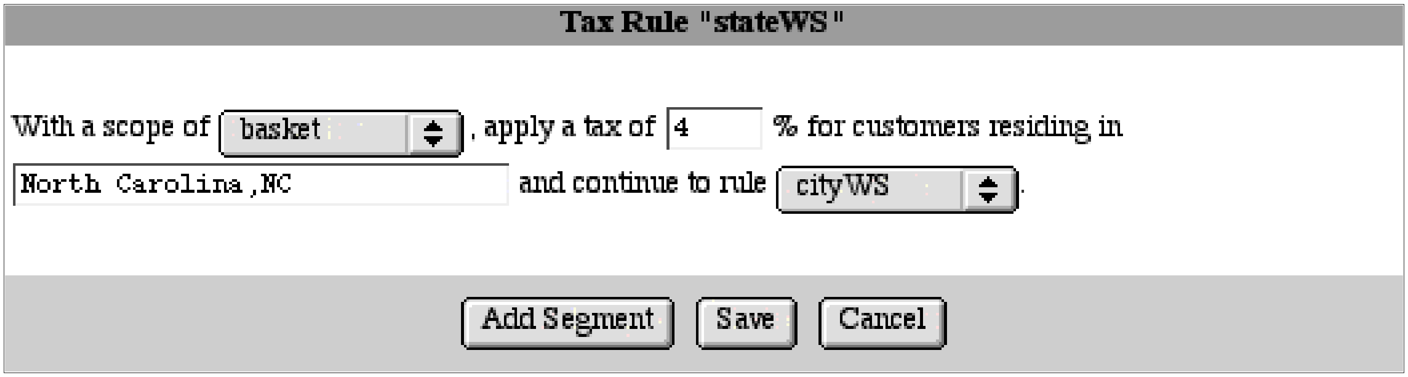


Tax Rules

cityWS	Edit Delete
stateWS	Edit Delete

To configure a tax rule, click the "Edit" button beside the tax rule. The options for a tax rule are the scope of the tax rule (item or basket), the percentage tax charge, the location(s) receiving the tax charge and the next tax rule to apply in the chain.

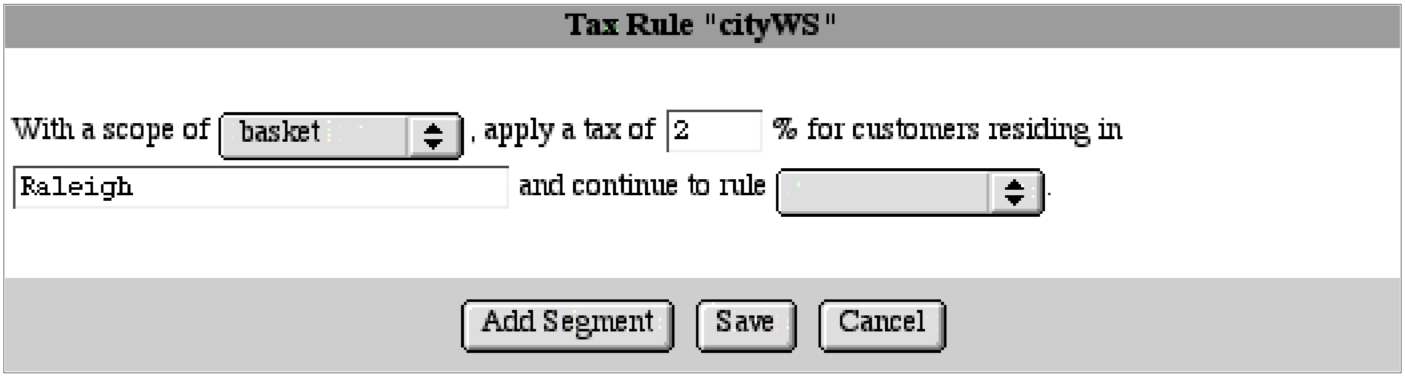
Configure the "stateWS" tax rule as a 4% tax for residents of North Carolina (NC). The scope of the tax charge is a basket. The tax charge is 4 percent. The location of the tax charge is North Carolina. Multiple locations or multiple values should be entered as a comma-delimited list (e.g., "North Carolina, NC"). After the "state" tax is applied for North Carolina, continue to the "cityWS" tax rule. Click "Save" to save the tax rule and return to the tax rules interface.



Tax Rule "stateWS"

With a scope of basket, apply a tax of 4 % for customers residing in North Carolina ,NC and continue to rule cityWS.

Configure the "cityWS" tax rule as a 2% tax for residents of Raleigh. The scope of the tax charge is a basket. The tax charge is 2 percent. The location of the tax charge is Raleigh. Multiple locations or multiple values should be entered as a comma-delimited list. After the "city" tax is applied for North Carolina, no additional taxes are charged. Click "Save" to save the tax rule and return to the tax rules interface.



Tax Rule "cityWS"

With a scope of basket, apply a tax of 2 % for customers residing in Raleigh and continue to rule .

Close the shopping basket admin page.

Step 10: Create product pages and open them in a text editor.

Create pages to display products for sale. You may choose any design and layout for products including pictures, descriptions, pricing information, etc. Insert placeholders for a quantity box for each product and a submit button. The default quantity box is 3 characters wide. Finally, open your product pages in a text editing program that allows you to view and modify the HTML and code of the page.

Step 11: Insert an mgiBuyMe tag and a submit button for each product.

Replace the quantity box placeholder for each product with an mgiBuyMe tag, productID parameter, name parameter, and price parameter. In the productID parameter, enter a unique code for the product or embed the product's unique ID from a database search. Product information is tracked through the shopping basket by the productID. In the name parameter, enter a short description of the product or embed the product's short description from a database search. In the price parameter, enter the per item price of each product as a decimal number or embed the product's price from a database search. Do not enter characters such as dollar signs in the price parameter.

Replace the submit button placeholder for each item (or for all items) with an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button (e.g., "Add to Shopping Basket").

A form action adds the quantity entered in the quantity box(s) to the shopping basket page for processing when the submit button is clicked. Enter the name of the shopping basket page (basket.mgi) in the action parameter of the <FORM> tag. Enter "post" in the method parameter of the <FORM> tag. Enclose the mgiBuyMe tag and "Add to Shopping Basket" submit button for each item with HTML form tags. If you want to allow customers to add multiple items to a shopping basket at one time, enclose all mgiBuyMe tags and a single submit button or multiple submit buttons with HTML form tags.

The default shopping basket uses tokens to track the purchases of individual customers. Enter one beginning mgiToken tag before all products and submit buttons, and enter one ending mgiToken tag after all products and submit buttons. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example product page.

```
<mgiToken>

<P><CENTER>

<TABLE WIDTH="500" CELLPADDING="3"
CELLSPACING="2">
<TR><TD WIDTH="33%" VALIGN="TOP">

<FORM ACTION="basket.mgi" METHOD="POST">
<P><CENTER><B><FONT SIZE="+1">The Race
</FONT></B>
</CENTER></P>
<P><CENTER>
<IMG SRC="images/GCL632S.JPG" BORDER="0"
WIDTH="150" HEIGHT="100" ALIGN="BOTTOM">
<BR>Item: GCL632
<BR>Artist: Johansen
<BR>24&quot; x 36&quot;
<BR><MGIBUYME PRODUCTID="GCL632"
NAME="The Race" PRICE="16.00">
@ $16.00 ea.
<BR><MGIBUTTON VALUE="Buy Now!">
</CENTER>
</FORM>

</TD>
<TD WIDTH="33%" VALIGN="TOP">

<FORM ACTION="basket.mgi" METHOD="POST">
<P><CENTER>
<B><FONT SIZE="+1">Composition No. 6
</FONT></B>
</CENTER></P>
<P><CENTER>
<IMG SRC="images/GCL641S.JPG" BORDER="0"
WIDTH="150" HEIGHT="100" ALIGN="BOTTOM">
<BR>Item: GCL641
<BR>Artist: Kandinsky
```

```
<BR>24&quot; x 36&quot;
<BR><MGIBUYME PRODUCTID="GCL641"
NAME="Composition No. 6" PRICE="16.00">
@ $16.00 ea.
<BR><MGIBUTTON VALUE="Buy Now!">
</CENTER>
</FORM>

</TD>
</TR>
</TABLE>

</CENTER>
</P>

</mgiToken>
```

Step 12: Save the product pages.

Save the changes you have made to the product pages.

Step 13: Create a shopping basket page and open it in a text editor.

Create a page named "basket.mgi" to display the contents of a customer's shopping basket and open the page in a text editing program that allows you to view and modify the HTML and code.

Step 14: Insert the mgiShoppingBasket tag, rule parameters and mgiButton tag.

Insert your cursor in the HTML of the shopping basket page where you want the shopping basket to display and enter the beginning mgiShoppingBasket tag, handle parameter, rule parameters and ending mgiShoppingBasket tag. In the handle parameter, enter the name of the shopping basket handle you created in the shopping basket administration (i.e., "Wholesale").

When implementing a basket price rule, you must include the priceRule parameter in the mgiShoppingBasket tag. In the priceRule parameter, enter the name of the price rule to apply to the basket.

When implementing a basket shipping rule, you must include the shippingRule parameter in the mgiShoppingBasket tag. In the shippingRule parameter, enter the name of the shipping rule to apply to the basket.

Below the mgiShopping basket tag, enter an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to modify shopping basket quantities (e.g., "Modify Quantity").

Below the mgiShoppingBasket tag and modify quantity button, enter a second mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to proceed to the check out page (e.g., "Check Out").

On the shopping basket page, form actions perform two functions. One form action modifies the contents of the shopping basket (i.e., changes quantities). Enclose the mgiShoppingBasket tag and mgiButton tag used to modify the shopping basket with HTML form tags. Enter the name of the shopping basket page (basket.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag.

Another form action links customers to the check out page to enter payment, billing and shipping information. Enclose the mgiButton tag used to check out with HTML form tags. Enter the name of the check out page (checkout.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag. Note: To use a secure server to collect payment information, enter the URL to a secure server running MGI in the action parameter of the <FORM> tag (e.g., https://secure.domain.com/checkout.mgi). You may substitute a text link for the check out button and form tags.

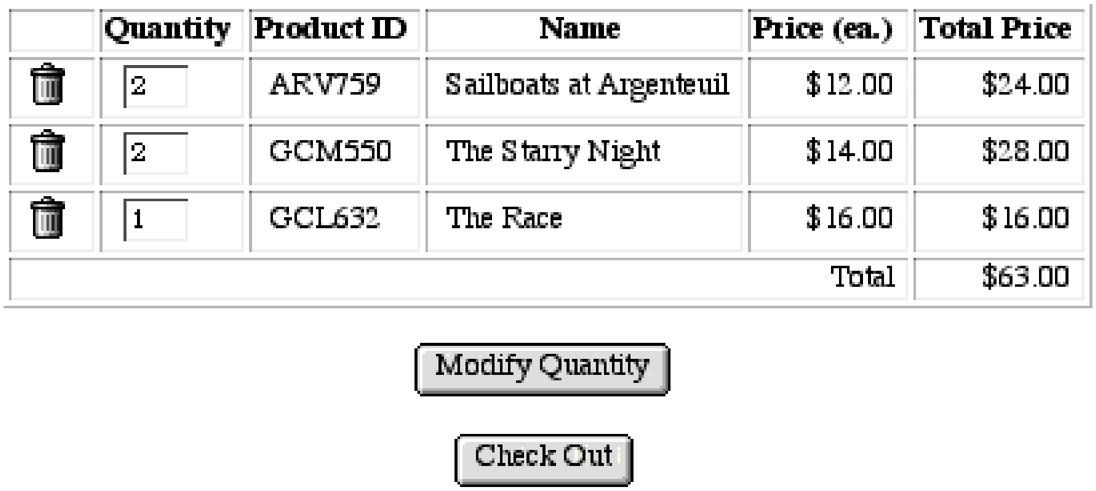
The default shopping basket uses tokens to track the purchases of individual customers. Enter a beginning mgiToken tag before the first HTML form tag and enter an ending mgiToken tag after the last closing HTML form tag. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example shopping basket page.

```
<mgiToken>

<center>

<h2>Shopping Basket</h2>

<form action="basket.mgi" method="post">

<p><mgiShoppingBasket handle="Wholesale"
priceRule="5dollars">
</mgiShoppingBasket>
```

```
<p><mgiButton value="Modify Quantity">

</form>

<form action="checkout.mgi" method="post">

<mgiButton value="Check Out">

</form>

</center>

</mgiToken>
```

The default shopping basket contents display in a table with columns for product removal, product quantity, product ID, product name, product price, multiplied product subtotals and order total. Price discounts are deducted and displayed in the "Total" price.



	Quantity	Product ID	Name	Price (ea.)	Total Price
🗑	2	ARV759	Sailboats at Argenteuil	$12.00	$24.00
🗑	2	GCM550	The Starry Night	$14.00	$28.00
🗑	1	GCL632	The Race	$16.00	$16.00
				Total	$63.00

Step 15: Save the shopping basket page.

Save the changes you have made to the shopping basket page (basket.mgi).

Step 16: Create a check out page and open it in a text editor.

Create a page named "checkout.mgi" to collect payment, billing and shipping information from the customer and open the page in a text editing program that allows you to modify the HTML and code of the page.

Step 17: Insert the mgiCollectUserInfo and mgiButton tags.

Insert your cursor in the HTML of the check out page where you want the payment, billing and shipping information tables to display and enter the mgiCollectUserInfo tag, handle parameter and shoppingBasketURL parameter. In the handle parameter, enter the name of the shopping basket handle you created in the shopping basket administration (i.e., "Wholesale"). In the shoppingBasketURL parameter, enter the full URL to the region where items were added to the shopping basket (and thus where the internal MGI shopping basket database was created and populated).

Below the mgiCollectUserInfo tag, enter an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to proceed to the order confirmation page (e.g., "Confirm Order").

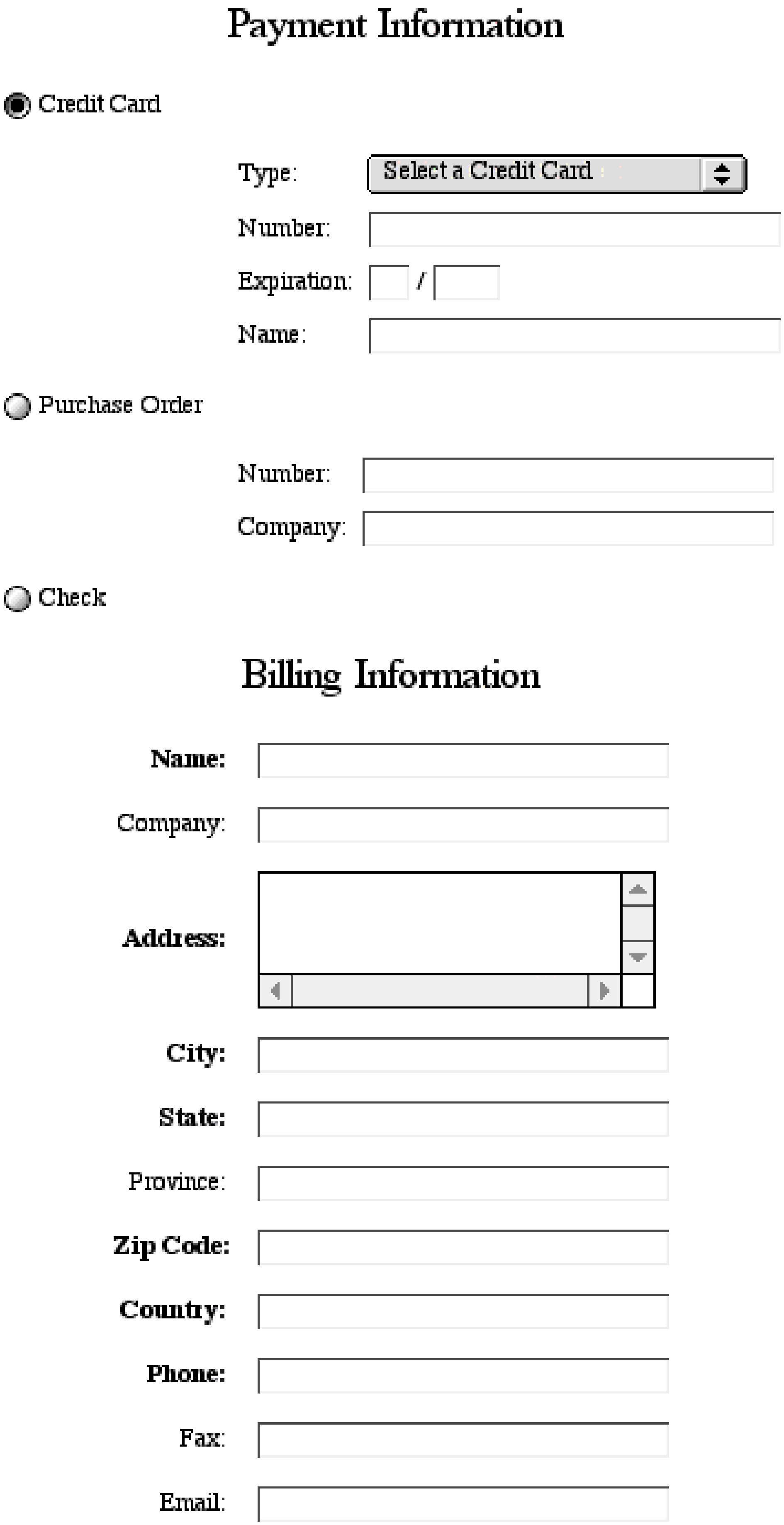
Form actions on the check out page link customers to the order confirmation page to review their order and customer information. Enclose the mgiCollectUserInfo tag, mgiButton tag, and any custom form elements with HTML form tags. Enter the name of the order confirmation page (confirm.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag.

The default shopping basket uses tokens to track the purchases of individual customers. Enter a beginning mgiToken tag before the beginning HTML form tag and enter an ending mgiToken tag after the ending HTML form tag. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example check out page.

```
<mgiToken>

<form action="confirm.mgi" method="post">

<center>

<mgiCollectUserInfo handle="Wholesale"
shoppingBasketURL="http://www.domain.com/shop/">

<p><mgiButton value="Confirm Order">

</center>

</form>

</mgiToken>
```

The default check out page display payment, billing and shipping information in tables. Custom form elements appear as you format them.



Payment Information

● Credit Card

Type: Select a Credit Card

Number:

Expiration: /

Name:

○ Purchase Order

Number:

Company:

○ Check

Billing Information

Name:

Company:

Address:

City:

State:

Province:

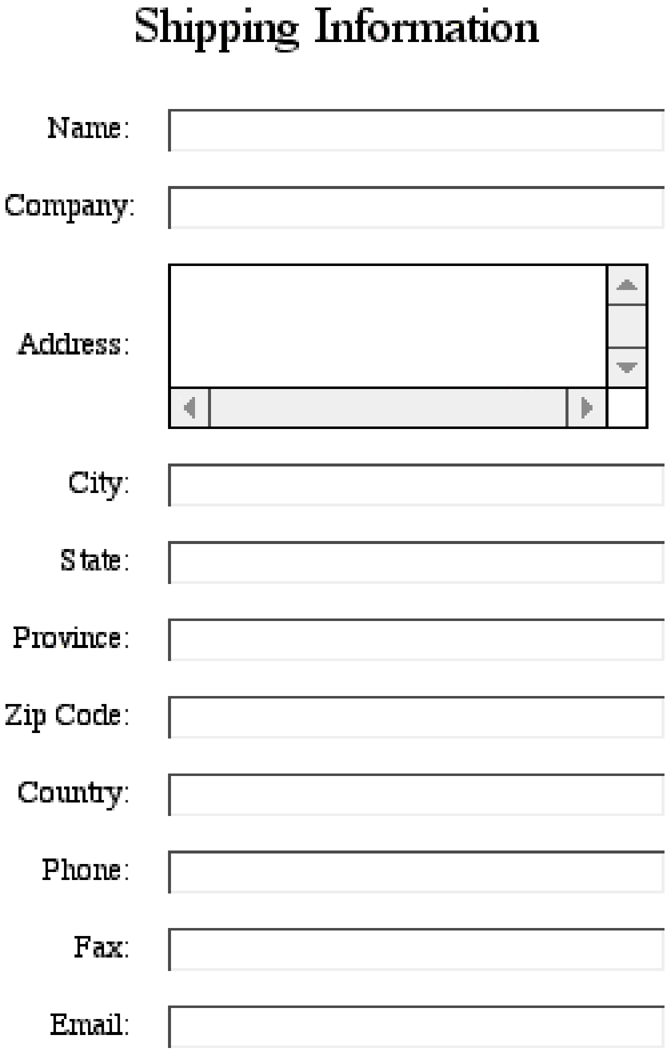
Zip Code:

Country:

Phone:

Fax:

Email:



Shipping Information

Name:

Company:

Address:

City:

State:

Province:

Zip Code:

Country:

Phone:

Fax:

Email:

Step 18: Save the check out page.

Save the changes you have made to the check out page.

Step 19: Create a confirm order page and open it in a text editor.

Create a page named "confirm.mgi" to present the final shopping basket, payment information, billing information, and shipping information for review. Open the confirm order page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 20: Insert the mgiConfirmOrder tag, rule parameters, customerLocation parameter and mgiButton tag.

Insert your cursor in the HTML of the confirm order page where you want the shopping basket, payment, billing, shipping, and additional information tables to display and enter the beginning mgiConfirmOrder tag, handle parameter, shoppingBasketURL parameter, rule parameters, customerLocation parameter and ending mgiConfirmOrder tag.

In the handle parameter, enter the name of the shopping basket handle you created in the shopping basket administration (i.e., "Wholesale"). In the shoppingBasketURL parameter, enter the full URL to the region where items were added to the shopping basket (and thus where the internal MGI shopping basket database was created and populated).

When implementing a basket price rule, you must include the priceRule parameter in the mgiConfirmOrder tag. In the priceRule parameter, enter the name of the price rule to apply to the basket.

When implementing a basket shipping rule, you must include the shippingRule parameter in the mgiConfirmOrder tag. In the shippingRule parameter, enter the name of the shipping rule to apply to the basket.

When implementing a basket tax rule, you must include the taxRule parameter in the mgiConfirmOrder tag. In the taxRule parameter, enter the name of the tax rule to apply to the basket. For tax rules, you must also collect the customer's location(s) from their billing information and embed it in the customerLocation parameter of the mgiConfirmOrder parameter. For "chained" tax rules, embed a comma-delimited list of the customer's location from the top-level tax rule to the bottom-level tax rule. The location list must be in the same order as the tax rule chain. Create the comma-delimited list in a variable (using mgiSet) before the mgiConfirmOrder tag and embed the variable in the customerLocation parameter of the mgiConfirmOrder tag (using mgiGet). In this example, the customer's state and city are compiled in a variable and embedded in the customerLocation parameter.

Below the mgiConfirmOrder tag, enter an mgiButton tag and value parameter. In the value parameter, enter the value that will display on the submit button to proceed to the final order processing page (e.g., "Complete Order").

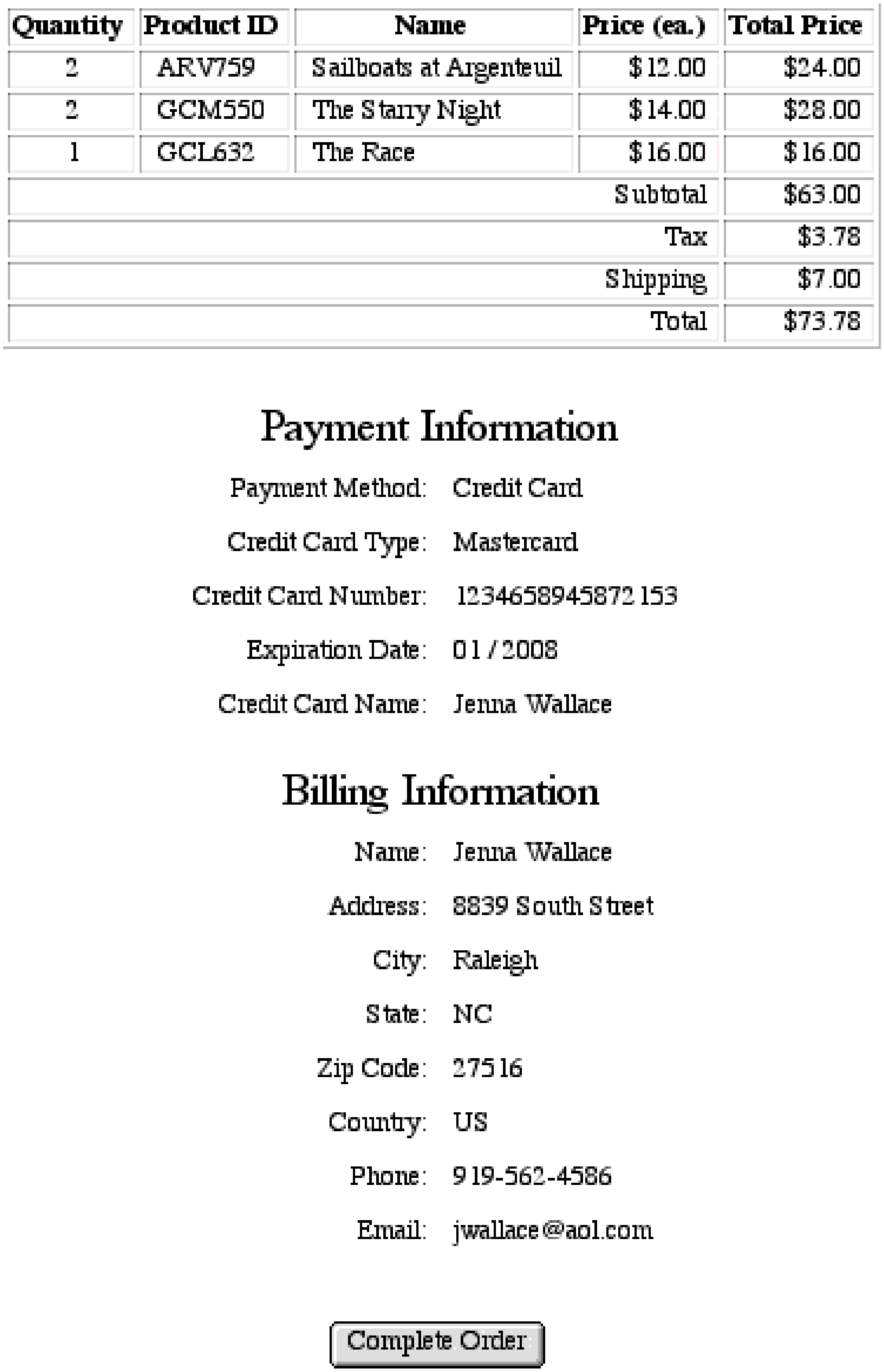
Form actions on the confirm order page link customers to the order processing page to complete and send their order. Enclose the mgiConfirmOrder tag and mgiButton tag with HTML form tags. Enter the name of the order processing page (process.mgi) in the action parameter of the <FORM> tag and enter "post" in the method parameter of the <FORM> tag.

The default shopping basket uses tokens to track the purchases of individual customers. Enter a beginning mgiToken tag before the beginning HTML form tag and enter an ending mgiToken tag after the ending HTML form tag. The mgiToken tags should enclose all links on the page including HREFs and FORM actions.

The following is an example confirm order page.

```
<mgiToken>

<form action="process.mgi" method="post">

<center>

<mgiSet name="locations">
<mgiPostArgument name="bSTATE">,<mgiPostArgument
name="bCITY">
</mgiSet>

<mgiConfirmOrder handle="Wholesale"
shoppingBasketURL="http://www.domain.com/shop/"
priceRule="5dollars" shippingRule="scheduleA"
taxRule="stateWS"
customerLocation={mgiGet name="locations"}>
</mgiConfirmOrder>

<p><mgiButton value="Complete Order">

</center>

</form>

</mgiToken>
```

The default confirm order page displays completed elements of the payment, billing and shipping information. The confirm order page also displays the calculated price (reflected in the Subtotal), shipping and tax charges.



Quantity	Product ID	Name	Price (ea.)	Total Price
2	ARV759	Sailboats at Argenteuil	$12.00	$24.00
2	GCM550	The Starry Night	$14.00	$28.00
1	GCL632	The Race	$16.00	$16.00
			Subtotal	$63.00
			Tax	$3.78
			Shipping	$7.00
			Total	$73.78

Payment Information

Payment Method:	Credit Card
Credit Card Type:	Mastercard
Credit Card Number:	1234658945872153
Expiration Date:	01 / 2008
Credit Card Name:	Jenna Wallace

Billing Information

Name:	Jenna Wallace
Address:	8839 South Street
City:	Raleigh
State:	NC
Zip Code:	27516
Country:	US
Phone:	919-562-4586
Email:	jwallace@aol.com

Step 21: Save the confirm order page.

Save the changes you have made to the confirm order page.

Step 22: Create an order processing page and open it in a text editor.

Create an page named "process.mgi" to present a "thank you for purchasing" message to customers and to send a formatted email of the order to a specified address. Open the order processing page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 23: Insert the mgiSendOrder tag, rule parameters and customerLocation parameter.

Enter a "thank you" message to display to customers when they complete their order.

At any place on the order processing page, enter a beginning mgiSendOrder tag, handle parameter, shoppingBasketURL parameter, rule parameters, customerLocation parameter, to parameter, from parameter, mailServer parameter, subject parameter, and ending mgiSendOrder tag.

In the handle parameter, enter the name of the default handle you created in the shopping basket administration (i.e., "Posters"). In the shoppingBasketURL parameter, enter the full URL to the region where items were added to the shopping basket (and thus where the internal MGI shopping basket database was created and populated).

When implementing a basket price rule, you must include the priceRule parameter in the mgiSendOrder tag. In the priceRule parameter, enter the name of the price rule to apply to the basket.

When implementing a basket shipping rule, you must include the shippingRule parameter in the mgiSendOrder tag. In the shippingRule parameter, enter the name of the shipping rule to apply to the basket.

When implementing a basket tax rule, you must include the taxRule parameter in the mgiSendOrder tag. In the taxRule parameter, enter the name of the tax rule to apply to the basket. For tax rules, you must also collect the customer's location(s) from their billing information and embed it in the customerLocation parameter of the mgiSendOrder parameter. For "chained" tax rules, embed a comma-delimited list of the customer's location from the top-level tax rule to the bottom-level tax rule. The location list must be in the same order as the tax rule chain. Create the comma-delimited list in a variable (using mgiSet) before the mgiSendOrder tag and embed the variable in the customerLocation parameter of the mgiSendOrder tag (using mgiGet). In this example, the customer's state and city are compiled in a variable and embedded in the customerLocation parameter.

In the "to" parameter, enter the email address to receive shopping basket orders. In the "from" parameter, enter the email address that appears in the "from" line of shopping basket orders. In the mailServer parameter, enter the address for the outgoing SMTP mail server of your domain (e.g., mail.domain.com). In the subject parameter, enter the subject of the order email. The mgiSendOrder tag does not display information to the customer.

The following is an example order processing page.

```
<center>

<h2>Order Complete</h2>

<p>Thank you for ordering.  Your order has been
processed and you should receive the merchandise
within 2 to 4 weeks.

<mgiSet name="locations">
<mgiPostArgument name="bSTATE">,<mgiPostArgument
name="bCITY">
</mgiSet>

<mgiSendOrder handle="Posters"
shoppingBasketURL="http://www.domain.com/shop/"
to="sales@domain.com" from="webmaster@domain.com"
mailServer="mail.domain.com"
subject="Online Order"
priceRule="5dollars" shippingRule="scheduleA"
taxRule="stateWS"
customerLocation={mgiGet name="locations"}>
</mgiSendOrder>

</center>
```

The default order email is formatted with payment, billing, shipping, and order information.

```
Payment Information
-------------------
 Payment Method:  Credit Card
           Type:  Mastercard
         Number:  1234658945872153
Expiration Date:  01/2008
           Name:  Jenna Wallace

Billing Information
-------------------
           Name:  Jenna Wallace
        Address:  8839 South Street
```

```
           City:  Raleigh
          State:  NC
       Zip Code:  27516
        Country:  US
          Phone:  919-562-4586
          Email:  jwallace@aol.com

Product Information
--------------------
       Quantity:  2
     Product ID:  ARV759
           Name:  Sailboats at Argenteuil
    Total Price:  $24.00

       Quantity:  2
     Product ID:  GCM550
           Name:  The Starry Night
    Total Price:  $28.00

       Quantity:  1
     Product ID:  GCL632
           Name:  The Race
    Total Price:  $16.00

       Subtotal:  $63.00
            Tax:  $3.78
       Shipping:  $7.00
          Total:  $73.78
```

Step 24: Save the order processing page.

Save the changes you have made to the order processing page.

Step 25: Tokenize all pages of the web site.

The default shopping basket uses tokens to track the purchases of individual customers. Using a text editor, enter a beginning mgiToken tag and ending mgiToken tag on all additional pages of the web site. The mgiToken tags should enclose all links on the pages including HREFs and FORM actions.

```
<mgiToken>
All tags and links appear here.
</mgiToken>
```

Step 26: Save the web site pages.

Save the changes that you make as you tokenize each page.

Step 27: FTP all pages to the web server running MGI.

Upload all pages of the web site to the web server using an FTP program. If you are using a secure server, upload **only** the check out, order confirmation and order processing pages to the secure portion of your web site. Do not attempt to use the shopping basket page on a secure server.

Step 28: View a product page in a web browser and purchase a product.

View a product page in a web browser. Enter a quantity to purchase in the quantity text field and click the "Add to Shopping Basket" button. The item(s) appears in the shopping basket display on the shopping basket page. Click the "Check Out" button to proceed to the check out page. On the check out page, enter a payment choice, billing information and shipping information (if it is different). Click the "Confirm Order" button to proceed to the confirm order page. On the confirm order page, review your order, payment information, billing information and shipping information, then click the "Complete Order" button to complete the ordering process. The message from the order processing page is displayed and the shopping basket order is formatted and emailed.

[Return to the Shopping Online Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Customize the Shopping Basket Display

Introduction

Please review the basic shopping online tutorial. This tutorial is an extension of the basic shopping online tutorial and explains how to customize one aspect of the shopping basket system. View the other advanced tutorials to customize other aspects of the shopping basket system.

The default shopping basket table displays a trashcan icon to delete items, a modifiable quantity field, the product ID, product name, product price, extended product price, shipping total and total (does not include tax which is added in the mgiConfirmOrder tag). You may customize the shopping basket text, layout, item information and totals using the mgiShoppingBasket placeholders and variables

MGI Tags

- mgiGet
- mgiShoppingBasket

Steps

1. Open the shopping basket page in a text editor.
2. Customize the shopping basket.
3. Save the shopping basket page.
4. FTP the shopping basket page to the web server running MGI.
5. Add items to the shopping basket.

Step 1: Open the shopping basket page in a text editor.

Open the shopping basket page (e.g., shoppingbasket.mgi) in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Customize the shopping basket.

If the default shopping basket does not fit with your site design, you may customize the shopping basket layout and the content of the shopping basket using the mgiShoppingBasket placeholders and variables. With a custom layout you can add font properties, cell colors, images, etc.

In a custom shopping basket, the code in the body of the mgiShoppingBasket tags is repeated for each item in the shopping basket and the placeholders are replaced with the item's specific information. After the mgiShoppingBasket tags, the shopping basket variables display information calculated from the current shopping basket contents.

You may format the text and HTML of your shopping basket in any way. The following placeholders are available to display information for each item. The placeholders will be replaced with each product's specific information.

- **&mgiDBFieldQuantity;** is the quantity of the product.
- **&mgiDBFieldQuantityMultiplier;** is the quantity multiplier of the product.
- **&mgiDBFieldProductID;** is the product ID.
- **&mgiDBFieldName;** is the product name (short description).
- **&mgiDBFieldDescription;** is the product description
- **&mgiDBFieldPrice;** is the product price.
- **&mgiDBFieldShipping;** is the product shipping cost.
- **&mgiDBFieldQualifier1;** is the first product qualifier.
- **&mgiDBFieldQualifier2;** is the second product qualifier.
- **&mgiDBFieldQualifier3;** is the third product qualifier.
- **&mgiDBFieldQualifier4;** is the fourth product qualifier.
- **&mgiDBFieldQualifier5;** is the fifth product qualifier.
- **&mgiDBFieldWeight;** is the product weight.
- **&mgiSBDeleteTrashcan;** is a clickable trashcan icon that removes the product from the shopping basket.
- **&mgiSBDeleteRadioButtons;** is a set of Keep/Remove radio buttons that allows the product to be removed from the shopping basket.
- **&mgiSBModifiableQuantity;** is a text input containing the quantity purchased for the product. This value can be modified.
- **&mgiSBItemPriceTotal;** is the price of a product multiplied by the quantity purchased and includes any price rules that have been applied.

The following variables are available anywhere after the ending mgiShoppingBasket tag to display order totals. Variables can be displayed with the mgiGet tag.

- **mgiSBShippingTotal** is the shipping price of the order after all shipping rules have been applied.
- **mgiSBTotal** is the total price of the order after all price rules have been applied.

Customizing the display of the shopping basket page does not require you to customize any other aspect of the shopping basket system. The shopping basket may be customized independently of the check out page, order confirmation page and order email.

The following is the code of a custom shopping basket:

```
<mgiToken>

<form action="shoppingbasket.mgi" method="post">

<p><table cellpadding="5" cellspacing="1" border="0">
<tr bgcolor="#EEEECC">
<td align="center"><font size="-1"
face="helvetica, arial, sans-serif">
<b>Remove</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Qty</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Description</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Price</b></font></td>
</tr>

<mgiShoppingBasket handle="Default">

<tr>
<td align="center">&mgiSBDeleteTrashcan;</td>
<td>&mgiSBModifiableQuantity;</td>
<td><font size="-1" face="helvetica, arial, sans-serif">
&mgiDBFieldName;</font></td>
<td align="center">
<font size="-1" face="helvetica, arial, sans-serif">
$&mgiSBItemPriceTotal;</font></td>
</tr>

</mgiShoppingBasket>

<tr bgcolor="#EEEECC">
<td colspan="2"><font size="-2"
face="helvetica, arial, sans-serif">
Tax and Shipping will be calculated during Checkout.
</font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Subtotal</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
```

```
<b>$<mgiGet name="mgiSBTotal"></b></font></td>
</tr>
</table>

<p><mgiButton name="Update Basket">

<p><a href="checkout.mgi">Proceed to Checkout</a>

</form>

</mgiToken>
```

Step 3: Save the shopping basket page.

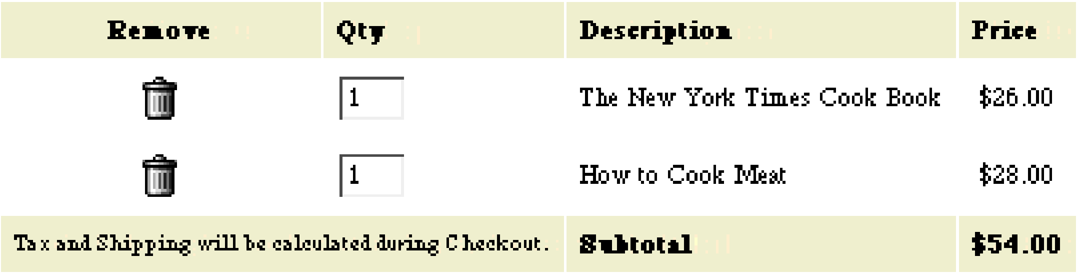
Save the changes you have made to the shopping basket page.

Step 4: FTP the shopping basket page to the web server running MGI.

Upload the shopping basket page from your local computer to the web server using an FTP program.

Step 5: Add items to the shopping basket.

Add items to your shopping basket to display the shopping basket page. The custom shopping basket in this example has the following display:



Remove	Qty	Description	Price
🗑	1	The New York Times Cook Book	$26.00
🗑	1	How to Cook Meat	$28.00
Tax and Shipping will be calculated during Checkout.		Subtotal	$54.00

[Return to the Shopping Online Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Customize the Payment, Billing and Shipping Information Display

Introduction

Please review the basic shopping online tutorial. This tutorial is an extension of the basic shopping online tutorial and explains how to customize one aspect of the shopping basket system. View the other advanced tutorials to customize other aspects of the shopping basket system.

In the shopping basket admin you may choose one of three methods of displaying payment, billing and shipping information in the "Customer Information Options": Automatic, Manual and Selective. The default of the shopping basket is to use the Automatic method and the mgiCollectUserInfo tag to display the default payment, billing, and shipping information tables with all available options. Note that all methods, including Automatic, allow you to select the type of credit cards you offer. To customize the display of payment, billing and shipping information, you have 3 options.

1. **Use the Selective Method in the shopping basket admin and the mgiCollectUserInfo tag**. This method allows you to choose which default fields to display and require. This method does **not** require the customization of the order confirmation and order email.
2. **Use the Automatic or Selective Method and create a custom layout using the default**

field names and form elements for payment, billing and shipping information. This method does **not** require the customization of the order confirmation and order email.

3. **Use the Manual Method and create a custom layout and custom fields for payment, billing, shipping or other information**. This method requires the customization of the order confirmation and order email.

Option 1, Selective Method with mgiCollectUserInfo, and Options 2 and 3, custom layouts and forms, are explained separately below. Options 2 and 3 differ only with respect to the form field names and are therefore included in the same tutorial.

MGI Tags

- mgiCollectUserInfo
- mgiShoppingBasket (admin mode)

Option 1: Selective Method with mgiCollectUser Info

Steps

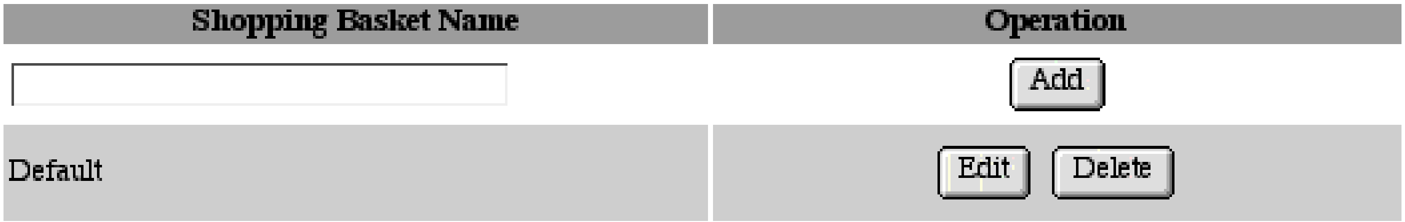
1. View the shopping basket administration page in a web browser.
2. Edit and Save the Customer Information Options.
3. Add items to the shopping basket and check out.

Step 1: View the shopping basket administration page in a web browser.

View the shopping basket administration page (e.g., sbadmin.mgi) in a web browser. The first screen of the web-based, administration interface displays the current shopping basket configurations (handles)

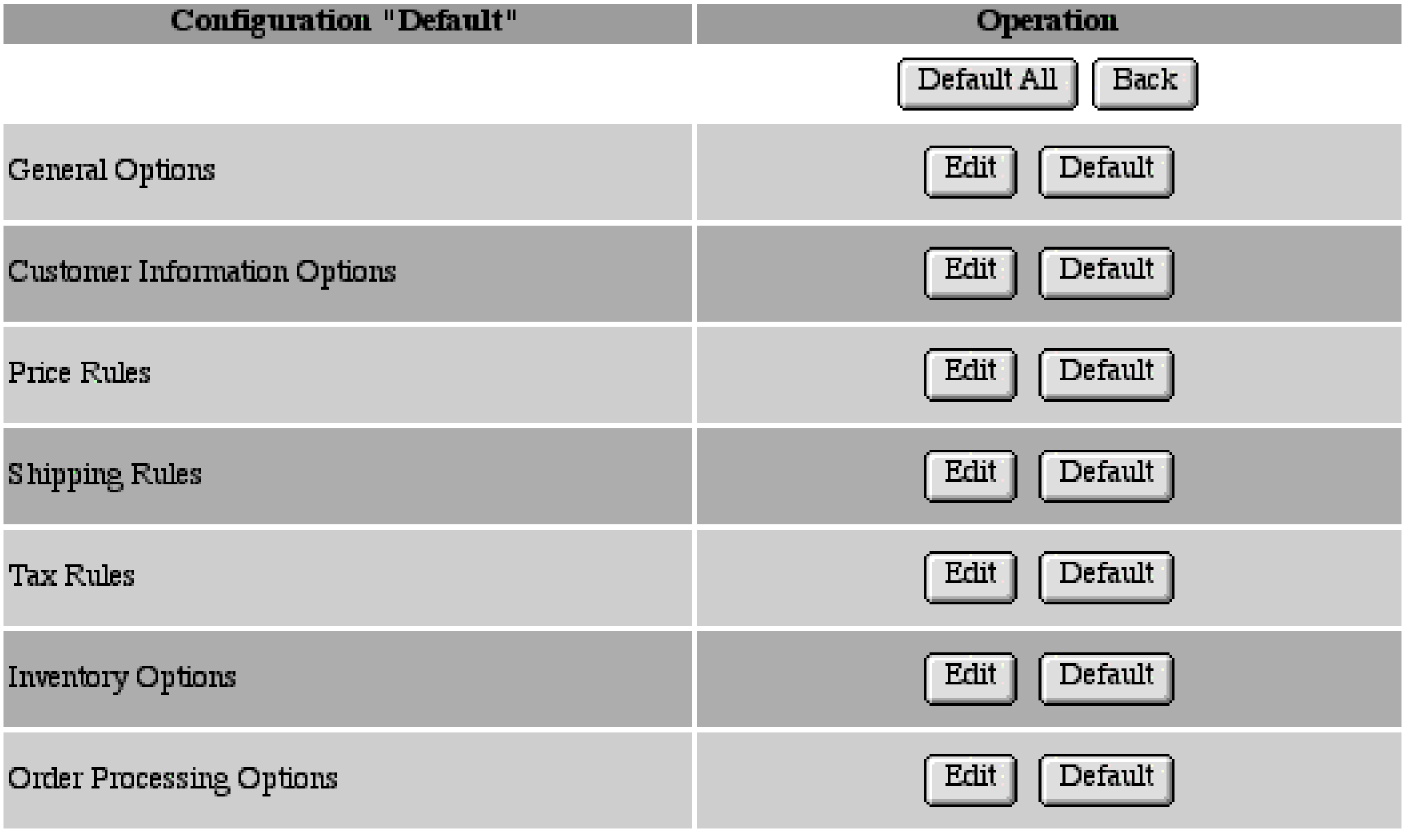
Step 2: Edit and Save the Customer Information Options".

Click the "Edit" button beside the shopping basket configuration (handle) to edit.



Shopping Basket Name	Operation
	Add
Default	Edit Delete

Click the "Edit" button beside "Customer Information Options".



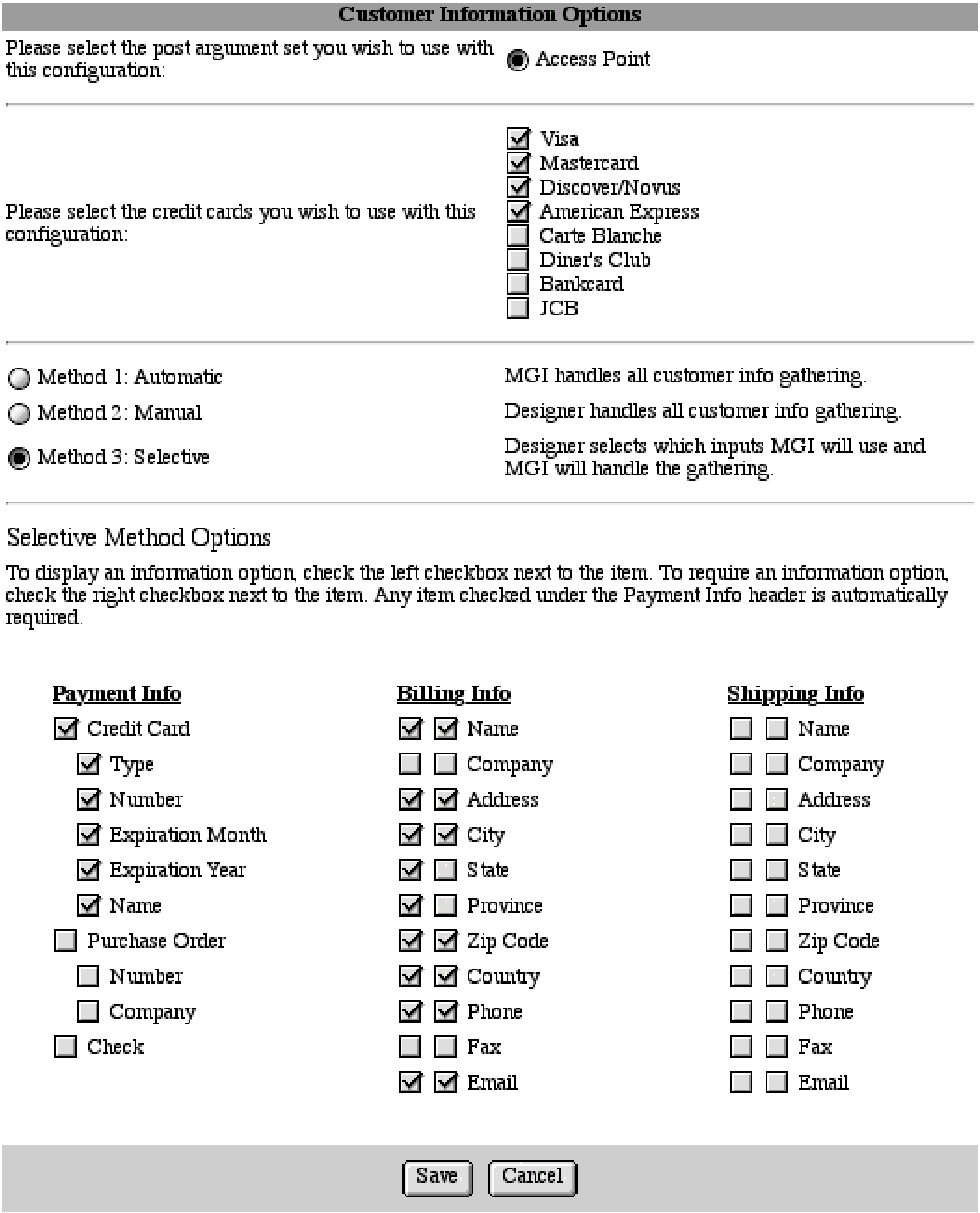
Configuration "Default"	Operation
	Default All Back
General Options	Edit Default
Customer Information Options	Edit Default
Price Rules	Edit Default
Shipping Rules	Edit Default
Tax Rules	Edit Default
Inventory Options	Edit Default
Order Processing Options	Edit Default

In the second section, check the boxes beside each type of credit card that you accept.

In the third section, select the "Selective" method.

Under "Selective Method Options" check the left box beside any field you wish to display. Check the right box beside any field you wish to require. In this example, all credit card options are selected in addition to the Name, full address, phone and email for billing. All billing information is required except State and Province. Shipping information is not displayed or required in this example.

Click the "Save" button to save the changes to the shopping basket configuration. The main menu displays after the configuration is saved.



Customer Information Options

Please select the post argument set you wish to use with this configuration: ● Access Point

Please select the credit cards you wish to use with this configuration:

☑ Visa
☑ Mastercard
☑ Discover/Novus
☑ American Express
□ Carte Blanche
□ Diner's Club
□ Bankcard
□ JCB

○ Method 1: Automatic	MGI handles all customer info gathering.
○ Method 2: Manual	Designer handles all customer info gathering.
● Method 3: Selective	Designer selects which inputs MGI will use and MGI will handle the gathering.

Selective Method Options

To display an information option, check the left checkbox next to the item. To require an information option, check the right checkbox next to the item. Any item checked under the Payment Info header is automatically required.

Payment Info

☑ Credit Card
- ☑ Type
- ☑ Number
- ☑ Expiration Month
- ☑ Expiration Year
- ☑ Name

□ Purchase Order
- □ Number
- □ Company

□ Check

Billing Info

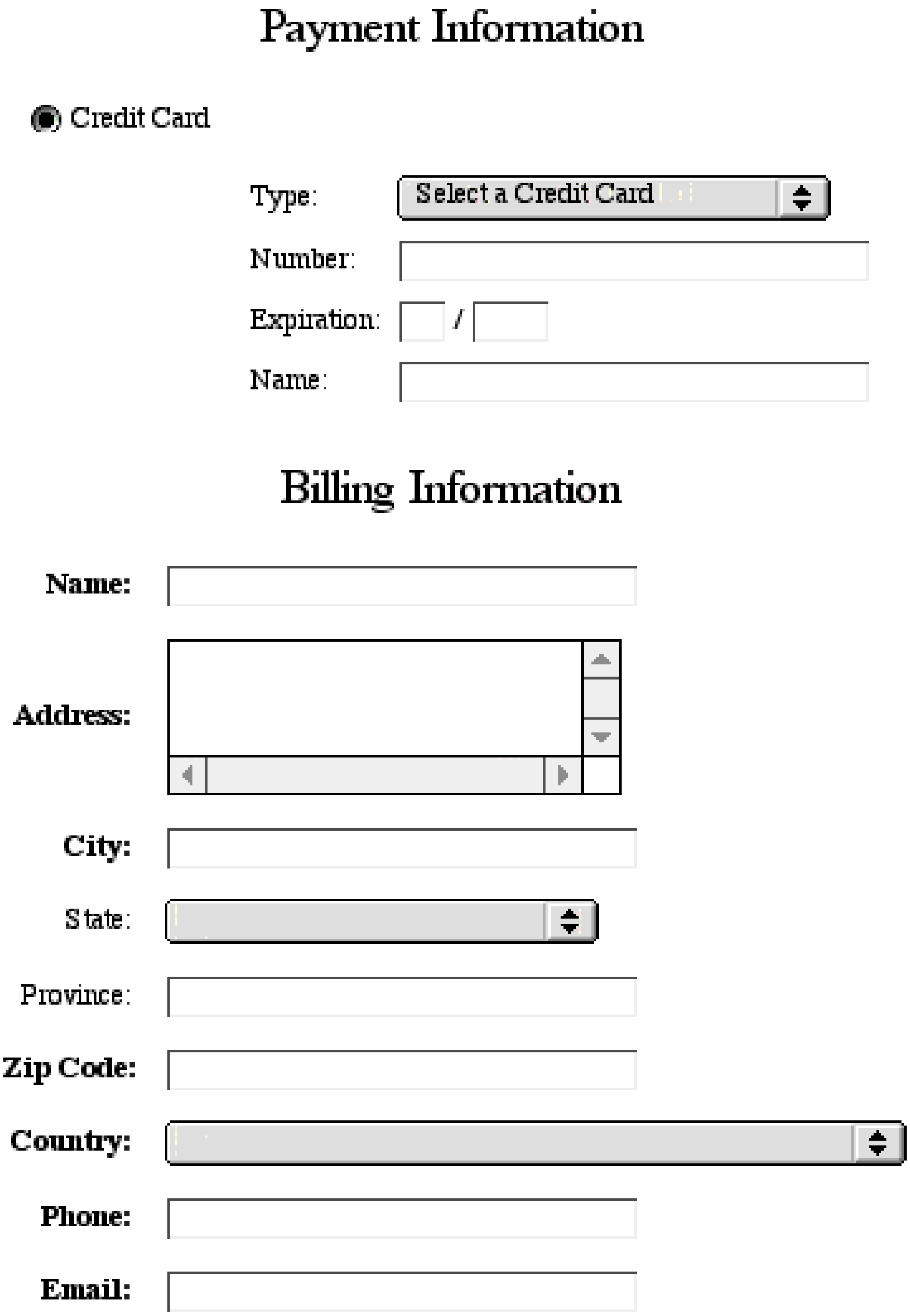
☑ ☑ Name
□ □ Company
☑ ☑ Address
☑ ☑ City
☑ □ State
☑ □ Province
☑ ☑ Zip Code
☑ ☑ Country
☑ ☑ Phone
□ □ Fax
☑ ☑ Email

Shipping Info

□ □ Name
□ □ Company
□ □ Address
□ □ City
□ □ State
□ □ Province
□ □ Zip Code
□ □ Country
□ □ Phone
□ □ Fax
□ □ Email

Step 3: Add items to the shopping basket and check out.

Add items to the shopping basket and view the check out page. The form options in this example display in the default format with only the selected form fields. Required fields appear in bold.



Payment Information

● Credit Card

Type: Select a Credit Card

Number:

Expiration: /

Name:

Billing Information

Name:

Address:

City:

State:

Province:

Zip Code:

Country:

Phone:

Email:

Options 2 and 3: Custom Layouts and Forms

Steps

1. View the shopping basket administration page in a web browser.
2. Edit and Save the Customer Information Options.

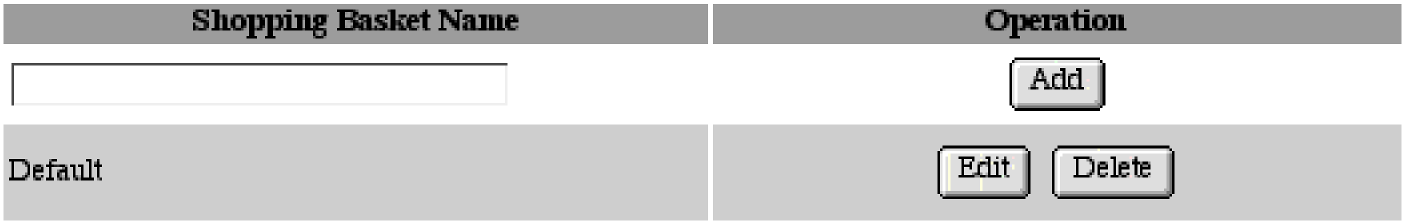
3. Open the check out page in a text editor.
4. Insert a custom form for payment, billing and shipping information.
5. Save the check out page.
6. Customize the order confirmation and order email, if necessary.
7. FTP all changed pages to the web server running MGI.
8. Add items to the shopping basket and check out.

Step 1: View the shopping basket administration page in a web browser.

View the shopping basket administration page (e.g., sbadmin.mgi) in a web browser. The first screen of the web-based, administration interface displays the current shopping basket configurations (handles)

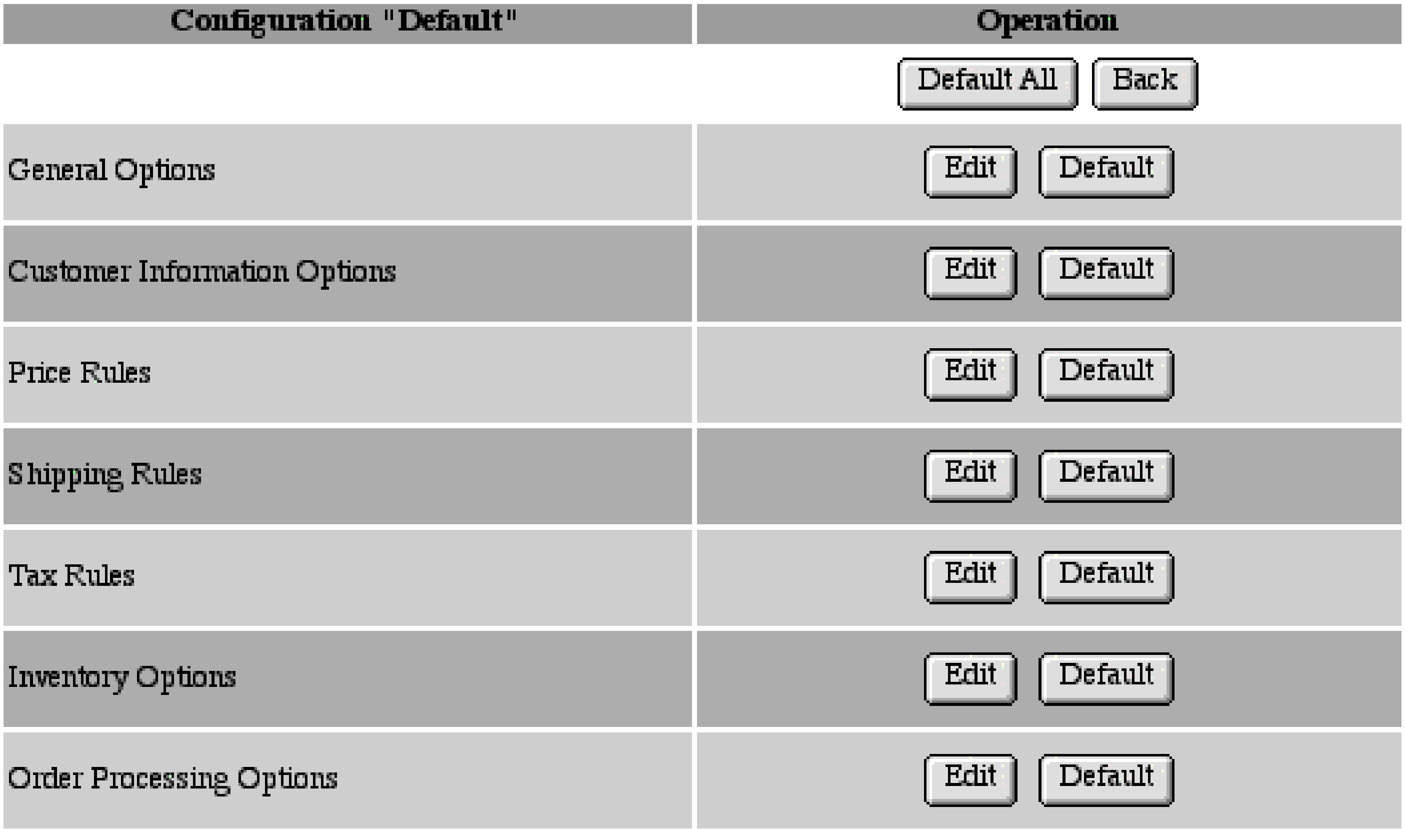
Step 2: Edit and Save the Customer Information Options.

Click the "Edit" button beside the shopping basket configuration (handle) to edit.



Shopping Basket Name	Operation
	Add
Default	Edit Delete

Click the "Edit" button beside "Customer Information Options".

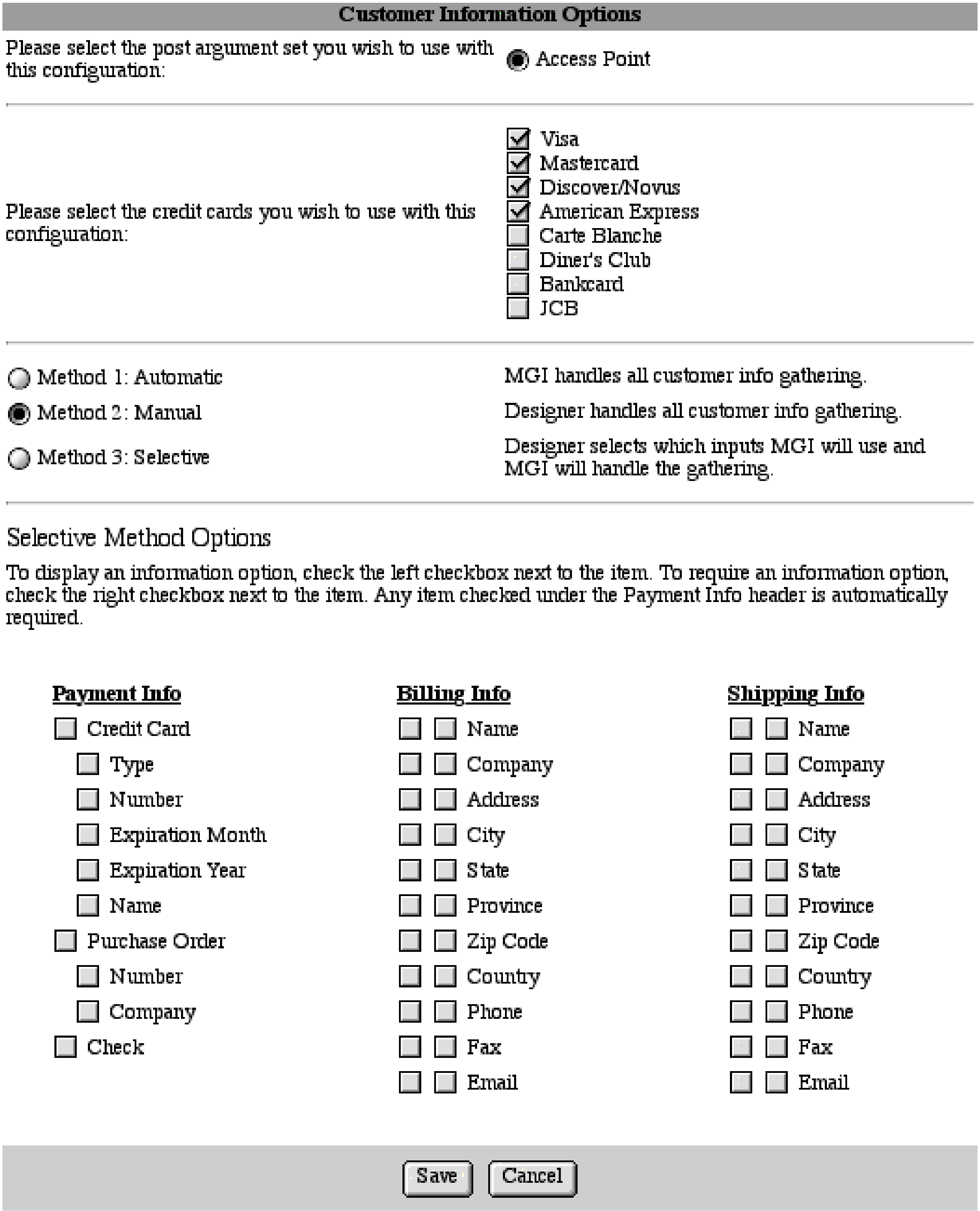


Configuration "Default"	Operation
	Default All Back
General Options	Edit Default
Customer Information Options	Edit Default
Price Rules	Edit Default
Shipping Rules	Edit Default
Tax Rules	Edit Default
Inventory Options	Edit Default
Order Processing Options	Edit Default

In the second section, check the boxes beside each type of credit card that you accept.

For option 2 (Automatic or Selective method with custom layout and default form fields), select either the Automatic or Selective method. For option 3 (Manual method with custom layout and custom or additional fields), select the Manual method. Option 3 is shown.

Click the "Save" button to save the changes to the shopping basket configuration. The main menu displays after the configuration is saved.



Customer Information Options

Please select the post argument set you wish to use with this configuration: ● Access Point

Please select the credit cards you wish to use with this configuration:

☑ Visa
☑ Mastercard
☑ Discover/Novus
☑ American Express
□ Carte Blanche
□ Diner's Club
□ Bankcard
□ JCB

○ Method 1: Automatic — MGI handles all customer info gathering.

● Method 2: Manual — Designer handles all customer info gathering.

○ Method 3: Selective — Designer selects which inputs MGI will use and MGI will handle the gathering.

Selective Method Options

To display an information option, check the left checkbox next to the item. To require an information option, check the right checkbox next to the item. Any item checked under the Payment Info header is automatically required.

Payment Info	**Billing Info**	**Shipping Info**
□ Credit Card	□ □ Name	□ □ Name
□ Type	□ □ Company	□ □ Company
□ Number	□ □ Address	□ □ Address
□ Expiration Month	□ □ City	□ □ City
□ Expiration Year	□ □ State	□ □ State
□ Name	□ □ Province	□ □ Province
□ Purchase Order	□ □ Zip Code	□ □ Zip Code
□ Number	□ □ Country	□ □ Country
□ Company	□ □ Phone	□ □ Phone
□ Check	□ □ Fax	□ □ Fax
	□ □ Email	□ □ Email

Step 3: Open the check out page in a text editor.

Open the check out page in a text editing program that allows you to view and modify the

HTML and code of the page.

Step 4: Insert a custom form for payment, billing and shipping information.

For option 2 (Automatic or Selective method with custom layout and default form fields) and option 3 (Manual method with custom layout and custom or additional fields) remove the mgiCollectUserInfo tag from the check out page and enter a custom layout to collect payment, billing, shipping and other information.

For option 2 (Automatic or Selective method with custom layout and default form fields) you may choose any text or layout, but you must use the following default field names and form elements. Using the default field names allows you to use the default order confirmation and order email pages if you prefer.

- **PaymentMethod** - Shopping basket payment method. Values include "Credit Card", "Purchase Order", and "Check".
- **CardType** - Credit card type. Values include "Visa", "Mastercard", "Disover/Novus", "American Express", "Carte Blanche", "Diner's Club", "Bankcard", and/or "JCB" depending on shopping basket configuration.
- **CCNumber** - Credit card number.
- **ExpMo** - Credit card expiration month.
- **ExpYe** - Credit card expiration year.
- **CCName** - Name that appears on credit card.
- **PONumber** - Purchase order number.
- **POCompany** - Purchase order company.
- **BName** - Billing name
- **BCompany** - Billing company
- **BAddress1** - Billing address
- **BCity** - Billing city
- **BState** - Billing state
- **BProvince** - Billing province
- **BZipCode** - Billing zip code
- **BCountry** - Billing country
- **BPhone** - Billing phone
- **BFax** - Billing fax
- **BEmail** - Billing email
- **SCompany** - Shipping company
- **SAddress1** - Shipping address
- **SCity** - Shipping city

- **SState -** Shipping state
- **SProvince -** Shipping province
- **SZipCode -** Shipping zip code
- **SCountry -** Shipping country
- **SPhone -** Shipping phone
- **SFax -** Shipping fax
- **SEmail -** Shipping email

For option 3 (Manual method with custom layout and custom or additional fields) you may choose any text, layout or form fields. You may also add additional form fields for information you wish to collect. Since you are using custom form fields, you are required to customize the order confirmation display and order email.

The following is a custom layout to collect payment, billing, shipping and additional information:

```
<mgitoken>

<FORM ACTION="confirmorder.mgi" METHOD="POST">

<P><CENTER>
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0"
WIDTH="525">
<TR HEIGHT="375">
<TD WIDTH="300" HEIGHT="375" VALIGN="TOP" ALIGN="CENTER">
<P>

<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="3"
WIDTH="250">
<TR HEIGHT="30">
<TD HEIGHT="30" COLSPAN="2"><B>BILLING ADDRESS</B></TD>
</TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">Name*</FONT></B>
</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BName" SIZE="27">
</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Company</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BCompany" SIZE="27">
</TD></TR>
<TR><TD WIDTH="75">
```

```
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">Address*</FONT></B>
</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BAddress1"
SIZE="27">
</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">City*</FONT></B>
</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BCity" SIZE="27">
</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">State*</FONT></B>
</TD> <TD WIDTH="175"><SELECT name="BSTATE">
<OPTION>
<OPTION>Alabama
<OPTION>Alaska
<OPTION>Arizona
<OPTION>Arkansas
<OPTION>California
<OPTION>Colorado
<OPTION>Connecticut
<OPTION>Delaware
<OPTION>District of Columbia
<OPTION>Florida
<OPTION>Georgia
<OPTION>Hawaii
<OPTION>Idaho
<OPTION>Illinois
<OPTION>Indiana
<OPTION>Iowa
<OPTION>Kansas
<OPTION>Kentucky
<OPTION>Louisiana
<OPTION>Maine
<OPTION>Maryland
<OPTION>Massachusetts
<OPTION>Michigan
<OPTION>Minnesota
<OPTION>Mississippi
<OPTION>Missouri
<OPTION>Montana
<OPTION>Nebraska
<OPTION>Nevada
<OPTION>New Hampshire
<OPTION>New Jersey
```

```
<OPTION>New Mexico
<OPTION>New York
<OPTION>North Carolina
<OPTION>North Dakota
<OPTION>Ohio
<OPTION>Oklahoma
<OPTION>Oregon
<OPTION>Pennsylvania
<OPTION>Rhode Island
<OPTION>South Carolina
<OPTION>South Dakota
<OPTION>Tennessee
<OPTION>Texas
<OPTION>Utah
<OPTION>Vermont
<OPTION>Virginia
<OPTION>Washington
<OPTION>West Virginia
<OPTION>Wisconsin
<OPTION>Wyoming
</SELECT></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Province</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BProvince"
SIZE="27">
</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">Zip Code*
</FONT></B>
</TD><TD WIDTH="175"><INPUT TYPE="text" NAME="BZipCode"
SIZE="27">
</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Country</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BCountry"
SIZE="27">
</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">Phone*
</FONT></B>
</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BPhone"
SIZE="27">
</TD></TR>
<TR><TD WIDTH="75">
```

```
<P ALIGN=RIGHT>Fax</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BFax"
SIZE="27">
</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">E-mail*
</FONT></B>
</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BEmail"
SIZE="27">
</TD></TR>
<TR><TD WIDTH="75"></TD>
<TD WIDTH="175"></TD>
</TR></TABLE></P>
<P><TABLE BORDER="0" CELLPADDING="0" CELLSPACING="3"
WIDTH="260">
<TR HEIGHT="25">
<TD HEIGHT="25" COLSPAN="2"><B>PAYMENT INFORMATION
</B></TD>
</TR><TR HEIGHT="20">
<TD HEIGHT="20" COLSPAN="2"><INPUT TYPE="radio"
VALUE="Credit Card" NAME="PaymentMethod">
<B>Credit Card</B>
</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Type</TD>
<TD WIDTH="176">
<SELECT NAME="CardType">
<OPTION VALUE="" SELECTED>Select a Card Type
<OPTION>Visa
<OPTION>MasterCard
<OPTION>Discover/Novus
</SELECT></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Number</TD>
<TD WIDTH="176"><INPUT TYPE="text" NAME="CCNumber"
SIZE="26">
</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Expiration</TD>
<TD WIDTH="176"><INPUT TYPE="text" NAME="EXPMO" SIZE="4"> /
<INPUT TYPE="text" NAME="EXPYE" SIZE="4"></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Name</TD>
<TD WIDTH="176"><INPUT TYPE="text" NAME="CCName"
```

```
SIZE="26"></TD></TR>
<TR HEIGHT="25">
<TD HEIGHT="25" COLSPAN="2"><INPUT TYPE="radio"
VALUE="Check" NAME="PaymentMethod">
<B>Check or Money Order</B>
</TD></TR>
<TR><TD COLSPAN="2">&nbsp;</TD></TR>
<TR><TD WIDTH="75"></TD>
<TD WIDTH="176"></TD></TR></TABLE>
</TD>
<TD WIDTH="300" HEIGHT="375" VALIGN="TOP"
ALIGN="CENTER">
<P><TABLE BORDER="0" CELLPADDING="0" CELLSPACING="3"
WIDTH="250">
<TR HEIGHT="30">
<TD HEIGHT="30" COLSPAN="2"><B>SHIPPING ADDRESS
</B>(if different)</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Name</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SName"
SIZE="27"></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Company</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SCompany"
SIZE="27"></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Address</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SAddress1"
SIZE="27"></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>City</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SCity"
SIZE="27"></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>State</TD>
<TD WIDTH="175"><SELECT name="SSTATE">
<OPTION>
<OPTION>Alabama
<OPTION>Alaska
<OPTION>Arizona
<OPTION>Arkansas
<OPTION>California
<OPTION>Colorado
<OPTION>Connecticut
<OPTION>Delaware
```

```
<OPTION>District of Columbia
<OPTION>Florida
<OPTION>Georgia
<OPTION>Hawaii
<OPTION>Idaho
<OPTION>Illinois
<OPTION>Indiana
<OPTION>Iowa
<OPTION>Kansas
<OPTION>Kentucky
<OPTION>Louisiana
<OPTION>Maine
<OPTION>Maryland
<OPTION>Massachusetts
<OPTION>Michigan
<OPTION>Minnesota
<OPTION>Mississippi
<OPTION>Missouri
<OPTION>Montana
<OPTION>Nebraska
<OPTION>Nevada
<OPTION>New Hampshire
<OPTION>New Jersey
<OPTION>New Mexico
<OPTION>New York
<OPTION>North Carolina
<OPTION>North Dakota
<OPTION>Ohio
<OPTION>Oklahoma
<OPTION>Oregon
<OPTION>Pennsylvania
<OPTION>Rhode Island
<OPTION>South Carolina
<OPTION>South Dakota
<OPTION>Tennessee
<OPTION>Texas
<OPTION>Utah
<OPTION>Vermont
<OPTION>Virginia
<OPTION>Washington
<OPTION>West Virginia
<OPTION>Wisconsin
<OPTION>Wyoming
</SELECT></TD> </TR>
<TR><TD WIDTH="75">
```

```
<P ALIGN=RIGHT>Province</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SProvince"
SIZE="27"></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Zip Code</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SZipCode"
SIZE="27"></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Country</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SCountry"
SIZE="27"></TD> </TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Phone</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SPhone"
SIZE="27"></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Fax</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SFax"
SIZE="27"></TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>E-mail</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SEmail"
SIZE="27"></TD></TR>
<TR><TD WIDTH="75"></TD>
<TD WIDTH="175"></TD></TR></TABLE></P>
<P><TABLE BORDER="0" CELLPADDING="0" CELLSPACING="3"
WIDTH="260">
<TR HEIGHT="25">
<TD HEIGHT="25" COLSPAN="2">
<B>ADDITIONAL INFORMATION</B>
</TD></TR>
<TR><TD WIDTH="75">
<P ALIGN=RIGHT>Comments</TD>
<TD WIDTH="176">
<TEXTAREA NAME="Comments" COLS="25" ROWS="4">
</TEXTAREA></TD></TR>
<TR><TD WIDTH="75"></TD>
<TD WIDTH="176"></TD></TR>
<TR><TD WIDTH="75"></TD>
<TD WIDTH="176"></TD></TR>
<TR><TD WIDTH="75"></TD>
<TD WIDTH="176"></TD></TR>
</TABLE></TD></TR>
<TR HEIGHT="40">
<TD HEIGHT="40" COLSPAN="2">
```

```
<P><CENTER><mgibutton value="Confirm Order">
</CENTER></TD></TR>
</TABLE></CENTER></P>

</FORM>

</mgitoken>
```

Step 5: Save the check out page.

Save the changes you have made to the check out page.

Step 6: Customize the order confirmation and order email, if necessary.

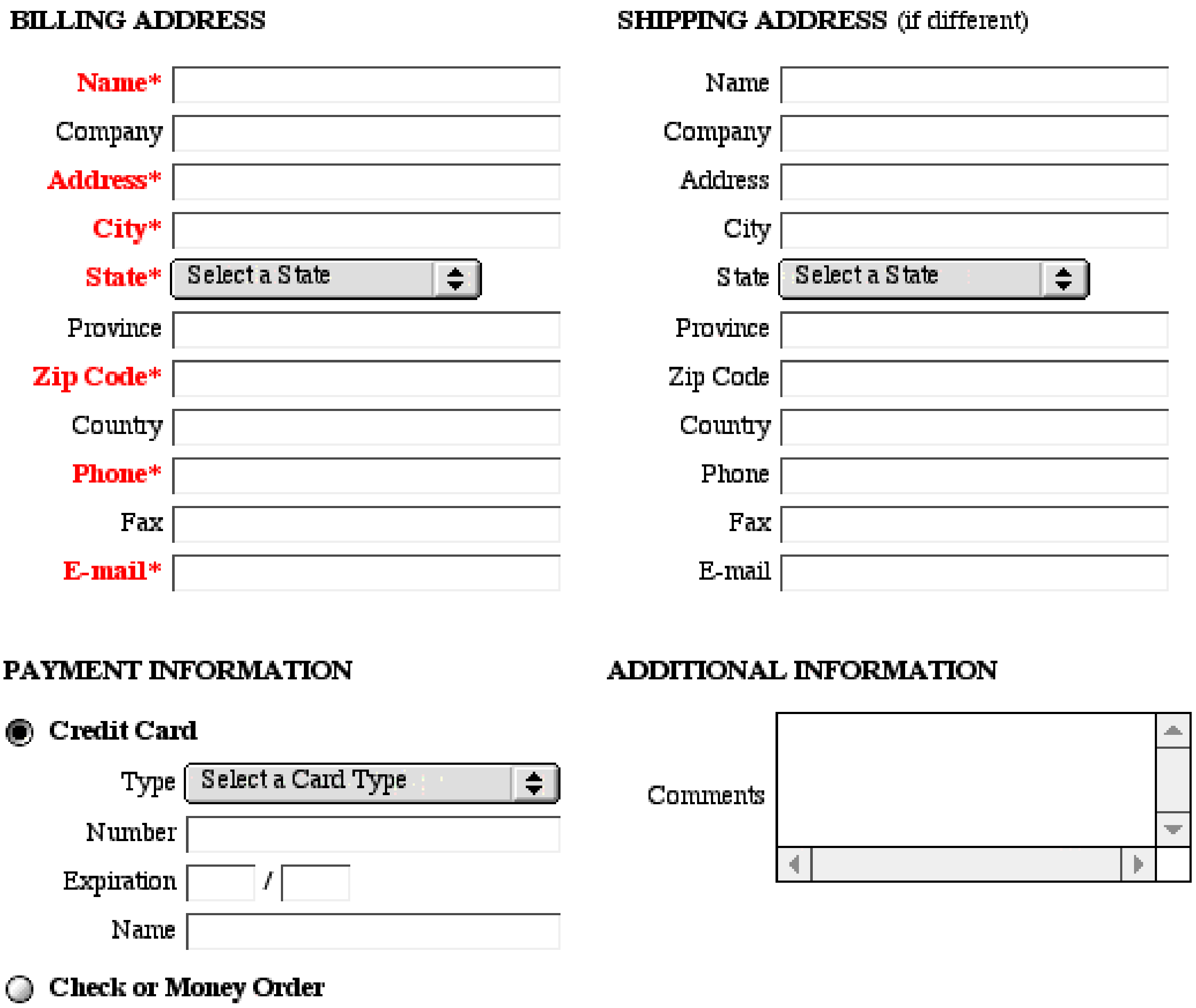
For option 3 (Manual method with custom layout and custom or additional fields) you must customize the order confirmation and order email. See the advanced tutorials for more information.

Step 7: FTP all changed pages to the web server running MGI.

Upload the check out page and order confirmation and order email pages (if changed) from your local computer to the web server using an FTP program.

Step 8: Add items to the shopping basket and check out.

Add items to the shopping basket and view the check out page. The following is the custom check out display from this example.



BILLING ADDRESS

Name*	
Company	
Address*	
City*	
State*	Select a State
Province	
Zip Code*	
Country	
Phone*	
Fax	
E-mail*	

SHIPPING ADDRESS (if different)

Name	
Company	
Address	
City	
State	Select a State
Province	
Zip Code	
Country	
Phone	
Fax	
E-mail	

PAYMENT INFORMATION

● **Credit Card**

Type Select a Card Type

Number

Expiration /

Name

○ **Check or Money Order**

ADDITIONAL INFORMATION

Comments

[Return to the Shopping Online Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Customize the Order Confirmation Display

Introduction

Please review the basic shopping online tutorial. This tutorial is an extension of the basic shopping online tutorial and explains how to customize one aspect of the shopping basket system. View the other advanced tutorials to customize other aspects of the shopping basket system.

The default shopping basket display of the mgiConfirmOrder tag includes the quantity, product ID, product name, product price, extended product price, subtotal, tax, shipping and total. The mgiConfirmorder tag also displays the payment, billing, and shipping information that was entered by the customer. You may customize the shopping basket layout and the content of the shopping basket using the mgiConfirmOrder placeholders and variables. You may use the default display of payment, shipping and billing information via the variables in mgiConfirmOrder or you may customize the display of that information with mgiPostArgument tags.

MGI Tags

- mgiConfirmOrder
- mgiGet
- mgiPostArgument

Steps

1. Open the confirm order page in a text editor.
2. Customize the shopping basket and customer information displays.
3. Save the confirm order page.
4. FTP the confirm order page to the web server running MGI.
5. Add items to the shopping basket, checkout and confirm the order.

Step 1: Open the confirm order page in a text editor.

Open the confirm order page (e.g., confirmorder.mgi) in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Customize the shopping basket and customer information displays.

If the default shopping basket does not fit with your site design, you may customize the shopping basket layout and the content of the shopping basket using the mgiConfirmOrder placeholders and variables. With a custom layout you can add font properties, cell colors, images, etc.

In a custom shopping basket, the code in the body of the mgiConfirmOrder tags is repeated for each item in the shopping basket and the placeholders are replaced with the item's specific information. After the mgiConfirmOrder tags, the shopping basket variables display information calculated from the current shopping basket contents.

You may format the text and HTML of your shopping basket in any way. The following placeholders are available to display information for each item. The placeholders will be replaced with each product's specific information.

- **&mgiDBFieldQuantity;** is the quantity of the product.
- **&mgiDBFieldQuantityMultiplier;** is the quantity multiplier of the product.
- **&mgiDBFieldProductID;** is the product ID.
- **&mgiDBFieldName;** is the product name (short description).
- **&mgiDBFieldDescription;** is the product description
- **&mgiDBFieldPrice;** is the product price.
- **&mgiDBFieldShipping;** is the product shipping cost.
- **&mgiDBFieldQualifier1;** is the first product qualifier.
- **&mgiDBFieldQualifier2;** is the second product qualifier.
- **&mgiDBFieldQualifier3;** is the third product qualifier.
- **&mgiDBFieldQualifier4;** is the fourth product qualifier.
- **&mgiDBFieldQualifier5;** is the fifth product qualifier.
- **&mgiDBFieldWeight;** is the product weight.
- **&mgiSBItemPriceTotal;** is the price of a product multiplied by the quantity purchased and includes any price rules that have been applied.
- **&mgiSBItemShippingTotal;** is the shipping price of a product after shipping rules have been applied.
- **&mgiSBItemTotal;** is the total price of a product (price subtotal plus shipping

subtotal) after price and shipping rules have been applied.

Use the shopping basket variables to display order totals and the default payment, billing and shipping tables or create your own displays of the payment, billing and shipping information with text, HTML and mgiPostArgument tags. If you use the mgiCollectUserInfo tag to collect payment, billing, and shipping information, then you can display that information by using the default post arguments. However, if you created a custom form to collect payment, billing and shipping information, then use your post argument names to display that information for confirmation.

The following variables are available anywhere after the ending mgiConfirmOrder tag. Variables can be displayed with the mgiGet tag.

- **mgiSBSubtotal** is the price of the order after the price rule(s) have been applied.
- **mgiSBTax** is the tax on the order after the tax rule(s) have been applied.
- **mgiSBShippingTotal** is the shipping price of the order after the shipping rule(s) have been applied.
- **mgiSBTotal** is the total price of the order after all rules have been applied.
- **mgiSBPaymentInfo** is a table of the customer's payment information. The mgiSBPaymentInfo variable is only available in Automatic and Selective modes of the shopping basket.
- **mgiSBBillingInfo** is a table of the customer's billing information. The mgiSBBillingInfo variable is only available in Automatic and Selective modes of the shopping basket.
- **mgiSBShippingInfo** is a table of the customer's shipping information. The mgiSBShippingInfo variable is only available in Automatic and Selective modes of the shopping basket.

Customizing the display of the confirm order shopping basket itself does not require you to customize any other aspect of the shopping basket system The confirm order shopping basket may be customized independently of the shopping basket page, check out page, and order email.

The following is the code of a custom order confirmation:

```
<mgiToken>

<form action="completeorder.mgi" method="post">

<p><table cellpadding="5" cellspacing="1" border="0">
<tr bgcolor="#EEEECC">
```

```
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Qty</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Description</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Price Ea.</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Item Total</b></font></td>
</tr>

<mgiConfirmOrder handle="Default"
shoppingBasketURL="http://www.domain.com/">

<tr>
<td>&mgiDBFieldQuantity;</td>
<td><font size="-1" face="helvetica, arial, sans-serif">
&mgiDBFieldName;</font></td>
<td align="center">
<font size="-1" face="helvetica, arial, sans-serif">
$&mgiDBFieldPrice;</font></td>
<td align="center">
<font size="-1" face="helvetica, arial, sans-serif">
$&mgiSBItemPriceTotal;</font></td>
</tr>

</mgiConfirmOrder>

<tr bgcolor="#EEEECC">
<td colspan="3" align="right">
<font size="-1" face="helvetica, arial, sans-serif">
<b>Subtotal</b></font></td>
<td align="center">
<font size="-1" face="helvetica, arial, sans-serif">
<b>$<mgiGet name="mgiSBSubtotal"></b></font></td>
</tr>

<tr bgcolor="#EEEECC">
<td colspan="3" align="right">
<font size="-1" face="helvetica, arial, sans-serif">
<b>Tax</b></font></td>
<td align="center">
<font size="-1" face="helvetica, arial, sans-serif">
<b>$<mgiGet name="mgiSBTax"></b></font></td>
</tr>
```

```
<tr bgcolor="#EEEECC">
<td colspan="3" align="right">
<font size="+1" face="helvetica, arial, sans-serif">
<b>Total</b></font></td>
<td align="center">
<font size="+1" face="helvetica, arial, sans-serif">
<b>$<mgiGet name="mgiSBTotal"></b></font></td>
</tr>

</table>

<p><mgiGet name="mgiSBPaymentInfo">
<p><mgiGet name="mgiSBBillingInfo">
<p><mgiGet name="mgiSBShippingInfo">

<p><mgiButton name="Complete Order">

</form>

</mgiToken>
```

Step 3: Save the confirm order page.

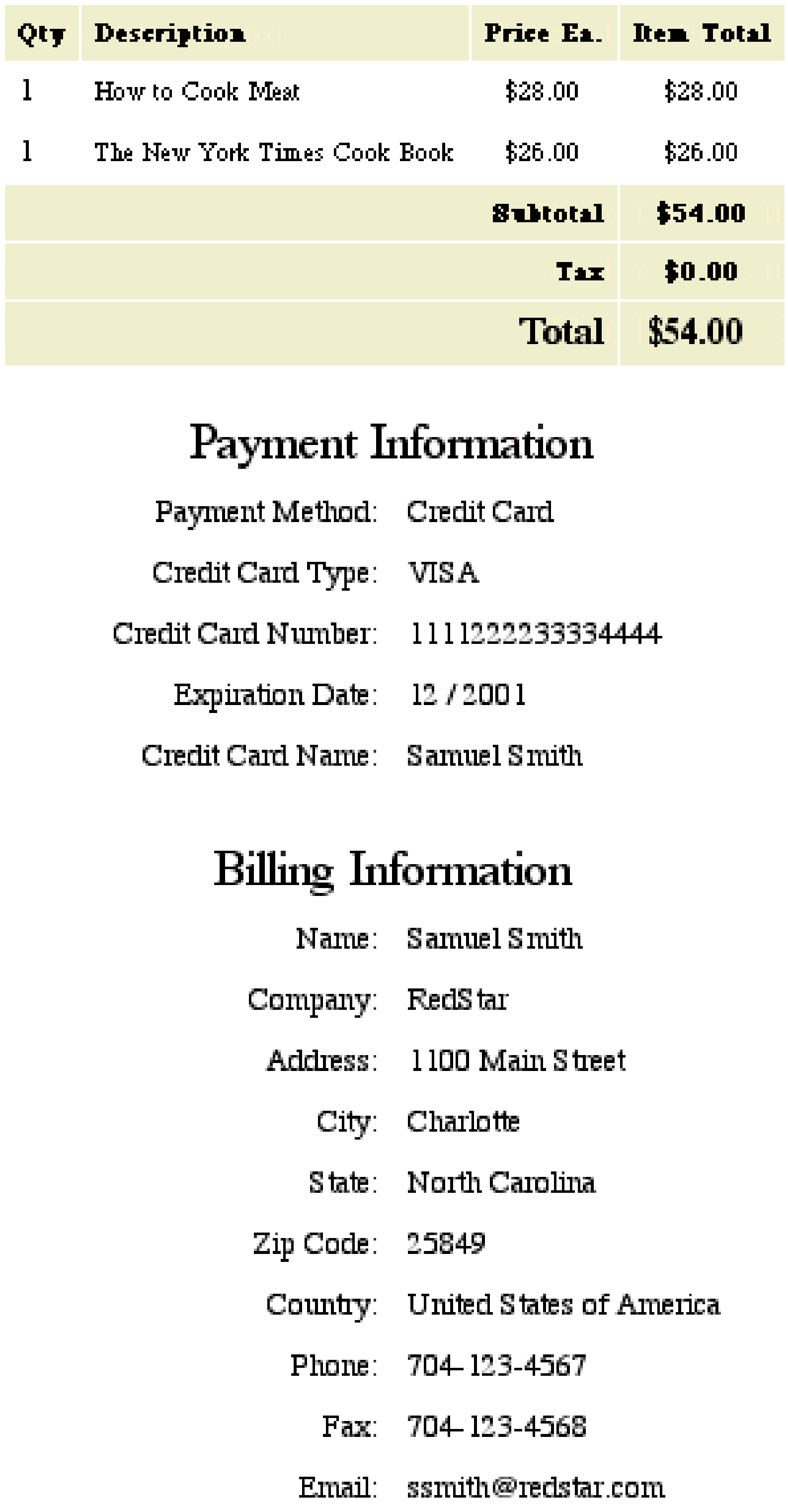
Save the changes you have made to the confirm order page.

Step 4: FTP the confirm order page to the web server running MGI.

Upload the confirm order page from your local computer to the web server using an FTP program.

Step 5: Add items to the shopping basket, checkout and confirm the order.

Add items to your shopping basket, complete the check out page and proceed to the confirm order page to display the confirmation of your order. The custom order confirmation in this example has the following display:



Qty	Description	Price Ea.	Item Total
1	How to Cook Meat	$28.00	$28.00
1	The New York Times Cook Book	$26.00	$26.00
		Subtotal	$54.00
		Tax	$0.00
		Total	$54.00

Payment Information

Payment Method:	Credit Card
Credit Card Type:	VISA
Credit Card Number:	1111222233334444
Expiration Date:	12 / 2001
Credit Card Name:	Samuel Smith

Billing Information

Name:	Samuel Smith
Company:	RedStar
Address:	1100 Main Street
City:	Charlotte
State:	North Carolina
Zip Code:	25849
Country:	United States of America
Phone:	704-123-4567
Fax:	704-123-4568
Email:	ssmith@redstar.com

[Return to the Shopping Online Menu]

Customize the Order Email

Introduction

Please review the basic shopping online tutorial. This tutorial is an extension of the basic shopping online tutorial and explains how to customize one aspect of the shopping basket system. View the other advanced tutorials to customize other aspects of the shopping basket system.

The default order email displays formatted payment, billing, shipping and order information aligned by colons. If you prefer a different layout or require a specific format for an order processing program, for example, use the mgiSendOrder placeholders and mgiPostArgument tags to customize the order email. Please note that the mgiSendOrder placeholders are not "post-processed" which means that placeholder values cannot be used in math calculations, etc. Those values are simply replaced with the customer's values and order information.

MGI Tags

- mgiPostArgument
- mgiSendOrder

Steps

1. Open the order processing page in a text editor.
2. Customize the order email.
3. Save the order processing page.
4. FTP the order processing page to the web server running MGI.
5. Add items to the shopping basket, check out, confirm and process the order.

Step 1: Open the order processing page in a text editor.

Open the order processing page (e.g., completeorder.mgi) in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Customize the order email.

If you prefer a different email format, use the mgiSendOrder placeholders and mgiPostArgument tags to customize the email layout. Placeholders between the "Begin Template" and "End Template" comments will be repeated for each item ordered and replaced with the item's specific information. If you use the mgiCollectUserInfo tag to collect payment, billing, and shipping information, then you can display that information in the custom order email by using the default post arguments. However, if you created a custom form to collect payment, billing and shipping information, then use your post argument names to display that information in the custom order email. Custom order emails are sent as text, therefore you may use spaces to align information.

The **mgiSendOrder email is not "post-processed"** which means that you cannot perform MGI functions such as math calculations with placeholders in the body of the mgiSendOrder tags. If you need to perform such functions with placeholder information, we suggest you perform the function in the mgiConfirmOrder tag and use a hidden post argument to transfer the result to the mgiSendOrder page. The custom order email in this example has the following format:

You may format the order email using the following placeholders for item and order total information. The placeholders will be replaced with each product's specific information or total.

- **&mgiDBFieldQuantity;** is the quantity of the product.
- **&mgiDBFieldQuantityMultiplier;** is the quantity multiplier of the product.
- **&mgiDBFieldProductID;** is the product ID.
- **&mgiDBFieldName;** is the product name (short description).
- **&mgiDBFieldDescription;** is the product description
- **&mgiDBFieldPrice;** is the product price.
- **&mgiDBFieldShipping;** is the product shipping cost.
- **&mgiDBFieldQualifier1;** is the first product qualifier.
- **&mgiDBFieldQualifier2;** is the second product qualifier.
- **&mgiDBFieldQualifier3;** is the third product qualifier.
- **&mgiDBFieldQualifier4;** is the fourth product qualifier.
- **&mgiDBFieldQualifier5;** is the fifth product qualifier.
- **&mgiDBFieldWeight;** is the product weight.
- **&mgiSBItemPriceTotal;** is the price of a product multiplied by the quantity purchased and includes any price rules that have been applied.
- **&mgiSBItemShippingTotal;** is the shipping price of a product after shipping rules have been applied.
- **&mgiSBItemTotal;** is the total price of a product (price subtotal plus shipping subtotal) after price and shipping rules have been applied.
- **&mgiSBSubtotal;** is the price of the order after the price rule(s) have been applied.
- **&mgiSBTax;** is the tax on the order after the tax rule(s) have been applied.

- **&mgiSBShippingTotal;** is the shipping price of the order after the shipping rule(s) have been applied.
- **&mgiSBTotal;** is the total price of the order after all rules have been applied.

You may include information using other MGI tags in the custom order email. In this example the token value and current date are displayed in the order email.

Customizing the display of the order processing page itself does not require you to customize any other aspect of the shopping basket system. The order processing page may be customized independently of the shopping basket page, check out page, and order confirmation page.

The following is the code of a custom order email:

```
<mgiSendOrder handle="Default"
shoppingBasketURL="http://www.domain.com/"
to="orders@domain.com" from="webmaster@domain.com"
mailServer="mail.domain.com"
subject="Test MGI2 SB Order">

Payment
-------
       Type: <mgiPostArgument name="CardType">
     Number: <mgiPostArgument name="CCNumber">
        Exp: <mgiPostArgument name="EXPMO">
<mgiPostArgument name="EXPYE">

Billing
-------
       Name: <mgiPostArgument name="bName">
    Company: <mgiPostArgument name="bCompany">
    Address: <mgiPostArgument name="bAddress1">
             <mgiPostArgument name="bCity">
<mgiPostArgument name="bState">
<mgiPostArgument name="bZipCode">
             <mgiPostArgument name="bCountry">
      Phone: <mgiPostArgument name="bPhone">
        Fax: <mgiPostArgument name="bFax">
      Email: <mgiPostArgument name="bEmail">

Order
-----
  Order Ref: <mgiPathArgument name="mgiToken">
 Order Date: <mgiDate>
```

```
<!-- Begin Template -->

 Product ID: &mgiDBFieldProductID;
        Qty: &mgiDBFieldQuantity;
    Product: &mgiDBFieldName;
 Price Each: $&mgiDBFieldPrice;
 Item Total: $&mgiSBItemPriceTotal;
<!-- End Template -->

        Tax: $&mgiSBTax;
      Total: $&mgiSBTotal;
</mgiSendOrder>
```

Step 3: Save the order processing page.

Save the changes you have made to the order processing page.

Step 4: FTP the order processing page to the web server running MGI.

Upload the order processing page from your local computer to the web server using an FTP program.

Step 5: Add items to the shopping basket, check out, confirm and process the order.

Add items to your shopping basket, complete the check out page, confirm your order and process the order. The custom order email in this example has the following display:

```
Payment
-------
       Type: VISA
     Number: 1111222233334444
        Exp: 12 2001

Billing
-------
       Name: Samuel Smith
    Company: RedStar
    Address: 1100 Main Street
             Charlotte North Carolina 25849
             United States of America
      Phone: 704-123-4567
        Fax: 704-123-4568
      Email: ssmith@redstar.com
```

```
Order
-----
  Order Ref: 36Q03NHIB14GT5
 Order Date: Tuesday, August 21, 2001

 Product ID: 0-688-16199-5
        Qty: 1
    Product: How to Cook Meat
 Price Each: $28.00
 Item Total: $28.00

 Product ID: 0-060-16010-1
        Qty: 1
    Product: The New York Times Cook Book
 Price Each: $26.00
 Item Total: $26.00

        Tax: $0.00
      Total: $54.00
```

[Return to the Shopping Online Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Encrypting Orders

Introduction

Please review the basic shopping online tutorial. This tutorial is an extension of the basic shopping online tutorial and explains how to encrypt shopping basket orders.

By default shopping basket orders are sent as plain text emails. To secure the order emails, encrypt them with a passphrase. The email recipient can use encryption software in their email browser or as a separate application to decrypt the order emails.

MGI Tags

- mgiShoppingBasket

Steps

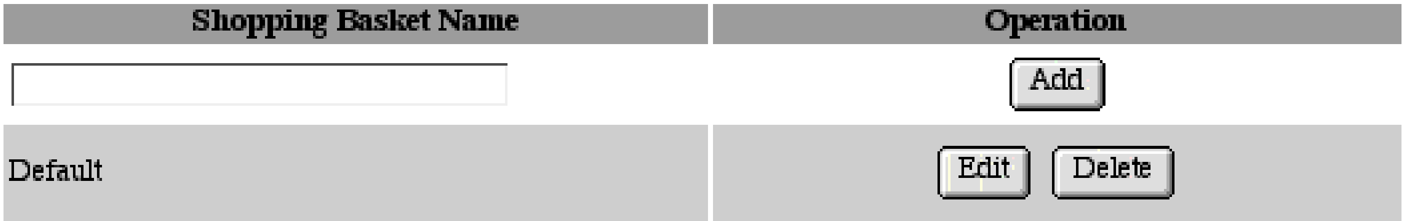
1. View the shopping basket administration page in a web browser.
2. Edit and Save the Order Processing Options.
3. Process an order.
4. Decrypt the order email.

Step 1: View the shopping basket administration page in a web browser.

View the shopping basket administration page (e.g., sbadmin.mgi) in a web browser. The first screen of the web-based, administration interface displays the current shopping basket configurations (handles)

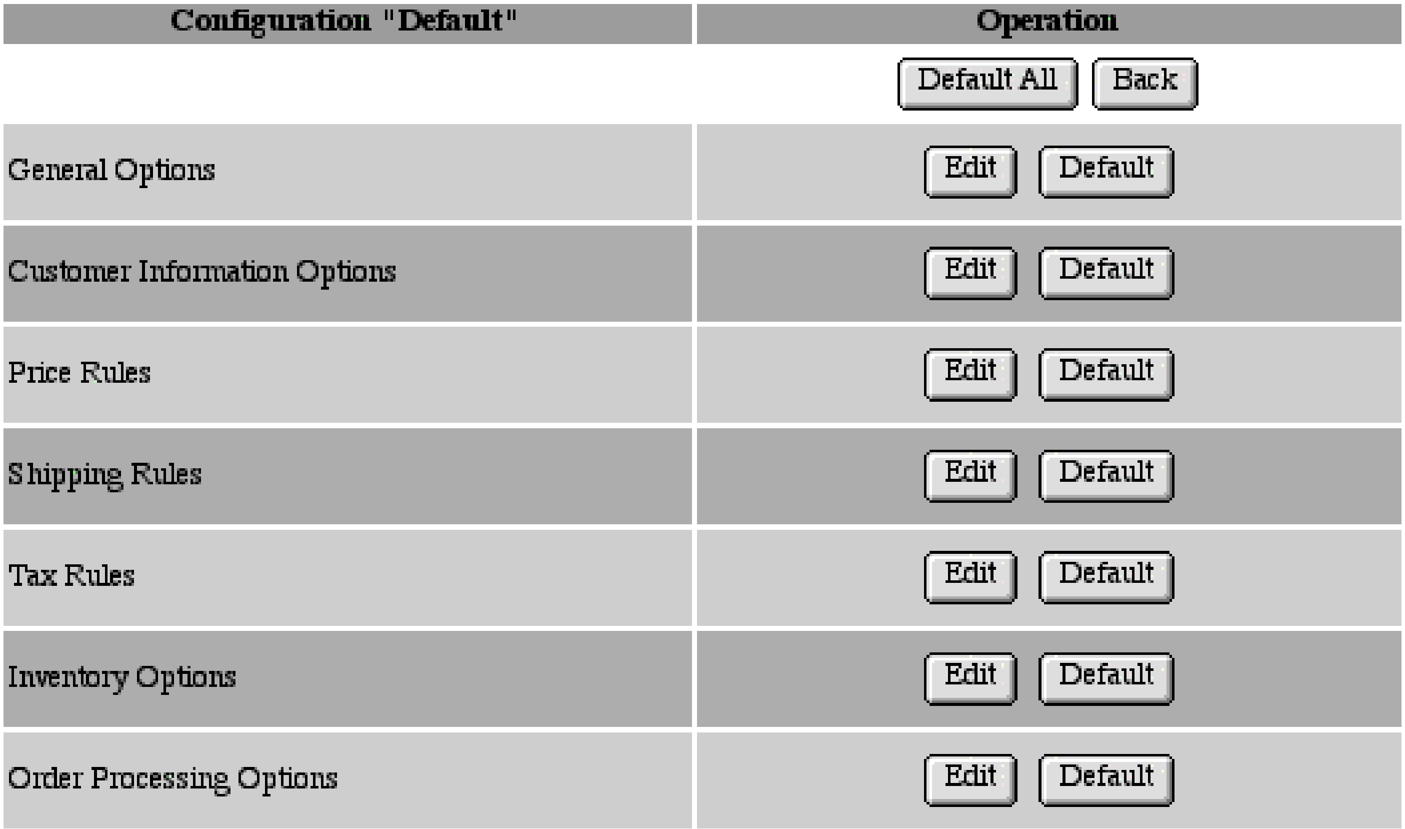
Step 2: Edit and Save the Order Processing Options.

Click the "Edit" button beside the shopping basket configuration (handle) to edit.



Shopping Basket Name	Operation
	Add
Default	Edit Delete

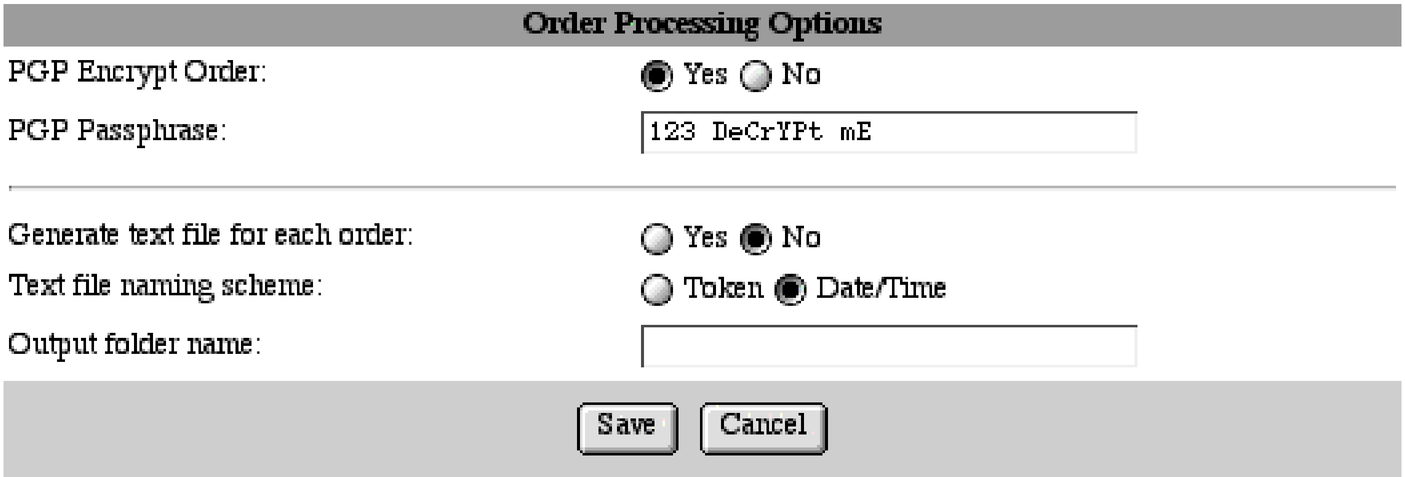
Click the "Edit" button beside "Order Processing Options".



Configuration "Default"	Operation
	Default All Back
General Options	Edit Default
Customer Information Options	Edit Default
Price Rules	Edit Default
Shipping Rules	Edit Default
Tax Rules	Edit Default
Inventory Options	Edit Default
Order Processing Options	Edit Default

Beside "PGP Encrypt Order" click the "Yes" radio button. Beside "PGP Passphrase" enter a case-sensitive phrase to use for encryption. The phrase may contain any characters including spaces. The passphrase you enter in the Order Processing Options will be used by the order recipient to decrypt order emails.

Click the "Save" button to save the changes to the shopping basket configuration. The main menu displays after the configuration is saved.



Order Processing Options

PGP Encrypt Order:	● Yes ○ No
PGP Passphrase:	123 DeCrYPt mE
Generate text file for each order:	○ Yes ● No
Text file naming scheme:	○ Token ● Date/Time
Output folder name:	

Step 3: Process an order.

Process an order from your shopping basket. The order wil be encrypted and emailed to the recipient specified in the mgiSendOrder tag.

Step 4: Decrypt the order email.

When the encrypted order is received, it will have the following format:

```
-----BEGIN PGP MESSAGE-----

qANQR1DDDAQDAQLULCLaVkaRJaUCEqLsvLwkR2A3+R3wkyhPrY2st0y
2whsK2EmXLPTel65o8Bb95Q19e1+sMUYbC2h/xcPgMVTxAtMTZf+q+3
y/obdAjhYDmJqsBhlgsfN7n1XlMTUgMeVjxLKYb0cCXMJM6S/AAFkte
MZJkAFPKSZrTzOrv0PWjc9Me65a1een8f/vi0kV2xBHQSVGJZZXYPA1
zafCbm2Glmswke0Y3GWHV8/dxZSCpPUu/Bvjp7/4ZYlcMzSVwuadwND
0NrpKJ+Tz9gw04WOCLNHkBGyCHOaCVAEAB/mDLDBp8X05uQHCKYqh0C
YxVSdNfR+hCF/4EUzoDu0whbLGK2fHtNV0FgII2lBAI9EdKOCF6N2bf
kpecLS1V7UXElCL2hfD+WZ6LCO4kc2vUihkxh1YbhH/agkX7ik+g6Fb
lD3Ar1CkiWRrAUTbm0bQBu7zLiUK6yT0oTBs0Y/0EyFziMeEYElsQNp
2DWXay/G7Rd1q9YGrGxMIriJoJCfsuILBUQh7k0uTHEOAJxHv43D
=M8i4
-----END PGP MESSAGE-----
```

The spacing of the email is important and should be maintained. To decrypt the email follow the instructions for your decryption application. You may be required to copy and paste the encrypted text into the application or a separate text document. Free PGP utilities for personal use can be found at http://web.mit.edu/network/pgp.html. Commercial PGP utilities can be found at http://www.pgp.com.

[Return to the Shopping Online Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Using Inventory Control

Introduction

Please review the database and basic shopping online tutorial. This tutorial is an extension of the basic shopping online tutorial and explains how to implement inventory control with the shopping basket system.

Without inventory control, customers may purchase any quantity of any item that you offer for sale. If you prefer to limit purchases to only available quantities (as listed in your product database),you may choose to implement inventory controls with your shopping basket system. Inventory control may only be used with database-driven shopping baskets.

The database name, product ID field, and inventory field is stored with each item that is added to a shopping basket. During the final order processing using mgiSendOrder, the ordered product quantities are compared with the available quantities in your product database. If a customer attempts to order a product quantity that is not available, the customer is redirected to the inventory error page that is listed in your shopping basket administration. The specific products that have caused the inventory error are sent sequentially via path arguments to the inventory error page where you may present several courses of action including a shopping basket where the customer can update their order.

MGI Tags

- mgiBuyMe
- mgiShoppingBasket

Steps

1. Create a product database with an inventory field.
2. Open the product result pages in a text editor.
3. Insert the inventory parameters in the mgiBuyMe tags.
4. Save the product result pages.
5. Create an inventory error page.
6. Save the inventory error page.
7. FTP the product pages and inventory error page to the web server running MGI.
8. View the shopping basket administration page in a web browser.
9. Edit and Save the Inventory Options.
10. Process an order.

Step 1: Create a product database with an inventory field.

Inventory control may only be used with a database-driven shopping basket system. In addition to the product ID, product name and price, create a field for the current inventory. The field type must be either Whole Number (Integer) or Positive Number (Unsigned Integer).

Access the record interface of the database administration. Enter the current inventory for each product in the inventory field. Inventory figures can be updated at any time in the database administration interface.

Step 2: Open the product result pages in a text editor.

Open the database result pages that contain mgiBuyMe tags for your products in a text editing program that allows you to view and modify the HTML and code of the page.

Step 3: Insert the inventory parameters in the mgiBuyMe tags.

Add the productDB, inventoryFieldName, and productIDFieldName parameters to all mgiBuyMe tags. In the productDB parameter, enter the case-sensitive database name that contains the product's inventory information. In the inventoryFieldName parameter, enter the case-sensitive database field name that contains the product's current inventory. In the productIDFieldName parameter, enter the case-sensitive database field name that contains the product's unique ID.

The following is an example mgiSearchDatabase tag with inventory parameters.

```
<mgiToken>

<form action="shoppingbasket.mgi" method="post">

<h2>Search Results</h2>
<p>
<table width="500" cellspacing="0"
cellpadding="3" border="1">
<tr bgcolor="#eeeeee">
  <th>ISBN</th>
  <th>Title/Description</th>
  <th>Price</th>
  <th>Quantity</th>
</tr>

<mgiSearchDatabase databaseName="Products"
keyFieldName="ProductName" fieldValue="*"
```

```
orderByField="ProductID" resultsPerPage="5">

<tr>
  <td>&mgiDBFieldProductID;</td>
  <td><b>&mgiDBFieldProductName;</b>
  <p>&mgiDBFieldProductDescription;</td>
  <td align="right">$&mgiDBFieldProductPrice;</td>
  <td align="center">
<mgiBuyMe productID="&mgiDBFieldProductID;"
name="&mgiDBFieldProductName;"
price="&mgiDBFieldProductPrice;"
productDB="Products" productIDFieldName="ProductID"
inventoryFieldName="Inventory"></td>
</tr>

</mgiSearchDatabase>

<tr>
  <td colspan="4" align="right">
<mgiButton value="Add to Shopping Basket"></td>
</tr>
</table>
</p>

</form>

</mgiToken>
```

Step 4: Save the product result pages.

Save the changes you have made to the product result pages.

Step 5: Create an inventory error page.

Create a page to display when an inventory error occurs and open the page in a text editing program that allows you to view and modify the HTML and code of the page.

During the shopping basket process, customers add items to their shopping basket, checkout, confirm their order and finally submit their order. When a customer submits their order, the quantity of each item they purchase is compared with the available quantity as it appears in the inventory field of the specified database.

If the quantity ordered for all items is less than the quantity available, the quantity ordered is debited from the quantity available in the specified database and the order is processed.

If the quantity ordered for any item is greater than the quantity available, the customer is redirected to the error URL specified in the Inventory Options of the the shopping basket admin (see steps 8 and 9 for more information about the Inventory Options). The error URL is appended with sequential product IDs, quantities ordered and quantities available path arguments for each item with an inventory error.

You can use the mgiPathArgument tag and mgiLoop tag to display the product ID, quantity ordered and quantity available for each item with an inventory error on the error page. You can also offer the customer options for correcting the inventory error such as a shopping basket tag which allows them to update their quantities. If you offer a shopping basket for modifications, the inventory error page should be on your non-secure server.

The following is an example inventory error page.

```
<mgiToken>

<font size="+2">Inventory Errors</font>

<p>The following items in your purchase are not
currently available in the quantity ordered.
Please update the quantity in the shopping basket
below and re-submit your order.

<p>
<table border="1" cellspacing="0" cellpadding="3">
<tr>
<th>Product ID</th>
<th>Quantity Ordered</th>
<th>Quantity In Stock</th>
</tr>

<mgiloop first="1" last="10">
<mgiif
lhs={mgipathargument name="productID&mgiLoopIndex;"}
relationship="isnotempty">
<tr>
<td>
<mgipathargument name="productID&mgiLoopIndex;">
</td>
<td align="center">
<mgipathargument name="quantity&mgiLoopIndex;">
```

```
</td>
<td align="center">
<mgipathargument name="inventory&mgiLoopIndex;">
</td>
</tr>
</mgiif>
</mgiloop>

</table>

<form action="shoppingbasket.mgi" method="post">

<p><mgiShoppingBasket handle="Default">
</mgiShoppingBasket>

<p><mgiButton value="Modify Quantity">

</form>

<form action="checkout.mgi" method="post">
<mgiButton value="Check Out">
</form>

</mgiToken>
```

Step 6: Save the inventory error page.

Save the inventory error page and name it "inverror.mgi".

Step 7: FTP the product pages and inventory error page to the web server running MGI.

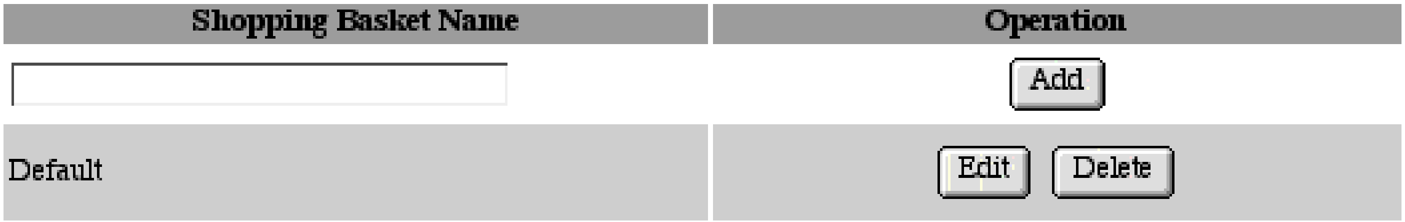
Upload the product pages and inventory error page (inverror.mgi) from your local computer to the web server using an FTP program.

Step 8: View the shopping basket administration page in a web browser.

View the shopping basket administration page (e.g., sbadmin.mgi) in a web browser. The first screen of the web-based, administration interface displays the current shopping basket configurations (handles)

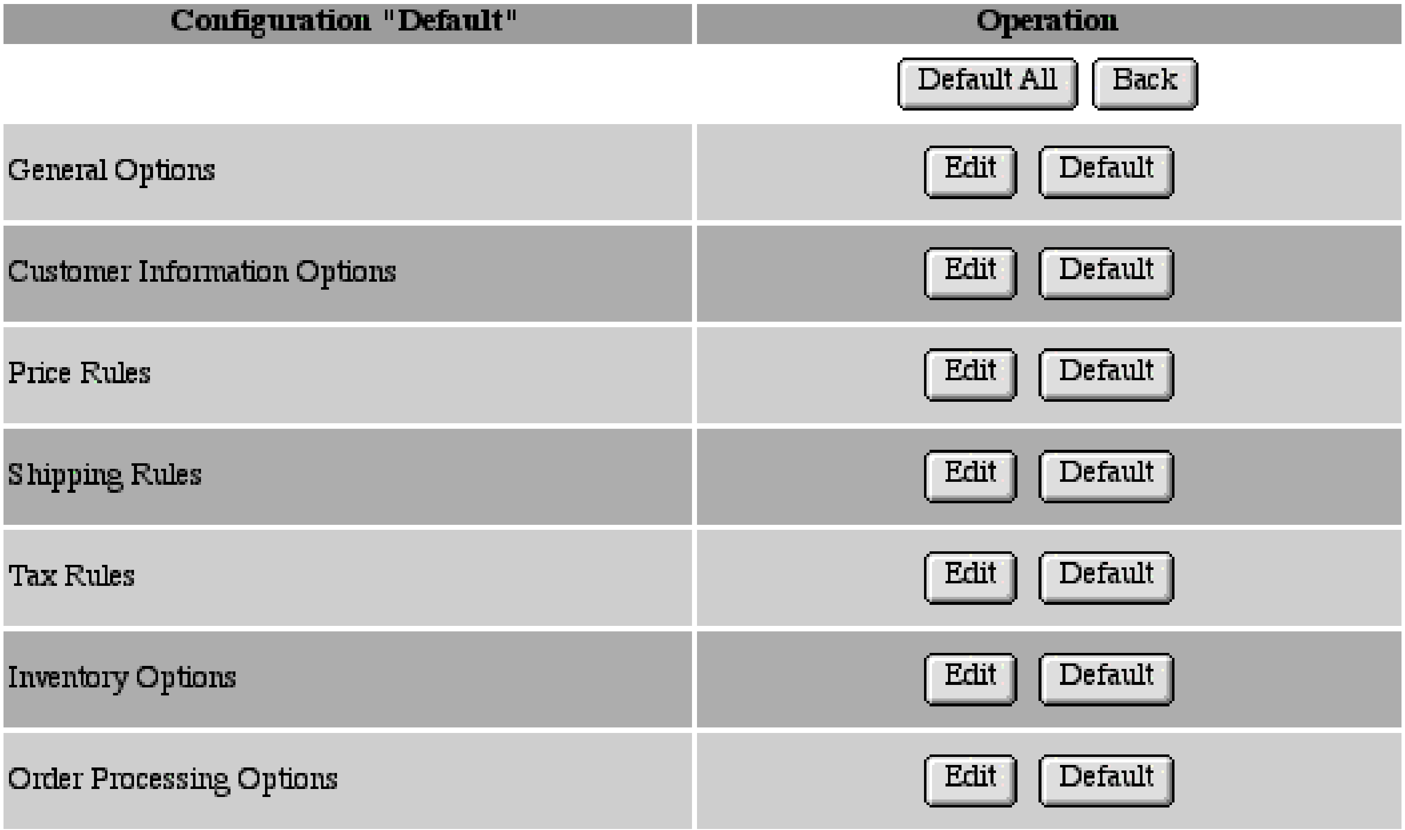
Step 9: Edit and Save the Inventory Options.

Click the "Edit" button beside the shopping basket configuration (handle) to edit.



Shopping Basket Name	Operation
	Add
Default	Edit Delete

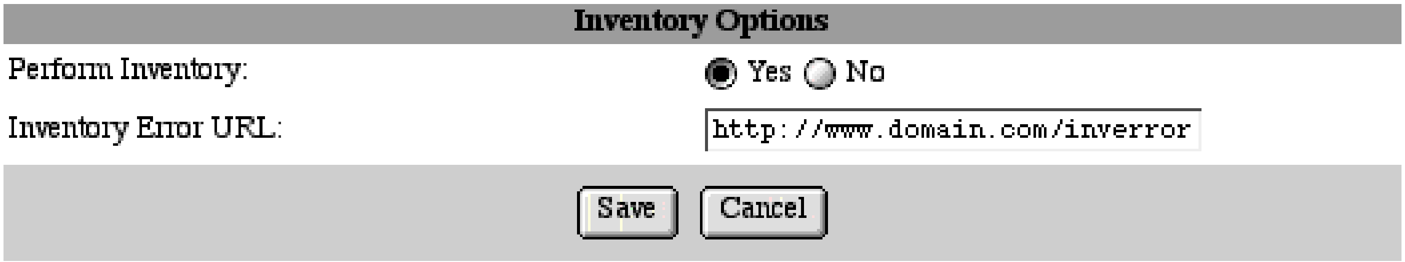
Click the "Edit" button beside "Inventory Options".



Configuration "Default"	Operation
	Default All Back
General Options	Edit Default
Customer Information Options	Edit Default
Price Rules	Edit Default
Shipping Rules	Edit Default
Tax Rules	Edit Default
Inventory Options	Edit Default
Order Processing Options	Edit Default

Beside "Perform Inventory" click the "Yes" radio button. Beside the "Inventory Error URL" enter the full absolute URL to the inventory error page (e.g., inverror.mgi).

Click the "Save" button to save the changes to the shopping basket configuration. The main menu displays after the configuration is saved.



Inventory Options	
Perform Inventory:	● Yes ○ No
Inventory Error URL:	http://www.domain.com/inverror
Save Cancel	

Step 10: Process an order.

Process an order from your shopping basket. If the quantity ordered for all products is less than the quantity available, the quantity ordered is debited from the quantity available and the order is processed. If the quantity ordered for any product is greater than the quantity available, you are redirected to the inventory error page.

[Return to the Shopping Online Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Authorizing Credit Cards via a Gateway

Introduction

Please review the basic shopping online tutorial. This tutorial is an extension of the basic shopping online tutorial and explains how to integrate a credit card gateway into the shopping basket system.

Credit card payments in a default shopping basket are not authorized or charged during the shopping basket process. Rather, the payment information is collected and the order recipient is responsible for authorizing the credit card via a terminal, online terminal, phone, etc.

In order to accept credit cards you must first obtain a merchant account. The merchant account gives you the ability to charge credit cards and transfer those funds to your checking account. All merchant accounts are issued by a bank. You may apply for a merchant account at your local bank or may get a recommendation from your internet service provider. Be aware that many of the largest banks in the U.S. do not grant merchant accounts to merchants with an internet only business. In addition, U.S. banks do not grant merchant accounts to merchants outside of the U.S. You may need to investigate several sources before you find a bank that meets your needs.

While the merchant account allows you to accept and process credit cards, the merchant account itself does not necessarily allow you to process credit cards online. This service may be provided in conjunction with your merchant account, but can also be obtained separately through a gateway service. Using a credit card gateway service you can authorize and/or charge a shopping basket (or form) sale real-time during the shopping process. The gateway service approves or declines a credit card charge and transmits that decision to your web site for further processing.

All gateway services have different requirements and transmit information via different methods. See the instructions for your specific credit card gateway for these requirements. Regardless of the service, the basic process of authorizing credit cards via a gateway is as follows:

1. **Collect customer order, payment, billing, and shipping information**. You are generally required to transmit the customer's information (e.g., credit card number, etc.) to the gateway service via post or path arguments from a secure server. Therefore the customer must first provide order, payment, billing, and shipping information via the MGI shopping basket or even a simple form.
2. **Display customer information for confirmation and prepare data for the gateway service**. On the order confirmation page, you can display the payment, billing, shipping and order information for the customer to review as well as prepare your data in the specific format for your gateway service (e.g., specific post argument names).
3. **Transmit data to the gateway service**. On the order confirmation page, you will transmit the data to the gateway service via a FORM post, an HREF with path

arguments, etc. Depending on the service you choose, the customer may view forms from the gateway service during the shopping process or all gateway processing may occur in the background through redirects such that the customer never appears to "leave" your web site.

4. **Receive response from the gateway and process order**. The gateway service will either approve or decline the credit card charge. Many gateway services also "mirror" the payment, billing, and shipping information that was provided to them. The gateway service may respond to different URLs based on the outcome of the charge or may just provide the decision in a path argument or post argument which you can use in conjunction with a conditional to determine the course of action. In any case, the service will provide some response regarding the credit card charge (approve or decline). If the credit card is successfully processed, process the order. If the credit card is declined, present the use with an error and other processing options OR process the order noting the decline and contact the customer directly for resolution.

If you are integrating a gateway service into an MGI shopping basket, you will need to modify your order confirmation page and customize your order email. You may also need to construct a separate error page for credit cards that are declined.

The Accesspoint gateway service is used as an example below, however, you can use almost any gateway service provided that the mgiToken value is transmitted back to the order processing page. The mgiToken value is required in order to access the customer's shopping basket information. If the mgiToken value cannot be transmitted through the gateway service, you may need to save the order information in a cookie or database and access that information to process the order.

MGI Tags

- mgiConfirmOrder
- mgiGet
- mgiIf
- mgiInlineToken
- mgiPathArgument
- mgiSendOrder

Steps

1. Open the confirm order page in a text editor.
2. Prepare data for the gateway service.
3. Save the confirm order page.
4. Open the order processing page in a text editor.
5. Customize the order email.

6. Save the order processing page.
7. FTP the confirm order and order processing pages to the web server running MGI.
8. Configure gateway settings.
9. Process an order in test mode.

Step 1: Open the confirm order page in a text editor.

Open the confirm order page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 2: Prepare data for the gateway service.

On the confirm order page, prepare the payment, billing, shipping and order information to transmit to the gateway service.

In this example, the payment, billing, shipping and order information is submitted via hidden post arguments created by the mgiConfirmOrder tag to the Accesspoint gateway server (https://secure1.merchantmanager.com/ccgateway.asp). The post argument names are specific to Accesspoint. The post argument names may be different for other gateway services.

If your post argument names are different, create custom hidden post arguments for payment, billing, shipping and order information with embedded mgiPostArgument and mgiGet tags using information from the check out form and order totals from the mgiConfirmOrder tag. The mgiConfirmOrder tag automatically creates hidden post arguments for all post arguments posted to the confirm order page. If you use a custom check out page with form fields specific to your processor, you may not need to add custom hidden post arguments on the confirm order page.

In addition, the merchant ID, request type, and transaction type specific to Accesspoint are submitted via post arguments. The merchant ID is your specific Accesspoint ID. The request type is "ApprovalOnly" which creates a seamless transition from the confirm order page to the approved or denied URL. With the "ApprovalOnly" request type all processing is performed in the background and the customer never appears to "leave" your web site. The transaction type is "Sale" to authorize and charge the credit card, but can be set to "PreAuth" to only authorize the card. See the Accesspoint manual for additional information about request types and transaction types.

The Accesspoint gateway redirects approved and declined charge requests to specified URLs. Those URLs are included as post arguments and are appended with the mgiToken using the mgiInlineToken tag. The approved URL may be different from the denied URL, however in this example both URLs redirect to the same page and the approval message is compared in a conditional to determine the course of action.

The following is an example confirm order page with data prepared to transmit to Accesspoint. Note that the mgiConfirmOrder tag automatically creates the hidden post argument for payment, billing and shipping information.

```
<FORM
ACTION="https://secure1.merchantmanager.com/ccgateway.asp"
METHOD="Post">

<mgiComment>Account Post Arguments</mgiComment>

<input type="hidden" name="REQUESTTYPE"
value="ApprovalOnly">
<input type="hidden" name="TRANSTYPE"
value="SALE">
        <input type="hidden" name="MERCHANTID"
value="accountID">

<mgiComment>
Approved and Denied URLs with Tokens</mgiComment>

<mgiSet name="Denial">
<mgiInlineToken
url="https://secure.domain.com/folder/denied.mgi">
</mgiSet>

<input type="hidden" name="DENIEDURL"
value={mgiGet name="Denial"}>

<mgiSet name="Approval">
<mgiInlineToken
url="https://secure.domain.com/folder/approved.mgi">
</mgiSet>

<input type="hidden" name="APPROVEDURL"
value={mgiGet name="Approval"}>

<mgiComment>Confirm Order and Order Total</mgiComment>

<mgiConfirmOrder handle="Default"
shoppingBasketURL="http://www.domain.com/shop/">
</mgiConfirmOrder>
```

```
<input type="hidden" name="AMOUNT"
value={mgiGet name="mgiSBTotal"}>

</FORM>
```

Step 3: Save the confirm order page.

Save the changes you have made to the confirm order page.

Step 4: Open the order processing page in a text editor.

Open the order processing page in a text editing program that allows you to view and modify the HTML and code of the page.

Step 5: Customize the order email.

In a default shopping basket, payment, billing, and shipping information is posted to the order processing page from the confirm order page and the token value is used to obtain order information from the shopping basket database. Real-time processing requires a custom order email because the payment, billing and shipping information may be transmitted from the gateway service via a different method (path arguments rather than post arguments) and the information may have different names.

In this example, Accesspoint transmits the mgiToken and approval decision, plus the order, payment, billing, and shipping information via path arguments. The approval path argument ("approved" equals "Y" or "N") is compared in a conditional statement with the mgiIf tag. If the order is approved, the order is processed with the subject "Online Order". If the credit card is declined, the order is processed with the subject "Declined Order - Followup With Customer" and the reason for the decline is included in the email. If you prefer, you can give the customer a tokenized link to the check out page to enter a different payment option or correct the payment information they entered before re-submitting the order.

The order email is customized with the payment, billing, and shipping path arguments. The credit card number is not transmitted from the gateway service in this example, but that information is available online via the gateway service administrative interface if needed. The items orderd are entered by the mgiSendOrder tag based on the mgiToken path argument.

The following is an example order processing page with information from the Accesspoint gateway.

```
<mgiIf lhs={mgiPathArgument name="approved"}
relationship="equals" rhs="Y">
```

```
<mgiSet name="Subject">
Online Order
</mgiSet>

<mgiElse>

<mgiSet name="Subject">
Declined Order - Followup With Customer
</mgiSet>

</mgiIf>

<mgiSendOrder handle="Default"
shoppingBasketURL="http://www.domain.com/"
to="orders@domain.com" from="webmaster@domain.com"
mailServer="mail.domain.com"
subject={mgiGet name="Subject"}>

Processing Information
----------------------
    Message: <mgiPathArgument name="msg">
 Invoice No: <mgiPathArgument name="invoiceno">

Payment Information
-------------------
  Card Type: <mgiPathArgument name="CardType">
 Expiration: <mgiPathArgument name="EXPMO">
<mgiPathArgument name="EXPYE">
       Name: <mgiPathArgument name="CCName">

Billing Information
-------------------
       Name: <mgiPathArgument name="bName">
    Company: <mgiPathArgument name="bCompany">
    Address: <mgiPathArgument name="bAddress1">
             <mgiPathArgument name="bCity">
<mgiPathArgument name="bState">
<mgiPathArgument name="bZipCode">
             <mgiPathArgument name="bCountry">
      Phone: <mgiPathArgument name="bPhone">
        Fax: <mgiPathArgument name="bFax">
      Email: <mgiPathArgument name="bEmail">

Shipping Information
--------------------
```

```
       Name: <mgiPathArgument name="sName">
    Company: <mgiPathArgument name="sCompany">
    Address: <mgiPathArgument name="sAddress1">
             <mgiPathArgument name="sCity">
<mgiPathArgument name="sState">
<mgiPathArgument name="sZipCode">
             <mgiPathArgument name="sCountry">
      Phone: <mgiPathArgument name="sPhone">
        Fax: <mgiPathArgument name="sFax">
      Email: <mgiPathArgument name="sEmail">

Order Information
-----------------
<!-- Begin Template -->

 Product ID: &mgiDBFieldProductID;
        Qty: &mgiDBFieldQuantity;
    Product: &mgiDBFieldName;
 Price Each: $&mgiDBFieldPrice;
 Item Total: $&mgiSBItemPriceTotal;
<!-- End Template -->

        Tax: $&mgiSBTax;
      Total: $&mgiSBTotal;
</mgiSendOrder>
```

Step 6: Save the order processing page.

Save the changes you have made to the order processing page.

Step 7: FTP the confirm order and order processing pages to the web server running MGI.

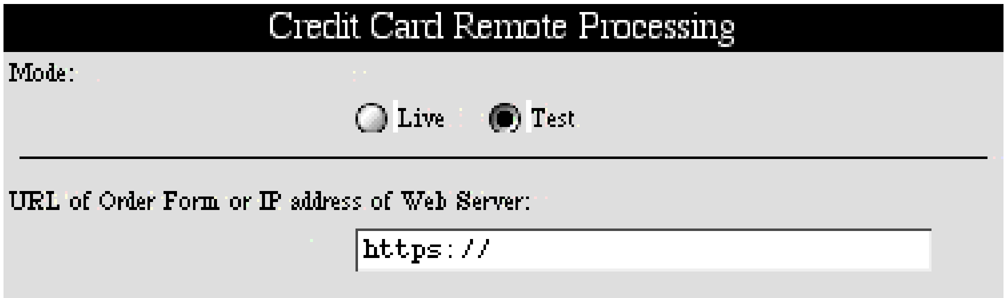
Upload the confirm order and order processing pages from your local computer to the web server using an FTP program.

Step 8: Configure gateway settings.

In order to use a gateway service, you may need to configure settings such as the URL of the confirm order page where orders originate, global approved and denied URLs, etc

For the Accesspoint gateway service, access the Transaction Manager interface. Select "Administration" then select "Remote Processing". From the Java menu or pop-up, just select "configure my remote processing options". Under "Mode", select "Test" while testing your service or select "Live" to use the service with a production web site. Under

"URL of Order Form or IP address of Web Server", enter "https://" if your orders originate from a secure server or enter "http://" if you orders originate from a non-secure server. Leave all other settings as default and click the "Update" button at the bottom of the page to save the settings.



Credit Card Remote Processing

Mode:

○ Live ● Test

URL of Order Form or IP address of Web Server:

https://

Step 9: Process an order in test mode.

Set the gateway options to "test" mode and process a shopping basket order.

For Accesspoint, set the "Remote Processing" mode to "Test". Use the credit card number "5111111111111111" with any other payment, billing, or shipping information to see an approved order and use the credit card number "4111111111111111" with any other payment, billing, or shipping information to see a denied order. When the real-time system is working to your satisfaction, set the "Remote Processing" mode to "Live".

[Return to the Shopping Online Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Using ODBC Databases

In the **Beginner Tutorial** section, learn guidelines for ODBC set up, tags that are affected by ODBC databases, and other isues that are specific to ODBC databases. In the **Reference** section, view a complete technical reference for each tag used for polls. In the **FAQs** section, read answers to frequently asked questions.

Beginner ODBC Tutorials

MGI is able to access ODBC databases with ODBC drivers. Some drivers are included in the MGI 2.x installers while others are included as part the the ODBC database's package. The following tutorials address some of the issues that you may encounter when using MGI with an ODBC database. We recommend that you read all of the tutorials before using MGI with an ODBC database. **Please contact the ODBC database manufacturer for any questions regarding the installation of the database itself**.

- B1. ODBC Database Set Up Guidelines
- B2. Tags Affected by ODBC Databases
- B3. MGI ID and Other Field Requirements
- B4. Potential Pitfalls of ODBC Databases
- B5. Using Flat File ODBC Databases (Text, dBase, Excel, Paradox, etc.)

ODBC MGI Tag Reference (tags that can use ODBC databases)

- mgiAuthenticateDB
- mgiBannerAd
- mgiCollectUserInfo
- mgiConfirmOrder
- mgiCounter
- mgiCreditBank
- mgiDynamicPopup
- mgiEditDatabase
- mgiGet
- mgiGuestbookDB
- mgiModifyDatabase

- mgiPoll
- mgiQuiz
- mgiRotate
- mgiSearchDatabase
- mgiSendOrder
- mgiSet
- mgiShoppingBasket

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

ODBC Database Set Up Guidelines

Before you can use MGI with an ODBC database, the server administrator must properly set up and test the ODBC data source in the control panels of the operating system. Next, the ODBC data source has been set up, the server administrator must configure the ODBC username and password in the database's proprietary interface or application.

Some database systems require user permissions to be set on a per-database basis. MGI cannot use a database unless MGI has been granted access to the database.

Some database drivers do not automatically enable important functions such as multi-threading, updates and deletes, and other important options. In order for MGI to function to the database's capacity, you must enable these functions when you configure the data source.

Please contact the ODBC database manufacturer for any questions regarding the installation of the database itself or configuration of the ODBC database.

MGI is able to access ODBC databases with ODBC drivers. Some drivers are included in the MGI 2.x installer while others are included as part the the ODBC database's package. The following drivers are included with the MGI Macintosh and NT/2000 installers.

Macintosh Drivers

- dBASE
- FoxPro
- Oracle 7
- Text (flat file driver)

MGI works with FileMaker, but the database driver is included with the FileMaker program and is not installed by MGI.

Win32 Drivers

- Btrieve
- dBASE
- Excel Workbook
- Informix 7.x, 9.x
- Oracle 7, 8
- Paradox
- PROGRESS
- SQL Server 7, 2000
- SQLBase

- Sybase Adaptive Server
- Text (flat file driver)

[Return to the Using ODBC Databases Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Tags Affected by ODBC Databases

In general, most MGI tags that use a database can be configured to use an ODBC database by setting the odbcDataSource, odbcUsername, and odbcPassword tag parameters.

The tag that is most directly affected by the use of an ODBC database versus the built-in MGI database is mgiEditDatabase. When a new database is created, mgiEditDatabase will attempt to add a numerical database field named "MGIIDFIELD". This is the default field that MGI uses as a unique ID for each database record. In addition to creating the "MGIIDFIELD" database field, MGI will attempt to make the "MGIIDFIELD" an auto-incrementing number. However, MGI will not always be able to create and auto-increment the "MGIIDFIELD" using an ODBC database.

If your ODBC database does not support auto-incrementing fields, the server administrator must populate the field manually. If you choose to not use the MGIIDFIELD as your unique ID field, you may remove it from the database, create a different unique ID field and reference your custom unique ID field by in the "uniqueIDFieldName" parameter of the mgiEditDatabase tag.

IMPORTANT: See the MGI ID and Other Field Requirements section for strict requirements on the contents of the MGIIDFIELD.

IMPORTANT: If MGI finds "MGIIDFIELD" as the **first** field in the database, MGI will try to use that field by default unless you specifically override it by including the "uniqueIDFieldName" parameter in the mgiEditDatabase tag. If the "MGIIDFIELD" is not the first field in the database **and** the "uniqueIDFieldName" parameter is not provided, the mgiEditDatabase tag will display an error.

The following tags will only work with an ODBC database that has an auto-incrementing (sometimes called a serial number) numerical field type. If MGI cannot create the field layout via the ODBC drivers, the server administrator will be responsible for recreating the exact field layout that is required by each MGI tag, including the "MGIIDFIELD" (which must be the first field and be an auto-incrementing number).

- mgiAuthenticateDB
- mgiBannerAd
- mgiCounter
- mgiCreditBank
- mgiGuestbookDB
- mgiPoll
- mgiQuiz
- mgiShoppingBasket

If necessary, you can determine the field layout of the databases for these MGI tags by first using the tag to create an internal MGI database. Next, create a page with the mgiEditDatabase tag and the "showMGIDatabases" parameter set to "Yes". Upload that database admin page to the server, access it with a web browser, and view the field layout for the database of any tag listed above.

[Return to the Using ODBC Databases Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

MGI ID and Other Field Requirements

MGIIDFIELD Requirements

The "MGIIDFIELD" is expected to be an ever-increasing number that is always in ascending order. If you are populating this field manually, you are responsible for keeping these numbers in order (which is possibly automatable with an mgiCounter using mgiModifyDatabase but not automatable inside mgiEditDatabase). Numbers can be missing from the order (i.e. 1, 2, 3, 5, 9, 132) due to the deletion of records, but no number can be less than the "MGIIDFIELD" value of the previous record.

Field Deletion

An ODBC database must contain at least one field at all times. If a user wants to delete the "MGIIDFIELD" that was added by default, they must first create another field.

[Return to the Using ODBC Databases Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Potential Pitfalls of ODBC Databases

All ODBC databases are not created alike! MGI can only use the tools that are available in the ODBC database driver. The following are some pitfalls that you may encounter when implementing MGI with an ODBC database.

1. Some ODBC databases will not allow the creation of new databases through ODBC drivers. With these databases you will have to do all database creation using the ODBC database's proprietary interface or application.
2. Some ODBC databases will not allow field management through ODBC drivers. With these databases you will have to do all database field layout using the ODBC database's proprietary interface or application.
3. MGI can only use the fields types that are available via the ODBC's driver. Even if the database supports many different field types in its proprietary interface, the database's driver may not necessarily support all of those field types via ODBC. This lack of field types could cause problems for some tags that require predictable field types to store their internal data.
4. Some databases do not provide an efficient way to count records. Some databases also do not provide an efficient way to position on a certain record. The mgiEditDatabase tag and any other tags that produces a record/result count or that iterates through records may perform poorly with such databases.

[Return to the Using ODBC Databases Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Using Flat File ODBC Databases

Flat file ODBC drivers such as the Text, dBASE, Excel, and Paradox drivers may not perform well in multi-threaded situations. Multiple **reading** threads should perform as expected (i.e. mgiSearchDatabase). However, multiple **writing** threads could be refused when the flat file is locked for updates. Flat file ODBC drivers are only recommended for moderate-to-high traffic sites where very few updates or insertions are being performed.

[Return to the Using ODBC Databases Menu]

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

Referencing MGI

In the Referencing MGI section, you will find a complete list of definitions and a technical reference for each MGI tag. Each MGI tag was written for a specific purpose, but you can choose tag options to customize the tag's function and you can combine tags to create new functions. Read the Referencing MGI section to customize the function of MGI for your web site.

mgiAbort
mgiAuthenticate
mgiAuthenticateDB
mgiBannerAd
mgiButton
mgiBuyMe
mgiCloak
mgiCollectUserInfo
mgiComment
mgiConfirmOrder
mgiCounter
mgiCreditBank
mgiCreditCard
mgiDate
mgiDoNotProcess
mgiDynamicList
mgiDynamicPopup
mgiEditDatabase
mgiEncryptURL
mgiGet
mgiGetCookie
mgiGetErrorParameter
mgiGetFileInfo
mgiGuestbook
mgiGuestbookDB
mgiIf
mgiIncludeFile
mgiIncludeHTTP
mgiIncrement
mgiInfoDumper
mgiInlineDoNotProcess
mgiInlineIf
mgiInlineString
mgiInlineToken
mgiJulianDay
mgiLink
mgiListFolder
mgiLoop
mgiMath
mgiModifyDatabase
mgiModifyFile
mgiModifyFileSystem
mgiPathArgument
mgiPGP
mgiPoll
mgiPOP
mgiPostArgument
mgiQuiz
mgiRandomNumber
mgiRedirect
mgiRequest
mgiRollOver
mgiRotate
mgiSearchDatabase
mgiSendMail
mgiSendOrder
mgiSet
mgiSetCookie
mgiSetResponseHeader
mgiShoppingBasket
mgiStop
mgiString
mgiSwitch
mgiTime
mgiToken
mgiValidateData

Escaped String Format
Regular Expression Symbols

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

The mgiAbort Tag

Tag Behavior

Use the mgiAbort tag completely stops the processing of the page including MGI tags, text and HTML. When the mgiAbort tag is processed by MGI, any part of the page processed prior to the mgiAbort tag is served to the browser. (see also mgiStop)

Tag Syntax

The mgiAbort tag has no required parameters and no optional parameters. The tag form is:

```
<mgiAbort>
```

Required Parameters:

- **None.**

Optional Parameters:

- **None.**

Example Usage and Output

```
<mgiIf lhs={mgiGet name="Errors"} relationship="isNotEmpty">
</BODY>
</HTML>
<mgiAbort>
</mgiIf>
```

In this example, the mgiAbort tag is used to abort a page after checking for errors using a conditional statement. Notice that ending Body and HTML elements are entered before the mgiAbort tag since the mgiAbort tags stops all processing of the page which may affect how the browser displays the information.

Suggested Usage

- Error Checking

The mgiAuthenticate Tag

Tag Behavior

Use the mgiAuthenticate tag to password-protect individual pages by displaying an authentication dialogue box and requiring a username and password. If the authentication fails (i.e. the username or the password does not match those coded in the tag), an error page is displayed.

Tag Syntax

The mgiAuthenticate tag has two required parameters and no optional parameters. The tag form is:

```
<mgiAuthenticate username="Name" password="Code">
```

Required Parameters:

- **username** - The username is the authorized login (i.e., name) necessary to access the page containing the mgiAuthenticate tag. The username must be unique. The username is case-sensitive and can consist of letters, numbers, multiple words, spaces and ASCII characters.
- **password** - The password is the authorized security code necessary to access the page containing the mgiAuthenticate tag. The password is case-sensitive and can consist of letters, numbers, multiple words, spaces and ASCII characters. A secure password should contain at least 8 characters and include both numbers and letters.

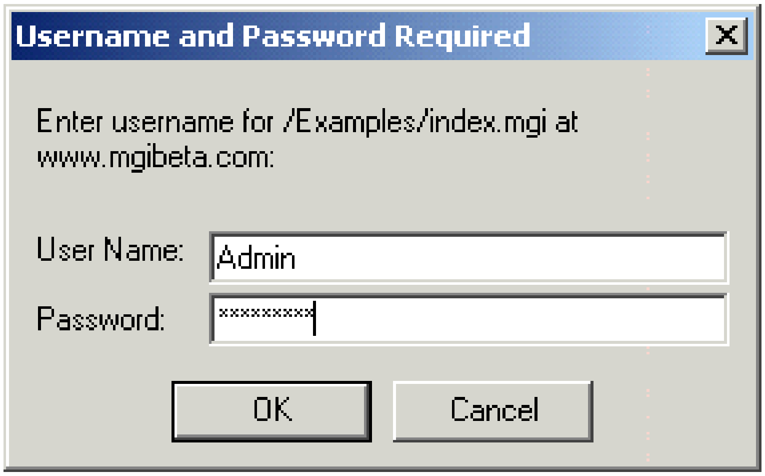
Optional Parameters:

- **None.**

Example Usage and Output

```
<mgiAuthenticate username="Admin" password="39K2p2w8M">
```

In this example, the username and password required to access a page are case-sensitive. The mgiAuthenticate tag, username parameter and password parameter are placed in the HTML code of a page named index.mgi. When a visitor accesses index.mgi, a dialogue box is displayed requesting a User Name and Password.



Username and Password Required

Enter username for /Examples/index.mgi at www.mgibeta.com:

User Name: Admin

Password: *********

If the username and password entered in the dialogue box match the username and password coded in the parameters of the mgiAuthenticate tag exactly, then the index.mgi page is displayed to the visitor. If an incorrect username or password is entered in the dialogue box, an error is displayed. If the correct username and password is entered in the dialogue box with incorrect capitalization, an error is displayed.

Suggested Usage

- Password Protection

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiAuthenticateDB Tag

Tag Behavior

Use the mgiAuthenticateDB tag to protect access to pages using security options such as multiple usernames and passwords and IP numbers. Managing the access of specific users and groups of users is done via a web-based administration interface or via HTML forms and different modes of the mgiAuthenticateDB tag.

Tag Syntax

The mgiAuthenticateDB tag has eight modes. Each mode has different required and optional parameters. The eight modes of mgiAuthenticateDB are:

- **admin** - Creates a web-based interface to manage specific users and groups of users.
- **authenticate** - Regulates access to a page by requiring authentication of username, password and IP number (if specified). "Authenticate" is the defalt mode of the mgiAuthenticateDB tag.
- **authenticateIPOnly** - Regulates access to a page by requiring authentication of an IP number only.
- **addUser** - Adds a new user to the authentication database.
- **deleteUser** - Deletes a user from the authentication database.
- **queryUser** - Queries the authentication database for the existence of a user.
- **changePassword** - Changes the password of an existing user.
- **sendPassword** - Emails a user's password to the email address listed in the authentication database.

Click any name above to view the mode's full Tag Syntax and Example Usage and Output.

Suggested Usage

- Password Protection for Multiple Users
- Local LAN IP Authentication
- Subscription or Member Systems
- User Sign-Up Forms

- Quizzes Start Date and Times

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiAuthenticateDB Tag

Admin Mode

Return to the mgiAuthenticateDB Main Menu

Tag Syntax

The Admin mode of mgiAuthenticateDB has one required parameter and four optional parameters. The tag form is:

```
<mgiAuthenticateDB mode="Admin" advancedSearch="On/Off"
headerColor="Hex Code" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiAuthenticateDB tag performs. In "**admin**" mode, the mgiAuthenticateDB tag creates a web-based interface that allows you to manage specific users and groups of users.

Optional Parameters:

- **advancedSearch** - The advancedSearch parameter determines whether the advanced search feature is available in the web-based administration interface. The advanced search feature allows you to enter complex search strings that are not available with the built-in search functions (e.g., "NOT" searches, "OR" searches, searching the same field multiple times, etc.). The search time will increase as the search complexity increases. If the advancedSearch parameter value is "**On**", then the advanced search field is displayed in the search screen of the admin interface. If the advancedSearch parameter value is "**Off**", then the advanced search field is not displayed in the search screen of the admin interface. The default value is "Off".
- **headerColor** - The headerColor parameter is the hex code (without the # symbol) of the color for the header table cells in the admin display of mgiAuthenticateDB. The default color is "cccccc" (gray).
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, authentication information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword

parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

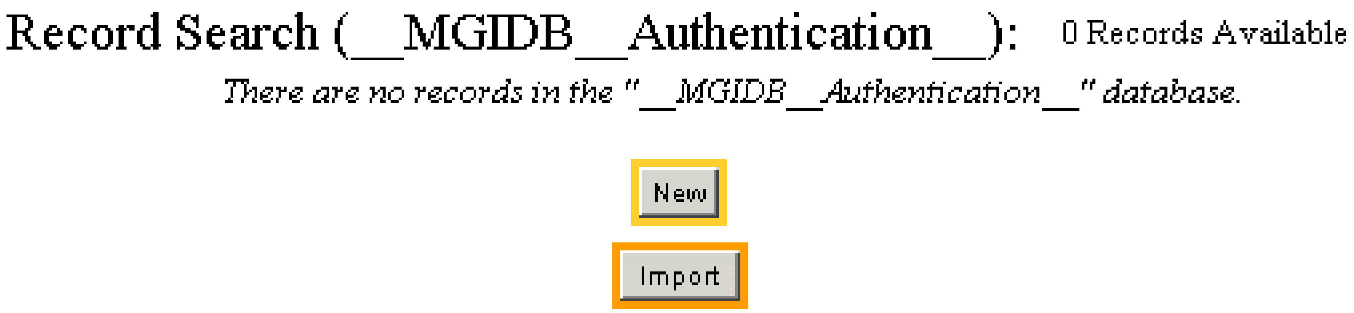
Example Usage and Output

```
<mgiAuthenticateDB mode="admin" advancedSearch="Yes">
```

In admin mode, the mgiAuthenticateDB tag dynamically creates and displays a web-based interface to manage authentication information for specific users and groups of users.

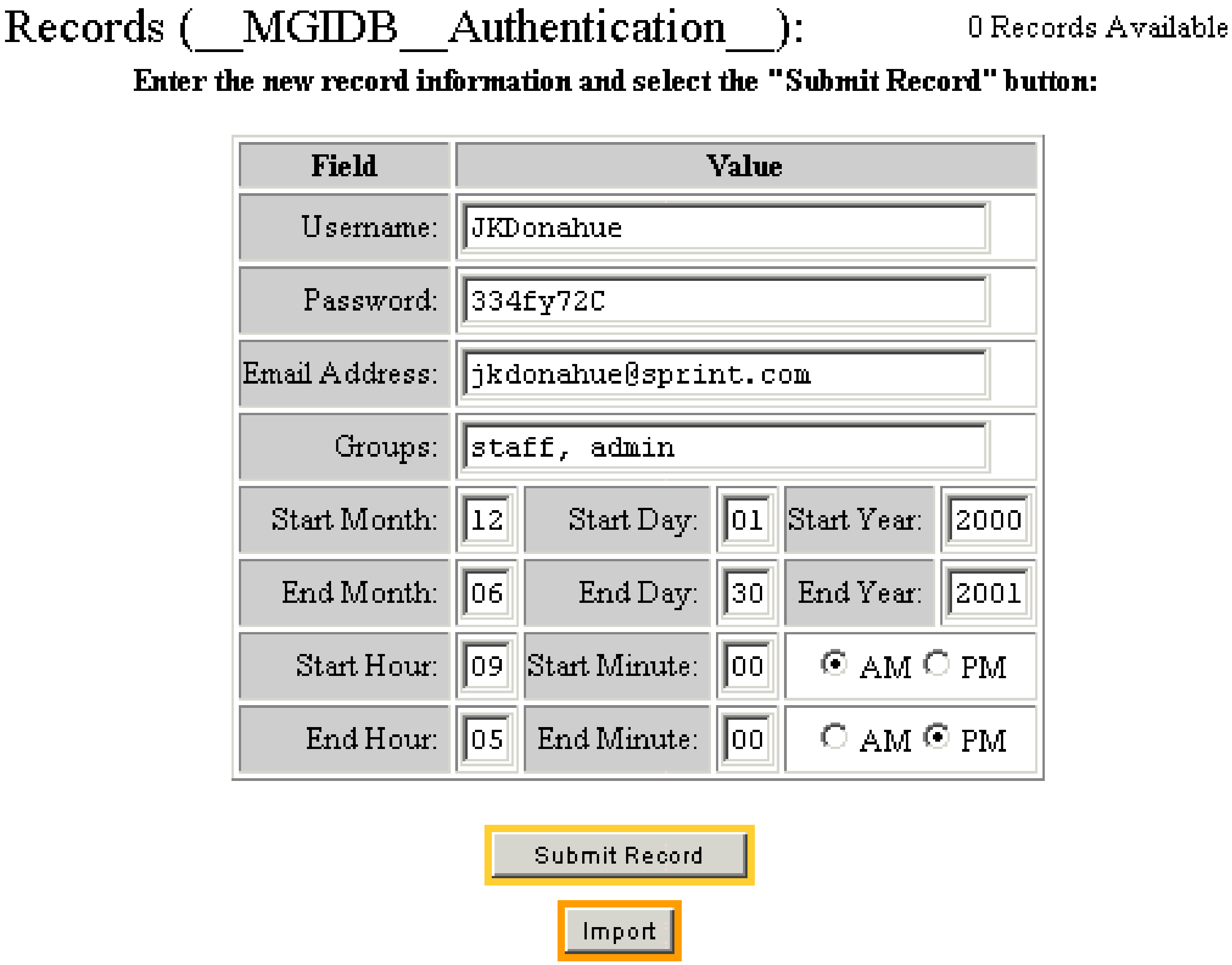
Adding New Users (see Importing below)

The first time you access a page with the mgiAuthenticateDB tag in admin mode, buttons for creating a new user and importing users display.


Record Search (__MGIDB__Authentication__): 0 Records Available

There are no records in the "__MGIDB__Authentication__" database.

To create a new user, click the "New" button. Enter the new user information in the form that displays. The username and password fields are required. In the Username field, enter the user's unique, case-sensitive login identification. In the Password field, enter the user's case-sensitive security code. In the Email Address field, enter the user's email address. The email address will be used to send the user's password in SendPassword mode. In the Groups field, enter the group name that the user belongs to. For multiple groups, enter a comma-delimited list in the Groups field. In the start date fields, enter the numeric month, day and 4-digit year when the user's username and password become valid. In the end date fields, enter the numeric month, day and 4-digit year when the user's username and password are no longer valid. In the start time fields, enter the time that the user's username and password become valid on the start date. In the end time fields, enter the time that the user's username and password are no longer valid on the end date. Click "Submit Record" to add the user. The message "Record successfully added." is displayed when the addition is complete and a blank form for adding additional users is displayed.



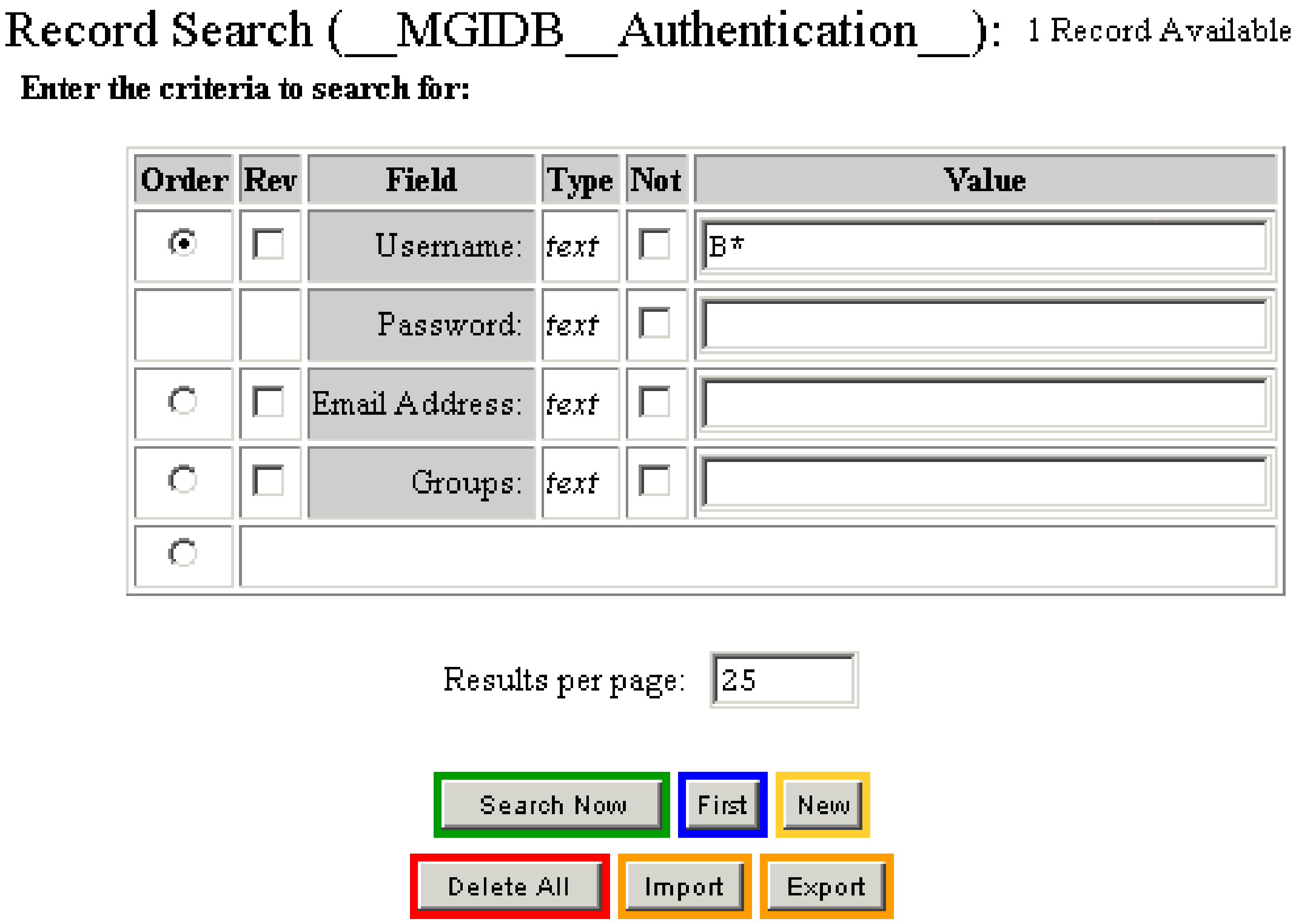
Records (__MGIDB__Authentication__): 0 Records Available

Enter the new record information and select the "Submit Record" button:

Field	Value				
Username:	JKDonahue				
Password:	334fy72C				
Email Address:	jkdonahue@sprint.com				
Groups:	staff, admin				
Start Month:	12	Start Day:	01	Start Year:	2000
End Month:	06	End Day:	30	End Year:	2001
Start Hour:	09	Start Minute:	00	● AM ○ PM	
End Hour:	05	End Minute:	00	○ AM ● PM	

Searching Authentication Records

When there is one or more users in the authentication database, the options for searching or browsing the records are available at the bottom of the admin interface. When you access the mgiAuthenticateDB admin with existing users, the search form is displayed by default. At any location in the admin interface, click the "Search" button to display the search form. Search for a specific user by entering search criteria in the Username, Password, Email Address and/or Groups fields under the Value column. To perform partial searches, enter an asterisk (*) for a wildcard value. For example, to search with all usernames beginning with the letter "B", enter "B*" in the Username field. Search criteria is not case-sensitive. To view all authentication records, leave the search criteria in all fields blank. To order your search results using the Username, Email Address, or Groups field, select the radio button beside the appropriate field in the Order column. Search results are displayed in ascending order (A to Z, smallest to largest) by default. To display search results in descending order (Z to A, largest to smallest), click the checkbox beside the appropriate field in the Rev column. Search results are displayed 25 per page by default. To display greater or fewer search results per page, enter a value in the Results Per Page field. To search, click the "Search Now" button. To view a specific record in the search results, select the radio button beside the record and click the "View" button.



Record Search (__MGIDB__Authentication__): 1 Record Available

Enter the criteria to search for:

Order	Rev	Field	Type	Not	Value
●	□	Username:	*text*	□	B*
		Password:	*text*	□	
○	□	Email Address:	*text*	□	
○	□	Groups:	*text*	□	
○					

Results per page: 25

Browsing Authentication Records

To browse authentication records, click the "First" button. The first authentication record appears. To view the next authentication record, click the ">>" button. To view the previous authentication record, click the "<<" button.

Modifying Authentication Records

To modify an authentication record, locate the record by searching or browsing for the record. Change the information in any field and click the "Save" button.

Importing and Exporting

Tab-delimited user information can be imported into the authentication database. To import new users, create a tab-delimited file with columns in this order: Username, Password, Email Address, Groups, Start Date, End Date, Start Time and End Time. A username and password is required for each user. For the Username, enter the user's unique, case-sensitive login identification. For the Password, enter the user's case-sensitive security code. For the Email Address, enter the user's email address. The email address will be used to send the user's password in SendPassword mode. For the Groups, enter the group name that the user belongs to. For multiple groups, enter a comma-delimited list in the Groups field. For the start date, enter the julian date when the user's username and password become valid. For the end date, enter the julian date when the user's username and password are no longer valid. For the start time, enter the time (in military format) that the user's username and password become valid on the start date. For the end time, enter the time (in military format) that the user's username and password are no longer valid on the end date.

To import, click the "Import" button. In the FileName field, enter the name of the tab-delimited file on the server that contains your import information. Click the "Import Now" button to import the users. If a username in your import file matches an existing username in the authentication database, the user's information is not imported.

MGI exports authentication records in tab-delimited text form. Exported authentication records are **not** encrypted. To export records, click the "Export" button. Enter the name of a new file to contain your authentication records and click the "Export Now" button. **Warning: An export file will overwrite an existing file of the same name!**

Deleting Records

To delete an individual authentication record, locate the record by searching or browsing for the record. Click the "Delete" button. Verify that you intended to delete the record by clicking the "Yes" button.

To delete all authentication records, click the "Delete All" button. Verify that you intended to delete the record by clicking the "Yes" button.

Return to the mgiAuthenticateDB Main Menu

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiAuthenticateDB Tag

Authenticate Mode

Return to the mgiAuthenticateDB Main Menu

Tag Syntax

The Authenticate mode of mgiAuthenticateDB has no required parameters and sixteen optional parameters. The tag form is:

```
<mgiAuthenticateDB mode="Authenticate" group="Name"
startDate="Date" endDate="Date" startTime="Time"
endTime="Time" dailyStartTime="Time"
dailyEndTime="Time" setCookie="Yes/No"
cookieExpirationDate="Date" cookieExpirationTime="Time"
allowedIP="IP Number" deniedIP="IP Number"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **None.**

Optional Parameters:

- **mode -** The mode is the function that the mgiAuthenticateDB tag performs. In "**authenticate**" mode, the mgiAuthenticateDB tag regulates access to the page containing the mgiAuthenticateDB tag by requiring a valid username, password and IP number (if specified). "Authenticate" is the default mode of the mgiAuthenticateDB tag.
- **group -** The group is the name of the user group allowed to access the page with a valid username and password. If a group is not specified, all groups are allowed to access the page with a valid username and password.
- **startDate -** The startDate is the first day access to the page is allowed with a valid username and password. The syntax of the date must be a julian day (see the mgiJulianDay tag). The startDate parameter can be used in conjunction with the endDate parameter to specify a definite time frame when access to a page is allowed or the startDate parameter can be entered alone to specify a start date for access without an expiration. If the startDate parameter is not included, the page can be accessed immediately with a valid username and password.
- **endDate -** The endDate is the last day access to the page is allowed with a valid

username and password. The syntax of the date must be a julian day (see the mgiJulianDay tag). The endDate parameter can be used in conjunction with the startDate parameter to specify a definite time frame when access to a page is allowed or entered alone to specify an expiration for access with an immediate start date. If the endDate parameter is not included, the page can be accessed any time after the specified start date.

- **startTime -** The startTime is the time that access to the page is allowed on the specified start date. The syntax of the time must be military format (i.e., hhmm). If the startTime parameter is not included, access to the page is available at midnight on the specified start date.
- **endTime -** The endTime is the time that access to the page is no longer allowed on the specified end date. The syntax of the time must be military format (i.e., hhmm). If the endTime parameter is not included, access to the page ends after 11:59 PM on the specified end date.
- **dailyStartTime -** The dailyStartTime is the time that access to the page is allowed during each day from the specified start date to the specified end date. The syntax of the time must be military format (i.e., hhmm). If the dailyStartTime is not included, then access to the page is valid each day from midnight until the daily end time during the period from the start time on the start date to the end time on the end date.
- **dailyEndTime -** The dailyEndTime is the time that access to the page is no longer allowed during each day from the specified start date to the specified end date. The syntax of the time must be military format (i.e., hhmm). If the dailyStartTime is not included, then access to the page is valid each day from the daily start time until 11:59 PM during the period from the start time on the start date to the end time on the end date.
- **setCookie -** The setCookie parameter determines whether a cookie with the user's authentication information is set during the user's first visit and retrieved upon subsequent visits to the page. If the setCookie parameter value is "**Yes**", then an encrypted cookie with the user's authentication information is set during the user's first visit to the page and retrieved during subsequent visits to the page until the cookie expiration time on the cookie expiration date. If the setCookie parameter value is "**No**", then the user must enter their authentication information during each visit to the page. The default value is "No". If you include the setCookie parameter, the cookieExpirationDate and cookieExpirationTime parameters are required.
- **cookieExpirationDate -** The cookieExpirationDate is the date that the user's authentication cookie expires. The syntax of the date must be a 2-digit month, 2-digit, and a 4-digit year separated by hyphens (i.e., mm-dd-yyyy). The cookieExpirationDate parameter is required if you include the setCookie parameter.
- **cookieExpirationTime -** The cookieExpirationTime is the time that the user's authentication cookie expires on the cookie expiration date. The syntax of the time must be military format seperated by a colon (i.e., hh:mm). The cookieExpirationTime parameter is required if you include the setCookie parameter.

- **allowedIP** - The allowedIP is the individual IP number, hostname, or the block of IP numbers or hostnames that are allowed to access the page with a valid username and password. For an individual IP number, the parameter value is the IP address. For multiple, individual IP numbers, include the allowedIP parameter multiple times within the mgiAuthenticateDB tag (e.g., allowedIP="63.45.215.48" allowedIP="208.196.25.1"). For a block of IP numbers or hostnames, use the asterisk (*) to represent a wildcard in any part of the IP number (e.g., allowedIP="201.199.24.*" or allowedIP="24.15.*.*"). If the allowedIP parameter is not included, a user from any IP number with a valid username and password is allowed to access the page. This parameter may be specified multiple times.
- **deniedIP** - The deniedIP is the individual IP number, hostname, or block of IP numbers or hostnames that are not allowed to access the page even if they have a valid username and password. For an individual IP number, the parameter value is the IP address. For multiple, individual IP numbers, include the deniedIP parameter multiple times within the mgiAuthenticateDB tag (e.g., deniedIP="155.26.14.241" deniedIP="155.26.54.214"). For a block of IP numbers or hostnames, use the asterisk (*) to represent a wildcard in any part of the IP number (e.g., deniedIP="213.14.144.*" or deniedIP="24.15.*.*"). If the deniedIP parameter is not included, users from any IP number with a valid username and password is allowed to access the page. This parameter may be specified multiple times.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, authentication information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Example Usage and Output

Multiple Username and Password Authentication

```
<mgiAuthenticateDB>
```

To password-protect a page with multiple usernames and passwords, enter users via the web-based admin or via a form processed by the addUser mode of mgiAuthenticateDB. Enter the mgiAuthenticateDB tag on any page you wish to password-protect. In this example, a user from any group with a valid username and password can access the page.

Groups

```
<mgiAuthenticateDB group="admin">
```

Password access to a page can be limited to users in a specific group. To limit access to one group, enter the group name in the Group parameter. In this example, only users in the "admin" group can access the page. Users in other groups cannot access the page even if they have a valid username and password.

Date and Time Limited Authentication

```
<mgiAuthenticateDB startDate={mgiJulianDay month="11;"
day="1" year="2000"}
endDate={mgiJulianDay month="6;" day="30" year="2001"}
startTime="0800" endTime="1700" dailyStartTime="0800"
dailyEndTime="1700">
```

Using date and time limited authentication, access to a page depends on the current date and time. In this example, users in any group with a valid username and password will be granted access to the page at 8 AM on 1 November 2000. Valid users will no longer be able to access the page after 5 PM on 30 June 2001. In addition, valid users will only be granted access from 8 AM to 5 PM each day from 8 AM on 1 November 2000 until 5 PM on 30 June 2001.

IP Authentication

```
<mgiAuthenticateDB group="staff" allowedIP="204.15.26.*">
```

Using the allowedIP and deniedIP parameters can further restrict the users who can access a page. In this example, only users accessing the page from the 204.15.26.* LAN with a valid username and password are allowed. For example, a user accessing the page from 204.15.25.241 with a valid username and password would be allowed, but a user accessing the page from 208.251.177.15 would not be allowed to access the page even if they have a valid username and password.

Cookies

```
<mgiSet name="1YrMonthExpiration">
<mgiDate format="paddedNumericMonth" text="-"
format="paddedDay">-<mgiMath resultPrecision="0"
workPrecision="2">
<mgiDate format="longYear">+1
</mgiMath>
</mgiSet>
```

```
<mgiAuthenticateDB setCookie="Yes"
cookieExpirationDate={mgiGet name="1YrMonthExpiration"}
cookieExpirationTime="2359">
```

Setting a user's information in a cookie makes it convenient for the user to access a password-protected page. During the user's first visit, an encrypted cookie with the user's authentication information is set. The user will not have to re-enter their username and password when they access the page until the cookie expires. In this example, users will be able to access the page until their cookie expires at 11:59 PM 1 year from the current date. To accomplish the dynamic expiration date, 1 is added to the current year and is set in a variable with the current month and day. The variable value is then embedded in the cookieExpirationDate parameter of the mgiAuthenticateDB tag.

Return to the mgiAuthenticateDB Main Menu

The mgiAuthenticateDB Tag

AuthenticateIPOnly Mode

Return to the mgiAuthenticateDB Main Menu

Tag Syntax

The AuthenticateIPOnly mode of mgiAuthenticateDB has three required parameters and no optional parameters. The tag form is:

```
<mgiAuthenticateDB mode="AuthenticateIPOnly"
allowedIP="IP Number" deniedIP="IP Number">
```

Required Parameters:

- **mode -** The mode is the function that the mgiAuthenticateDB tag performs. In "**authenticateIPOnly**" mode, the mgiAuthenticateDB tag regulates access to the page containing the mgiAuthenticateDB tag by requiring the user to access the page from a specific IP number. The user is not required to have a valid username or password in "authenticateIPOnly" mode.
- **allowedIP -** The allowedIP is the individual IP number, hostname, or the block of IP numbers or hostnames that are allowed to access the page with a valid username and password. For an individual IP number, the parameter value is the IP address. For multiple, individual IP numbers, include the allowedIP parameter multiple times within the mgiAuthenticateDB tag (e.g., allowedIP="63.45.215.48" allowedIP="208.196.25.1"). For a block of IP numbers or hostnames, use the asterisk (*) to represent a wildcard in any part of the IP number (e.g., allowedIP="201.199.24.*" or allowedIP="24.15.*.*"). This parameter may be specified multiple times. If one or more deniedIP parameters are provided, then the allowedIP parameter is not required.
- **deniedIP -** The deniedIP is the individual IP number, hostname, or block of IP numbers or hostnames that are not allowed to access the page even if they have a valid username and password. For an individual IP number, the parameter value is the IP address. For multiple, individual IP numbers, include the deniedIP parameter multiple times within the mgiAuthenticateDB tag (e.g., deniedIP="155.26.14.241" deniedIP="155.26.54.214"). For a block of IP numbers or hostnames, use the asterisk (*) to represent a wildcard in any part of the IP number (e.g., deniedIP="213.14.144.*" or deniedIP="24.15.*.*"). This parameter may be specified multiple times. If one or more allowedIP parameters are provided, then the deniedIP parameter is not required.

Optional Parameters:

- **None.**

Example Usage and Output

```
<mgiAuthenticateDB mode="AuthenticateIPOnly"
allowedIP="208.63.25*" deniedIP="206.63.*.*">
```

In this example, users accessing the page from the 208.63.25 IP block can access the page, but users accessing the page from any other part of the 206.63 IP block are not granted access.

Return to the mgiAuthenticateDB Main Menu

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiAuthenticateDB Tag

AddUser Mode

Return to the mgiAuthenticateDB Main Menu

Tag Syntax

The AddUser mode of mgiAuthenticateDB has three required parameters and nine optional parameters. The tag form is:

```
<mgiAuthenticateDB mode="AddUser" username="Name"
password="Code" group="Name" emailAddress="Email"
startDate="Date" endDate="Date" startTime="Time"
endTime="Time" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiAuthenticateDB tag performs. In "**addUser**" mode, the mgiAuthenticateDB tag adds a new user's authentication information to the authentication database.
- **username** - The username is the user's authorized login (i.e., name). The username is case-sensitive and must be unique across all groups. The username can consist of letters, numbers, multiple words, spaces and ASCII characters.
- **password** - The password is the user's authorized security code. The password is case-sensitive and can consist of letters, numbers, multiple words, spaces and ASCII characters. A secure password should contain at least 8 characters and include both numbers and letters.

Optional Parameters:

- **group** - The group is the name of the user group that the user belongs to. To add the user to multiple groups enter a comma-delimited list of group names (e.g., "admin, member"). Group names are case-sensitive.
- **emailAddress** - The emailAddress is the user's email address. Using the sendPassword mode of mgiAuthenticateDB, the user's password will be mailed to this email address.
- **startDate** - The startDate is the first day access to the page is allowed with a valid username and password. The syntax of the date must be a julian day (see the mgiJulianDay tag). The startDate parameter can be used in conjunction with the endDate parameter to specify a definite time frame when access to a page is allowed or the startDate parameter can be entered alone to specify a start date for access without

an expiration. If the startDate parameter is not included, the page can be accessed immediately with a valid username and password.

- **endDate** - The endDate is the last day access to the page is allowed with a valid username and password. The syntax of the date must be a julian day (see the mgiJulianDay tag). The endDate parameter can be used in conjunction with the startDate parameter to specify a definite time frame when access to a page is allowed or entered alone to specify an expiration for access with an immediate start date. If the endDate parameter is not included, the page can be accessed any time after the specified start date.
- **startTime** - The startTime is the time that the user's username and password are valid on the specified start date. The syntax of the time must be military format (i.e., hhmm). If the startTime parameter is not included, the user's username and password are valid at midnight on the specified start date.
- **endTime** - The endTime is the time that user's username and password are no longer valid on the specified end date. The syntax of the time must be military format (i.e., hhmm). If the endTime parameter is not included, the user's username and password are no longer valid after 11:59 PM on the specified end date.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, authentication information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Example Usage and Output

```
<mgiSet name="QueryResult">
<mgiAuthenticateDB mode="queryUser"
username={mgiPostArgument name="Username"}>
</mgiSet>

<mgiIf lhs={mgiGet name="QueryResult"}
relationship="equals" rhs="No">
```

```
<mgiAuthenticateDB mode="AddUser"
username={mgiPostArgument name="Username"}
password={mgiPostArgument name="Password"} group="Subscriber"
emailAddress={mgiPostArgument name="Email"}>

Thank you.  Your username and password are now active.

<mgiElse>

Error: Your username already exists.  Please use the
"Back" button on your browser and choose another username.

</mgiIf>
```

The addUser mode of mgiAuthenticateDB is useful for adding new users from a form submission. In this example, users entered a desired username, password, amd email address in a form that posts to the page containing these mgiAuthenticateDB tags. The username is first verified to insure that it is unique using the queryUser mode of mgiAuthenticateDB. If it is unique, the username is added to the authentication database, otherwise an error message is displayed.

Return to the mgiAuthenticateDB Main Menu

The mgiAuthenticateDB Tag

DeleteUser Mode

Return to the mgiAuthenticateDB Main Menu

Tag Syntax

The DeleteUser mode of mgiAuthenticateDB has two required parameters and three optional parameters. The tag form is:

```
<mgiAuthenticateDB mode="DeleteUser" username="Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiAuthenticateDB tag performs. In "**deleteUser**" mode, the mgiAuthenticateDB tag deletes a new user's authentication information from the authentication database.
- **username** - The username is the unique login (i.e., name) to be deleted. The username is case-sensitive.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, authentication information will be deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Example Usage and Output

```
<mgiAuthenticateDB mode="DeleteUser"
username={mgiPostArgument name="Username"}>
```

The deleteUser mode of mgiAuthenticateDB can be used in conjunction with a form submission to delete a user from the authentication database. In this example, the username is posted from a form submission and embedded in the username parameter of the mgiAuthenticateDB tag.

Return to the mgiAuthenticateDB Main Menu

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiAuthenticateDB Tag

QueryUser Mode

Return to the mgiAuthenticateDB Main Menu

Tag Syntax

The QueryUser mode of mgiAuthenticateDB has two required parameters and three optional parameters. The tag form is:

```
<mgiAuthenticateDB mode="QueryUser" username="Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiAuthenticateDB tag performs. In "**queryUser**" mode, the mgiAuthenticateDB tag checks the authentication database for the existence of the specified username in any group. If the username exists in any group, the value "**Yes**" is displayed. If the username does not exist, the value "**No**" is displayed.
- **username** - The username is the unique login (i.e., name) to check. The username is case-sensitive.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, authentication information will be queried from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Example Usage and Output

```
<mgiSet name="QueryResult">
<mgiAuthenticateDB mode="queryUser"
username={mgiPostArgument name="Username"}>
</mgiSet>

<mgiIf lhs={mgiGet name="QueryResult"}
relationship="equals" rhs="No">

<mgiAuthenticateDB mode="AddUser"
username={mgiPostArgument name="Username"}
password={mgiPostArgument name="Password"} group="Subscriber"
emailAddress={mgiPostArgument name="Email"}>

Thank you.  Your username and password are now active.

<mgiElse>

Error: Your username already exists.  Please use
the "Back" button on your browser and choose another username.

</mgiIf>
```

The queryUser mode of mgiAuthenticateDB checks the uniqueness of a username against existing usernames in the authentication database. In this example, users entered a desired username, password, and email address in a form that posts to the page containing these mgiAuthenticateDB tags. The username is first verified to insure that it is unique using the queryUser mode of mgiAuthenticateDB. If it is unique, the username is added to the authentication database using the addUser mode of mgiAuthenticateDB, otherwise an error message is displayed.

Return to the mgiAuthenticateDB Main Menu

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiAuthenticateDB Tag

ChangePassword Mode

Return to the mgiAuthenticateDB Main Menu

Tag Syntax

The ChangePassword mode of mgiAuthenticateDB has four required parameters and three optional parameters. The tag form is:

```
<mgiAuthenticateDB mode="ChangePassword"
username="Name" password="Code" newPassword="New Code"
odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiAuthenticateDB tag performs. In "**changePassword**" mode, the mgiAuthenticateDB tag replaces the user's current password with a new password.
- **username** - The username is the unique login (i.e., name) of the user who wishes to change their password. The username is case-sensitive.
- **password** - The password is the user's original security code. The password is case-sensitive.
- **newPassword** - The newPassword is the user's new security code. The new password is case-sensitive and can consist of letters, numbers, multiple words, spaces and ASCII characters. A secure password should contain at least 8 characters and include both numbers and letters.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, authentication information will be queried from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.

- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Example Usage and Output

```
<mgiAuthenticateDB mode="changePassword"
username={mgiPostArgument name="Login"}
password={mgiPostArgument name="OldPW"}
newPassword={mgiPostArgument name="NewPW"}>
```

The changePassword mode of mgiAuthenticateDB allows users to modify their password using a simple form submission. In this example, the user entered their username, existing password and new password into a form. Those values were posted to the page containing this mgiAuthenticateDB tag and embedded in the parameters of the mgiAuthenticateDB tag.

Return to the mgiAuthenticateDB Main Menu

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiAuthenticateDB Tag

SendPassword Mode

Return to the mgiAuthenticateDB Main Menu

Tag Syntax

The SendPassword mode of mgiAuthenticateDB has a beginning tag with five required parameters and four optional parameters, a body with placeholders for the username and password, and an ending mgiAuthenticateDB tag. The tag form is:

```
<mgiAuthenticateDB mode="SendPassword" emailAddress="Email"
from="Email" mailServer="SMTP Server" subject="Title"
message="Message with Placeholders"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiAuthenticateDB tag performs. In "**sendPassword**" mode, the mgiAuthenticateDB tag matches the user's email address to a record in the authentication database and emails the user's username and/or password in a custom email.
- **emailAddress** - The user's email address as it is listed in the authentication database.
- **from** - The email address listed in the "From" line of the email.
- **mailServer** - The name of the SMTP Outgoing mail server used to send the email (e.g., "mail.domain.com").
- **message** - The message that contains the username and password associated with the specified email. The username and password are automatially entered in the location of the placeholders (see below).

Optional Parameters:

- **subject** - The subject of the password email. The default subject is "Forgot your password?".
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, authentication information will be queried from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If

you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Placeholders:

- **&mgiAuthenticateDBUsername;** - When the password email is processed by MGI, the username placeholder (&mgiAuthenticateDBUsername;) is replaced with the specific user's username in the body of the email.
- **&mgiAuthenticateDBPassword;** - When the password email is processed by MGI, the password placeholder (&mgiAuthenticateDBPassword;) is replaced with the specific user's password in the body of the email.

Example Usage and Output

```
<mgiSet name="MessageText">
Dear Subscriber,

Your request for a copy of your username and
password is complete.

Username: &mgiAuthenticateDBUsername;
Password: &mgiAuthenticateDBPassword;

If you did not request this information, please
contact us immediately.
</mgiSet>

<mgiAuthenticateDB mode="sendPassword"
emailAddress={mgiPostArgument name="Email"}
from="registration@domain.com" mailServer="mail.domain.com"
subject="Registration Information"
message={mgiGet name="MessageText"}>
```

The sendPassword mode of mgiAuthenticateDB allows users to retrieve their username and password via email. The email entered by the user must match the email in the authentication database. In this example, the user's email was entered into a form, posted to the page containing this mgiAuthenticateDB tag and embedded in the emailAddress parameter. If the user's username is "bhjohnson" and the user's password is "pur2sd9", the following email would be generated:

```
Dear Subscriber,

Your request for a copy of your username and
password is complete.

Username: bhjohnson
Password: pur2sd9

If you did not request this information, please
contact us immediately.
```

Return to the mgiAuthenticateDB Main Menu

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiBannerAd Tag

Tag Behavior

Use the mgiBannerAd tag to randomly or sequentially display banner ads or other HTML, MGI and text information. Use the web-based administration interface to manage the banner ad information.

Tag Syntax

The mgiBannerAd tag has three modes. Each mode has different required and optional parameters. The modes of mgiBannerAd are:

- **display** - Displays banner ads.
- **admin** - Creates a web-based interface to manage banner ads.
- **clientStats** - Displays statistics for specified banner ad clients.

Display Mode

The display mode of mgiBannerAd has no required parameters and six optional parameters. The tag form is:

```
<mgiBannerAd mode="display" group="name" rotation="Type"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **None**.

Optional Parameters:

- **mode** - The mode is the function that the mgiBannerAd tag performs. In "**display**" mode, the mgiBannerAd tag rotates the display of banner ads. "Display" is the default mode of the mgiBannerAd tag.
- **group** - The group is the name of the banner ad group from which banners are selected

for display. If the group parameter is not included, banner ads will be selected from all groups.

- **rotation** - The rotation determines the algorithm used to select banner ads for display. If the rotation parameter value is "**random**", then banner ads are randomly selected for display. If the rotation parameter value is "**sequential**", then banner ads are displayed in the order they appear in the banner ad database. The default value is "random".
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, banner ad information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Admin Mode

The admin mode of mgiBannerAd has one required parameter and four optional parameters. The tag form is:

```
<mgiBannerAd mode="admin" advancedSearch="On/Off"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiBannerAd tag performs. In "**admin**" mode, the mgiBannerAd tag creates a web-based interface that allows you to add new banner ads, modify banner ads, and view banner ad statistics.

Optional Parameters:

- **advancedSearch** - The advancedSearch parameter determines whether the advanced search feature is available in the web-based administration interface. The advanced search feature allows you to enter complex search strings that are not available with the built-in search functions (e.g., "NOT" searches, "OR" searches, searching the same field multiple times, etc.). The search time will increase as the search complexity increases. If the advancedSearch parameter value is "**On**", then the advanced search field is displayed in the search screen of the admin interface. If the advancedSearch parameter value is "**Off**", then the advanced search field is not displayed in the search screen of the admin interface. The default value is "Off".

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, banner ad information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

ClientStats Mode

The clientStats mode of mgiBannerAd has two required parameter and three optional parameters. The tag form is:

```
<mgiBannerAd mode="clientStats" client="Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiBannerAd tag performs. In "**clientStats**" mode, the mgiBannerAd tag displays impression, clickthrough and date statistics for the specified client.
- **client** - The client is the name of the client whose statistics are displayed.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, banner ad information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.

- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Example Usage and Output

Display Mode

```
<mgiBannerAd group="top">
```

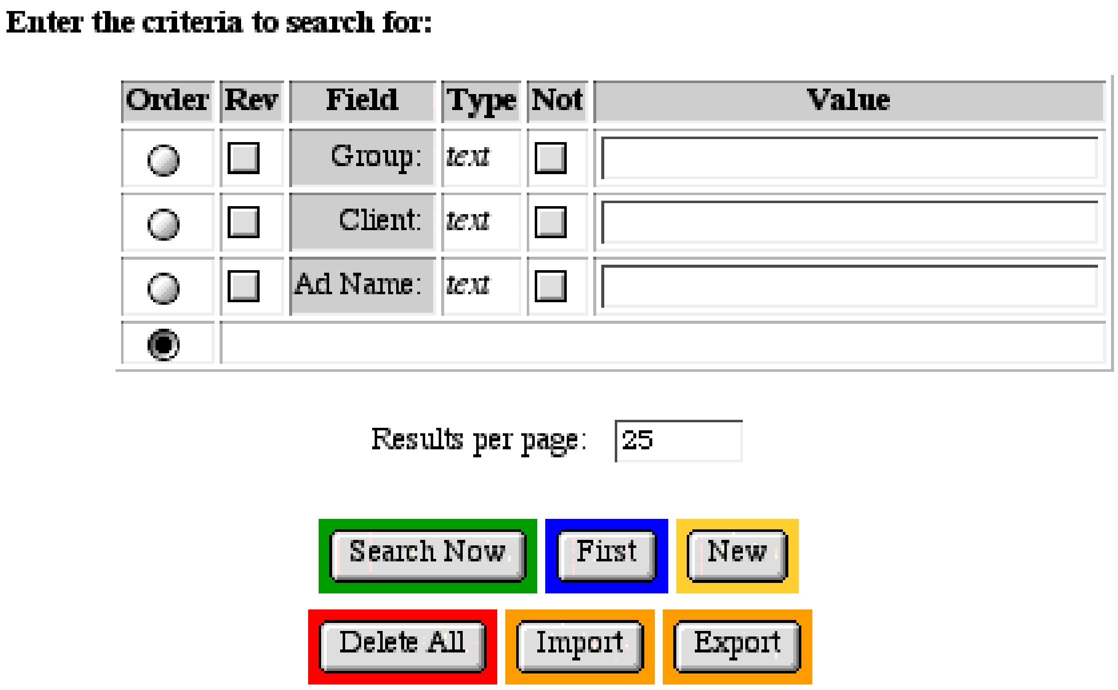
In this example, a banner ad from the group "top" are displayed randomly each time the page is accessed. The display of the individual banner ad is based on the HTML entered for the banner ad record.

Admin Mode

```
<mgiBannerAd mode="admin">
```

When you access a page with the mgiBannerAd tag in admin mode, buttons for creating a new banner and importing existing banner ads display.

The following is the main interface to mgiBannerAd in admin mode:

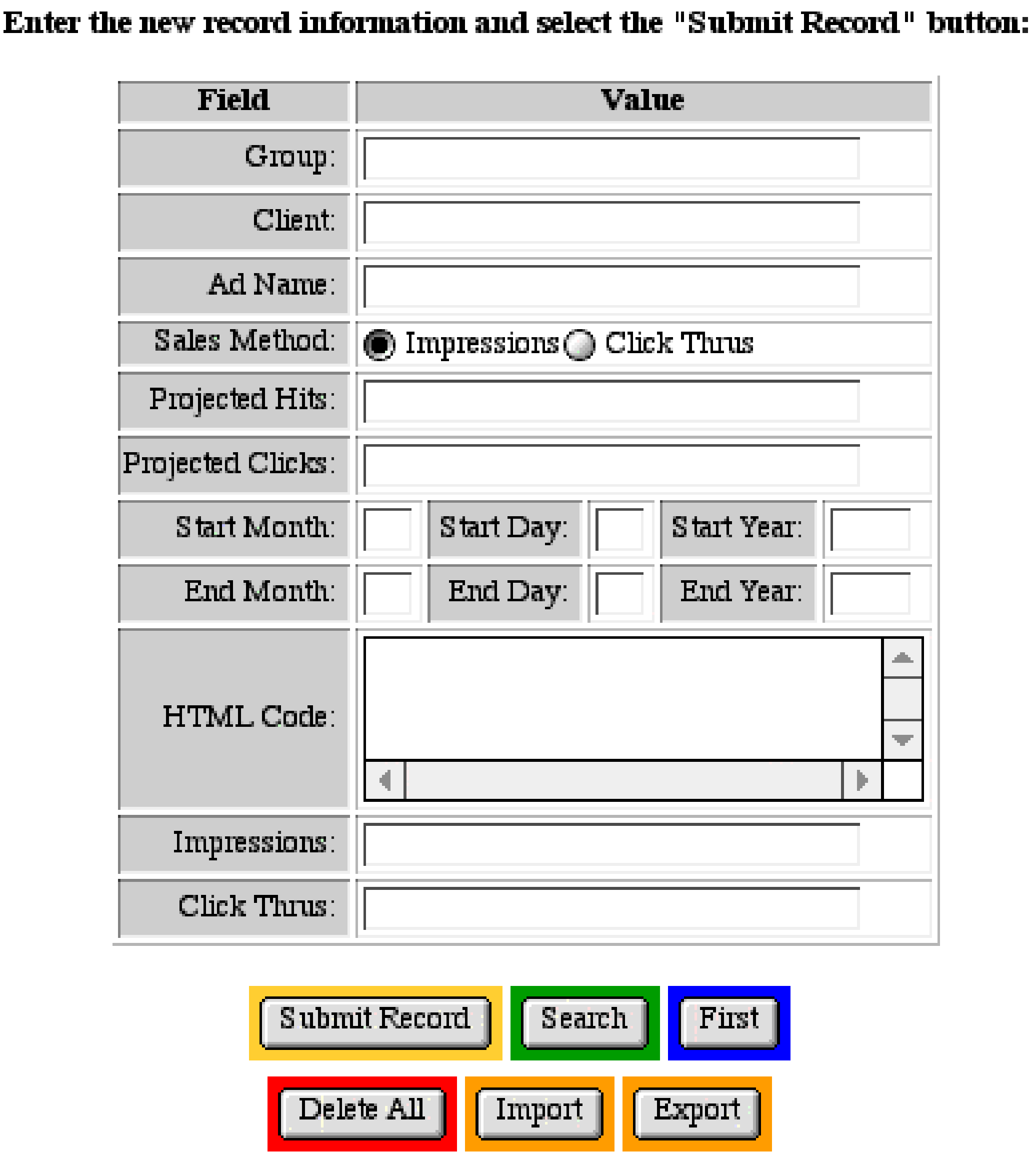


Enter the criteria to search for:

Order	Rev	Field	Type	Not	Value
○	□	Group:	*text*	□	
○	□	Client:	*text*	□	
○	□	Ad Name:	*text*	□	
●					

Results per page: 25

The following is the interface to add a new banner ad in the admin mode of mgiBannerAd:



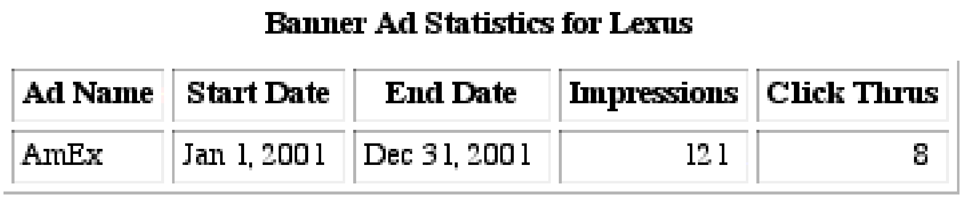
Enter the new record information and select the "Submit Record" button:

Field	Value				
Group:					
Client:					
Ad Name:					
Sales Method:	● Impressions ○ Click Thrus				
Projected Hits:					
Projected Clicks:					
Start Month:		Start Day:		Start Year:	
End Month:		End Day:		End Year:	
HTML Code:					
Impressions:					
Click Thrus:					

ClientStats Mode

```
<mgiBannerAd mode="clientStats" client="Lexus">
```

In this example, impression and click-through statistics for all of Lexus' banner ads are displayed in a table.



Banner Ad Statistics for Lexus

Ad Name	Start Date	End Date	Impressions	Click Thrus
AmEx	Jan 1, 2001	Dec 31, 2001	121	8

Suggested Usage

- Banner Ad Display

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiButton Tag

Tag Behavior

Use the mgiButton tag to create a submit button with a specified name, or an image to be use as a submit button. The mgiButton tag must be used in conjunction with HTML <FORM> tags to work in form submissions.

Tag Syntax

The mgiButton tag has no required parameters and three optional parameters. The tag form is:

```
<mgiButton name="Name" value="Value" imageLocation="Image">
```

Required Parameters:

- **None.**

Optional Parameters:

- **name -** The name is the button's unique post argument name. The default name is "Submit".
- **value -** The value is the text displays on the submit button or the value of the image button. The default value is "Submit".
- **imageLocation -** The imageLocation is the relative or absolute path to the image to use as a submit button. If the imageLocation parameter is not included, a standard submit button is displayed.

Example Usage and Output

```
<form action="shoppingbasket.mgi" method="post">
<mgiButton name="ViewSB" value="View Shopping Basket">
</form>
```

In this example, a standard submit button is created and links to a shopping basket page.

```
<form action="https://secure.domain.com/domain/checkout.mgi"
method="post">
<mgiButton name="CheckOut" imageLocation="Images/b_checkout.jpg">
```

```
</form>
```

In this example, a custom button is used to link customers to the secure check out page.

Suggested Usage

- Form Processing
- Shopping Basket
- Database Searching

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiBuyMe Tag

Tag Behavior

Use the mgiBuyMe tag to add a product to the shopping basket. The mgiBuyMe tag is used in conjunction with products hard-coded on a page or with products from a database.

Tag Syntax

The mgiBuyMe tag has one required parameter and 20 optional parameters. The tag form is:

```
<mgiBuyMe productID="Product ID" name="Product Name"
description="Product Description" price="Product Price"
shipping="Product Shipping Price" quantityMultiplier="integer"
qualifierPostArgument1="Post Argument Name"
qualifierPostArgument2="Post Argument Name"
qualifierPostArgument3="Post Argument Name"
qualifierPostArgument4="Post Argument Name"
qualifierPostArgument5="Post Argument Name"
productDB="Database Name" inventoryFieldName="Field Name"
productIDFieldName="Field Name" weight="Product Weight"
priceRule="Price Rule Name" shippingRule="Shipping Rule Name"
taxRule="Tax Rule Name" defaultValue="Integer"
mode="Visible/Hidden/Checkbox" size="Integer">
```

Required Parameters:

- **productID** - The product ID is the unique identification code of the product.

Optional Parameters:

- **name** - The name is the short description or title of the product. In the automatic shopping basket configuration, the name appears as the product description in the shopping basket.
- **description** - The description is the long description of the product.
- **price** - The price is the price of the product. (see also priceRule parameter below).
- **shipping** - The shipping is the shipping price of the product. (see also shippingRule parameter below).
- **quantityMultiplier** - The quantityMultiplier is the integer to multiply by the quantity

entered to get the actual quantity of products being purchased. The quantityMultiplier parameter is useful for products only sold in multiple quantities.

- **qualifierPostArgument1 -** The qualifierPostArgument1 is the name of a form field (i.e., a post argument name for a text box, popup menu, etc.) that contains a qualifying characteristic of the product. For example, if you have a popup menu of product colors, enter the name of that popup menu as a qualifierPostArgument.
- **qualifierPostArgument2 -** The qualifierPostArgument2 is the name of a form field that contains a qualifying characteristic of the product.
- **qualifierPostArgument3 -** The qualifierPostArgument3 is the name of a form field that contains a qualifying characteristic of the product.
- **qualifierPostArgument4 -** The qualifierPostArgument4 is the name of a form field that contains a qualifying characteristic of the product.
- **qualifierPostArgument5 -** The qualifierPostArgument5 is the name of a form field that contains a qualifying characteristic of the product.
- **productDB -** The productDB is the name of the database that contains the product information for inventory control.
- **inventoryFieldName -** The inventoryFieldName is the name of the database field that contains inventory values.
- **productIDFieldName -** The productIDFieldName is the name of the database field that contains the unique product identification codes for inventory control.
- **weight -** The weight is the weight of the product. The weight may be a decimal number.
- **priceRule -** The priceRule is the name of the price algorithm applied to the product. Price rules must have a scope of "item" to be applied in the mgiBuyMe tag. Price rules are defined in the administration interface of the mgiShoppingBasket tag.
- **shippingRule -** The shippingRule is the name of the shipping algorithm applied to the product. Shipping rules must have a scope of "item" to be applied in the mgiBuyMe tag. Shipping rules are defined in the administration interface of the mgiShoppingBasket tag.
- **taxRule -** The taxRule is the name of the tax algorithm applied to the product. Tax rules must have a scope of "item" to be applied in the mgiBuyMe tag. Tax rules are defined in the administration interface of the mgiShoppingBasket tag.
- **defaultValue -** The defaultValue is the default quantity of products ordered.
- **mode -** The mode determines the display of the quantity box (text field) created by the mgiBuyMe tag. If the mode parameter value is "**Visible**", then the mgiBuyMe tag displays a visible quantity box (text field) input. If the mode parameter value is "**Hidden**", then the mgiBuyMe tag creates a hidden quantity box (text field) input. Hidden quantities may still be added to a shopping basket. If the mode parameter is "**Checkbox**", then the mgiBuyMe tag displays a checkbox with a value equal to 1 unless the defaultValue parameter is present to override it. The default value is "Visible".
- **size -** The size is the length of the quantity box (text field) that is created by the mgiBuyMe tag. The size of a quantity box determines the number of characters that can

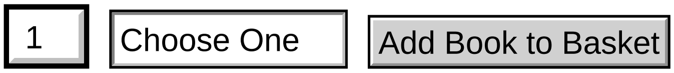
display. The default value is "3".

Example Usage and Output

Hard-Coded

```
<mgiToken>

<form action="shoppingbasket.mgi" method="post">

<mgiBuyMe productID="0-395-75283-3" name="Smart Eating"
description="Smart Eating, Choosing Wisely, Living Lean.
By Covert Bailey and Ronda Gates" price="6.99"
qualifierPostArgument1="Cover Type"
defaultValue="1">

<select name="Cover Type">
<option value="">Choose One
<option>Hard-Cover
<option>Soft-Cover
</select>

<mgiButton name="Add Book to Basket">

</form>

</mgiToken>
```

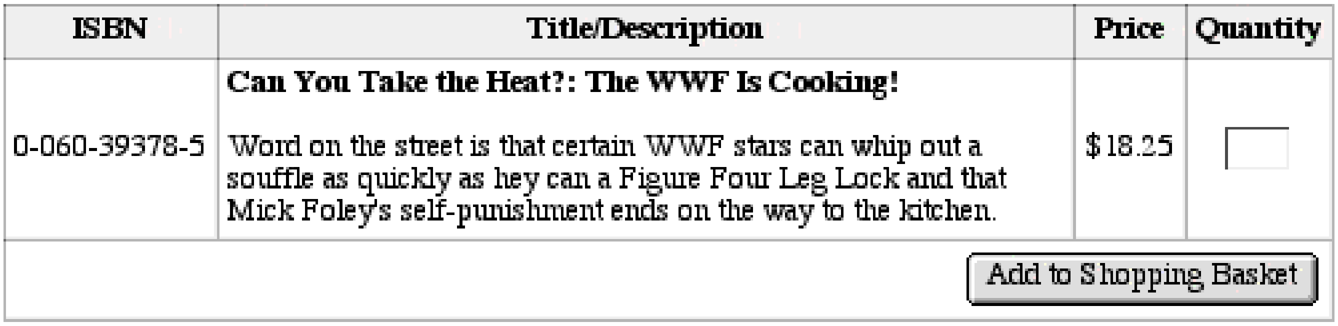
In this example, the information in the parameters of the mgiBuyMe tag is hard-coded directly into the tag. This mgiBuyMe tag would display a quantity box with the default value of 1 displayed. When this product is added to the shopping basket, the productID, name, description, price, and cover type is stored for this product.



Database Results

```
<mgiToken>

<p>
<table width="500" cellspacing="0" cellpadding="3" border="1">
<tr bgcolor="#eeeeee">
  <th>ISBN</th>
  <th>Title/Description</th>
  <th>Price</th>
  <th>Quantity</th>
</tr>

<mgiSearchDatabase databaseName="Products"
keyFieldName="ProductName" fieldValue="*"
orderByField="ProductID" resultsPerPage="20">

<tr>
  <td>&mgiDBFieldProductID;</td>
  <td><b>&mgiDBFieldProductName;</b>
<p>&mgiDBFieldProductDescription;</td>
  <td align="right">$&mgiDBFieldProductPrice;</td>
  <td align="center">
<mgiBuyMe productID="&mgiDBFieldProductID;"
name="&mgiDBFieldProductName;" price="&mgiDBFieldProductPrice;">
</td>
</tr>

</mgiSearchDatabase>

<tr>
  <td colspan="4" align="right">
<mgiButton value="Add to Shopping Basket"></td>
</tr>
</table>

</mgiToken>
```

In this example, the information in the parameters of the mgiBuyMe tag is provided by placeholders in the results of a database search. This mgiBuyMe tag displays a quantity box. When this product is added to the shopping basket, the productID, name, description, and price is stored for this product.



ISBN	Title/Description	Price	Quantity
0-060-39378-5	**Can You Take the Heat?: The WWF Is Cooking!** Word on the street is that certain WWF stars can whip out a souffle as quickly as hey can a Figure Four Leg Lock and that Mick Foley's self-punishment ends on the way to the kitchen.	$18.25	□

Suggested Usage

- Shopping Basket

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiCloak Tag

Tag Behavior

Use the mgiCloak tag to display information only to visitors specified in the client list. The mgiCloak tag is useful for dynamically displaying search engine keywords to search engine clients without displaying them to other web site visitors.

Tag Syntax

The mgiCloak tag has a beginning tag requiring at least one of the two optional parametrers, a body, and an ending tag. The tag form is:

```
<mgiCloak allowedIP="IP Numbers" clientListLocation="File Path">
Information displayed to IP or Client matches
</mgiCloak>
```

Required Parameters:

- **At least one optional parameter is required.**

Optional Parameters:

- **allowedIP -** The allowedIP is the IP number or client address of a visitor that is allowed to view the information in the body of the mgiCloak tag. To include multiple IP numbers, enter a comma-delimited list of IP numbers in one allowedIP parameter OR include multiple allowedIP parameters in the mgiCloak tag. The allowedIP parameter may contain wildcard values using an asterisk (*) (e.g., 215.233.144.*).
- **clientListLocation -** The clientListLocation is the relative path to the file that contains a list of IP numbers and/or client addresses that are allowed to view the information in the body of the mgiCloak tag. Wildcard values are not allowed in the client list.

Example Usage and Output

```
<mgiCloak clientListLocation="clients.txt">
Apparel, Clothing, Scarves, Hats, Winter Coats, S
weaters, Boots, Gloves, Socks, Shirts, Jeans
</mgiCloak>
```

In this example, search engine clients listed in the "clients.txt" file will be shown the keywords listed in the body of mgiCloak.

Suggested Usage

- Search Engine Keyword Placement
- Intranet Data Security

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiCollectUserInfo Tag

Tag Behavior

Use the mgiCollectUserInfo tag to collect payment, billing and/or shipping information about a visitor during the shopping basket checkout process.

Tag Syntax

The mgiCollectUserInfo tag has two required parameters and three optional parameters. The tag form is:

```
<mgiCollectUserInfo handle="Configuration Name"
shoppingBasketURL="URL" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **handle -** The handle is the name of the shopping basket configuration to use. Shopping basket configurations are defined using the admin mode of the mgiShoppingBasket tag.
- **shoppingBasketURL -** The shoppingBasketURL is the absolute address to the folder region containing the visitor's shopping basket (i.e., the region containing the mgiShoppingBasket tag that created the visitor's shopping basket).

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, shopping basket information will be retrieved from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Technical Notes

- **Default Payment, Billing and Shipping Post Arguments** - The mgiCollectUserInfo tag displays tables to collect payment, billing, and shipping information. The post argument names in the tables of payment, billing and shipping information are as follows. Note: these post argument names are only created in the Automatic and Selective shopping basket methods (chosen in the mgiShoppingBasket admin).
    - **PaymentMethod** - Shopping basket payment method. Values include "Credit Card", "Purchase Order", and "Check".
    - **CardType** - Credit card type. Values include "Visa", "Mastercard", "Disover/Novus", "American Express", "Carte Blanche", "Diner's Club", "Bankcard", and/or "JCB" depending on shopping basket configuration.
    - **CCNumber** - Credit card number.
    - **ExpMo** - Credit card expiration month.
    - **ExpYe** - Credit card expiration year.
    - **CCName** - Name that appears on credit card.
    - **PONumber** - Purchase order number.
    - **POCompany** - Purchase order company.
    - **BName** - Billing name
    - **BCompany** - Billing company
    - **BAddress1** - Billing address
    - **BCity** - Billing city
    - **BState** - Billing state
    - **BProvince** - Billing province
    - **BZipCode** - Billing zip code
    - **BCountry** - Billing country
    - **BPhone** - Billing phone
    - **BFax** - Billing fax
    - **BEmail** - Billing email
    - **SCompany** - Shipping company
    - **SAddress1** - Shipping address
    - **SCity** - Shipping city
    - **SState** - Shipping state
    - **SProvince** - Shipping province
    - **SZipCode** - Shipping zip code

- **SCountry** - Shipping country
- **SPhone** - Shipping phone
- **SFax** - Shipping fax
- **SEmail** - Shipping email

Example Usage and Output

Automatic and Selective Forms

```
<form action="confirm.mgi" method="post">

<mgiCollectUserInfo handle="Books"
shoppingBasketURL="http://www.domain.com/store/">

<mgiButton name="Confirm Order">

</form>
```

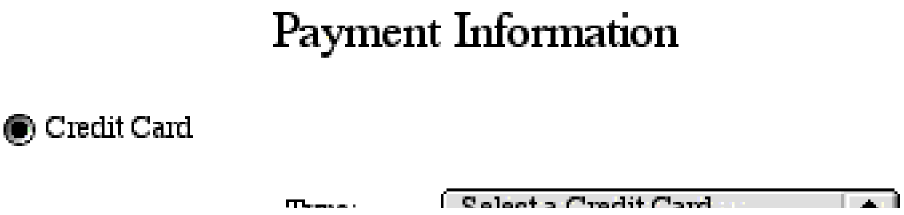
The mgiCollectUserInfo tag displays tables of customer payment, billing, and shipping information with the options defined in the "Books" configuration.

The options for the display of information appear in the Customer Information options in the admin mode of mgiShoppingBasket. In Automatic mode, all payment, billing and shipping information fields are displayed and fields are required based on whether they apply to all customers and are required to deliver an order. In Selective mode, only selected payment, billing and shipping information fields are displayed and only selected fields are required.

It is possible to use a custom form layout in the Automatic or Selective modes of the shopping basket. If you choose the Automatic or Selective shopping basket method in the mgiShoppingBasket admin AND if you maintain the field names (post argument names - listed above) of the Automatic/Selective payment, billing, and shipping forms, custom forms do not require custom mgiConfirmOrder and mgiSendOrder tags.

In Manual mode, the mgiCollectUserInfo tag should not be used. Rather, you should construct a custom form to collect payment, billing and shipping information. See below for more information about custom forms.

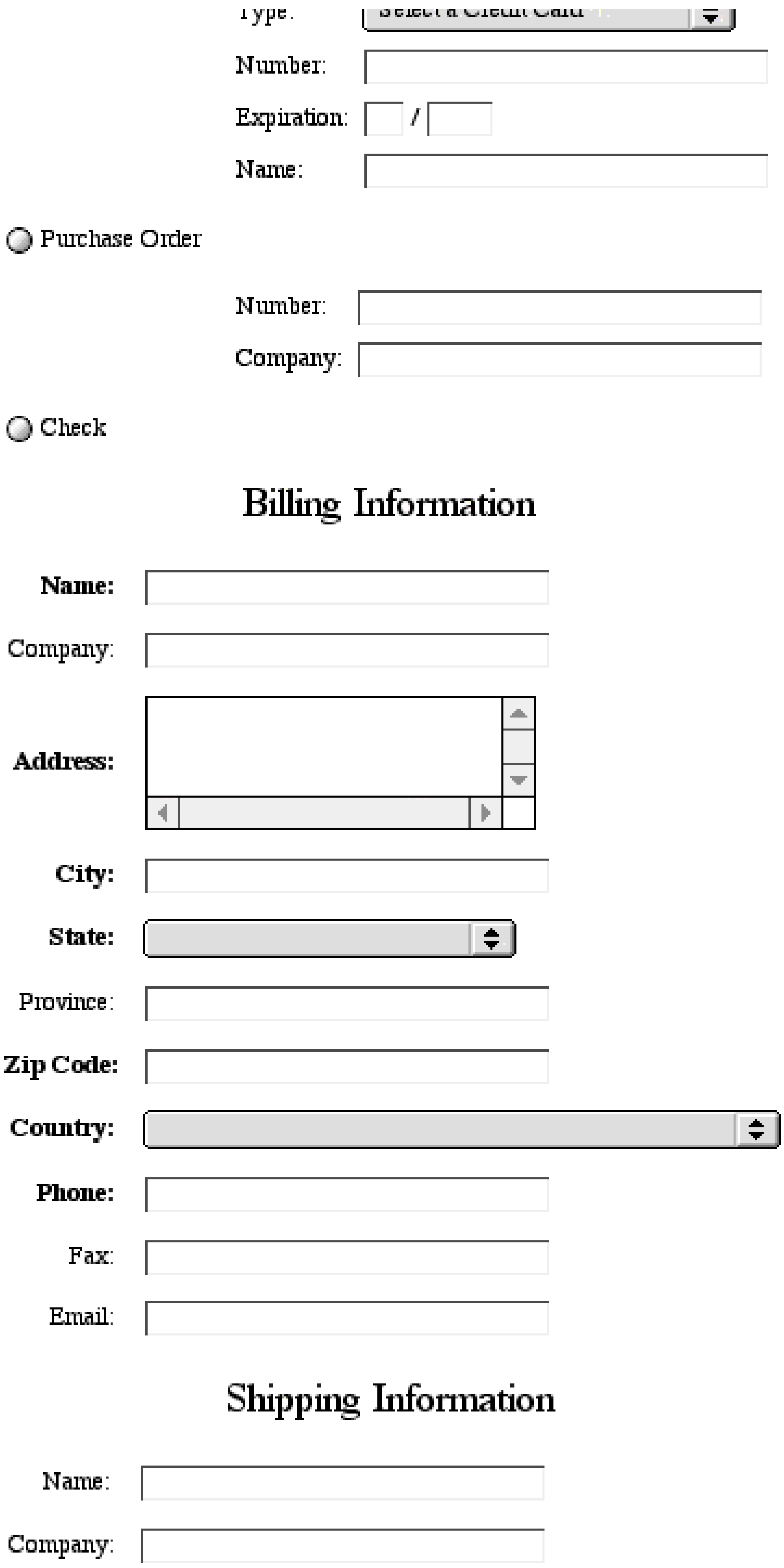
The default format of the payment, billing and shipping information tables follows:



Payment Information

● Credit Card

Type: Select a Credit Card



Type: Select a Credit Card

Number:

Expiration: /

Name:

○ Purchase Order

Number:

Company:

○ Check

Billing Information

Name:

Company:

Address:

City:

State:

Province:

Zip Code:

Country:

Phone:

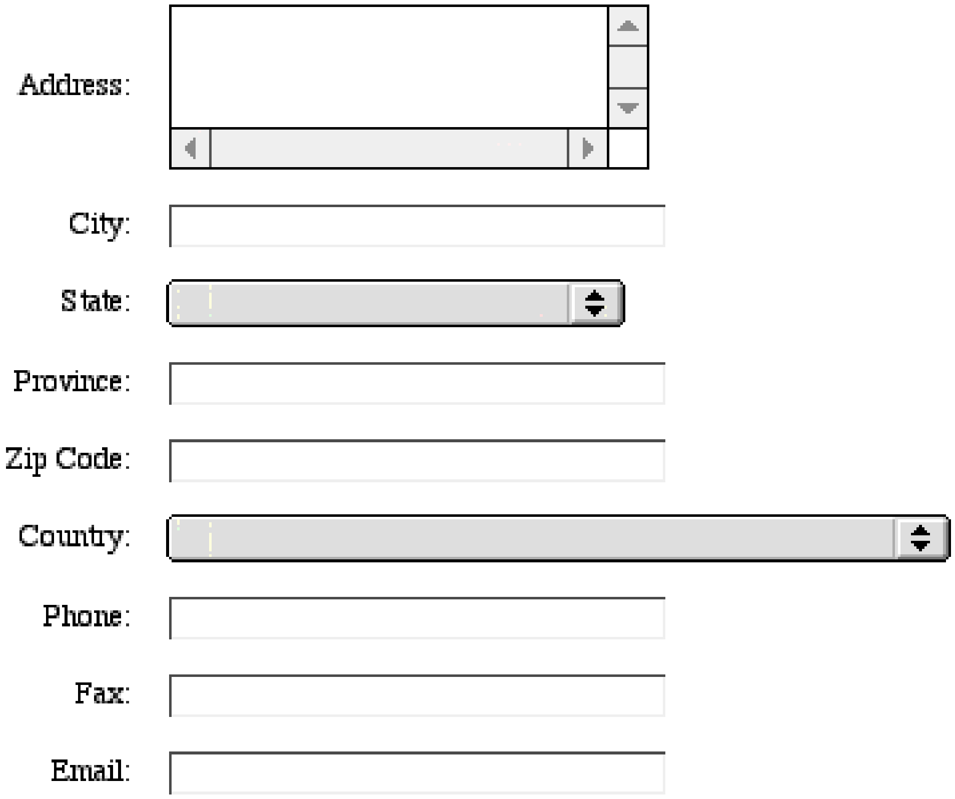
Fax:

Email:

Shipping Information

Name:

Company:



Address:

City:

State:

Province:

Zip Code:

Country:

Phone:

Fax:

Email:

Manual Form

```
<mgitoken>

<FORM ACTION="confirmorder2.mgi" METHOD="POST">

<P><CENTER>
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0"
WIDTH="525">
<TR HEIGHT="375">
<TD WIDTH="300" HEIGHT="375" VALIGN="TOP" ALIGN="CENTER">
<P>

<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="3"
WIDTH="250">
<TR HEIGHT="30">
<TD HEIGHT="30" COLSPAN="2"><B>BILLING ADDRESS</B></TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">Name*</FONT></B>
</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BName" SIZE="27">
</TD>
```

```
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Company</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BCompany" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">Address*</FONT></B>
</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BAddress1" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">City*</FONT></B>
</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BCity" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">State*</FONT></B>
</TD>
<TD WIDTH="175"><SELECT name="BSTATE">
<OPTION>
<OPTION>Alabama
<OPTION>Alaska
<OPTION>Arizona
<OPTION>Arkansas
<OPTION>California
<OPTION>Colorado
<OPTION>Connecticut
<OPTION>Delaware
<OPTION>District of Columbia
<OPTION>Florida
<OPTION>Georgia
<OPTION>Hawaii
<OPTION>Idaho
<OPTION>Illinois
<OPTION>Indiana
<OPTION>Iowa
<OPTION>Kansas
<OPTION>Kentucky
```

```
<OPTION>Louisiana
<OPTION>Maine
<OPTION>Maryland
<OPTION>Massachusetts
<OPTION>Michigan
<OPTION>Minnesota
<OPTION>Mississippi
<OPTION>Missouri
<OPTION>Montana
<OPTION>Nebraska
<OPTION>Nevada
<OPTION>New Hampshire
<OPTION>New Jersey
<OPTION>New Mexico
<OPTION>New York
<OPTION>North Carolina
<OPTION>North Dakota
<OPTION>Ohio
<OPTION>Oklahoma
<OPTION>Oregon
<OPTION>Pennsylvania
<OPTION>Rhode Island
<OPTION>South Carolina
<OPTION>South Dakota
<OPTION>Tennessee
<OPTION>Texas
<OPTION>Utah
<OPTION>Vermont
<OPTION>Virginia
<OPTION>Washington
<OPTION>West Virginia
<OPTION>Wisconsin
<OPTION>Wyoming
</SELECT></TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Province</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BProvince" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">Zip Code*</FONT></B>
</TD>
```

```
<TD WIDTH="175"><INPUT TYPE="text" NAME="BZipCode" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Country</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BCountry" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">Phone*</FONT></B>
</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BPhone" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Fax</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BFax" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT><B><FONT COLOR="#ff0000">E-mail*</FONT></B>
</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="BEmail" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75"></TD>
<TD WIDTH="175"></TD>
</TR>
</TABLE></P>
<P>
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="3" WIDTH="260">
<TR HEIGHT="25">
<TD HEIGHT="25" COLSPAN="2"><B>PAYMENT INFORMATION</B></TD>
</TR>
<TR HEIGHT="20">
<TD HEIGHT="20" COLSPAN="2"><INPUT TYPE="radio"
VALUE="Credit Card" NAME="PaymentMethod"> <B>Credit Card</B>
</TD>
</TR>
<TR>
```

```
<TD WIDTH="75">
<P ALIGN=RIGHT>Type</TD>
<TD WIDTH="176">
<SELECT NAME="CardType">
<OPTION VALUE="" SELECTED>Select a Card Type
<OPTION>Visa
<OPTION>MasterCard
<OPTION>Discover/Novus
</SELECT></TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Number</TD>
<TD WIDTH="176"><INPUT TYPE="text" NAME="CCNumber" SIZE="26">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Expiration</TD>
<TD WIDTH="176"><INPUT TYPE="text" NAME="EXPMO" SIZE="4"> /
<INPUT TYPE="text" NAME="EXPYE" SIZE="4"></TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Name</TD>
<TD WIDTH="176"><INPUT TYPE="text" NAME="CCName" SIZE="26"></TD>
</TR>
<TR HEIGHT="25">
<TD HEIGHT="25" COLSPAN="2"><INPUT TYPE="radio"
VALUE="Check" NAME="PaymentMethod"> <B>Check or Money Order</B>
</TD>
 </TR>
<TR>
<TD COLSPAN="2">&nbsp;</TD>
 </TR>
<TR>
<TD WIDTH="75"></TD>
<TD WIDTH="176"></TD>
</TR>
</TABLE>
</TD>
<TD WIDTH="300" HEIGHT="375" VALIGN="TOP" ALIGN="CENTER">
<P><TABLE BORDER="0" CELLPADDING="0" CELLSPACING="3"
WIDTH="250">
<TR HEIGHT="30">
```

```
<TD HEIGHT="30" COLSPAN="2"><B>SHIPPING ADDRESS
</B>(if different)</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Name</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SName" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Company</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SCompany" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Address</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SAddress1" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>City</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SCity" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>State</TD>
<TD WIDTH="175"><SELECT name="SSTATE">
<OPTION>
<OPTION>Alabama
<OPTION>Alaska
<OPTION>Arizona
<OPTION>Arkansas
<OPTION>California
<OPTION>Colorado
<OPTION>Connecticut
<OPTION>Delaware
<OPTION>District of Columbia
<OPTION>Florida
<OPTION>Georgia
<OPTION>Hawaii
<OPTION>Idaho
```

```
<OPTION>Illinois
<OPTION>Indiana
<OPTION>Iowa
<OPTION>Kansas
<OPTION>Kentucky
<OPTION>Louisiana
<OPTION>Maine
<OPTION>Maryland
<OPTION>Massachusetts
<OPTION>Michigan
<OPTION>Minnesota
<OPTION>Mississippi
<OPTION>Missouri
<OPTION>Montana
<OPTION>Nebraska
<OPTION>Nevada
<OPTION>New Hampshire
<OPTION>New Jersey
<OPTION>New Mexico
<OPTION>New York
<OPTION>North Carolina
<OPTION>North Dakota
<OPTION>Ohio
<OPTION>Oklahoma
<OPTION>Oregon
<OPTION>Pennsylvania
<OPTION>Rhode Island
<OPTION>South Carolina
<OPTION>South Dakota
<OPTION>Tennessee
<OPTION>Texas
<OPTION>Utah
<OPTION>Vermont
<OPTION>Virginia
<OPTION>Washington
<OPTION>West Virginia
<OPTION>Wisconsin
<OPTION>Wyoming
</SELECT></TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Province</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SProvince" SIZE="27">
</TD>
```

```
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Zip Code</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SZipCode" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Country</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SCountry" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Phone</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SPhone" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Fax</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SFax" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>E-mail</TD>
<TD WIDTH="175"><INPUT TYPE="text" NAME="SEmail" SIZE="27">
</TD>
</TR>
<TR>
<TD WIDTH="75"></TD>
<TD WIDTH="175"></TD>
</TR>
</TABLE></P>
<P><TABLE BORDER="0" CELLPADDING="0" CELLSPACING="3"
WIDTH="260">
<TR HEIGHT="25">
<TD HEIGHT="25" COLSPAN="2"><B>ADDITIONAL INFORMATION</B>
</TD>
</TR>
<TR>
<TD WIDTH="75">
<P ALIGN=RIGHT>Comments</TD>
```

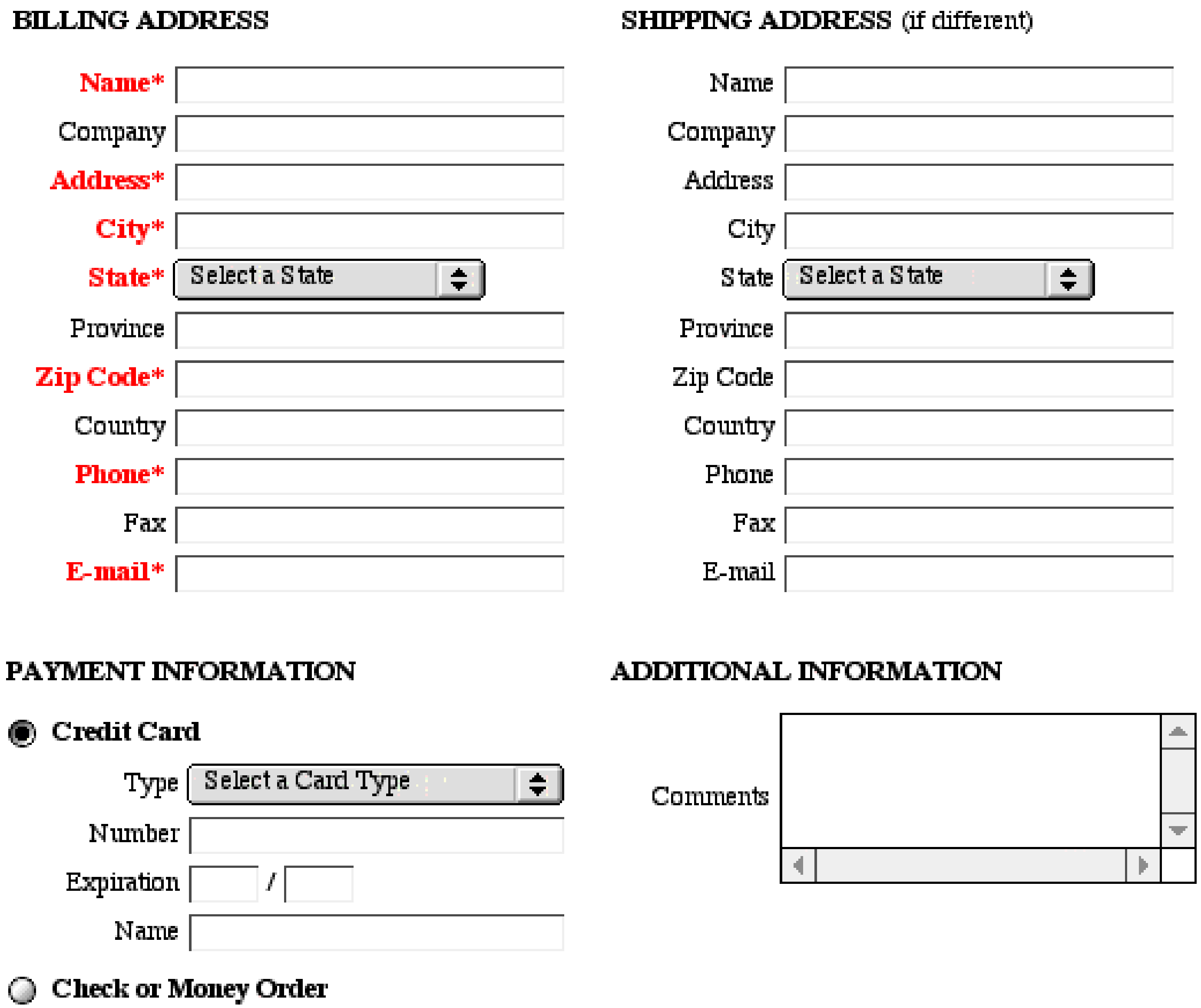
```
<TD WIDTH="176">
<TEXTAREA NAME="Comments" COLS="25" ROWS="4">
</TEXTAREA></TD>
</TR>
<TR>
<TD WIDTH="75"></TD>
<TD WIDTH="176"></TD>
</TR>
<TR>
<TD WIDTH="75"></TD>
<TD WIDTH="176"></TD>
</TR>
<TR>
<TD WIDTH="75"></TD>
<TD WIDTH="176"></TD>
</TR>
</TABLE>
</TD>
</TR>
<TR HEIGHT="40">
<TD HEIGHT="40" COLSPAN="2">
<P><CENTER><mgibutton value="Confirm Order"></CENTER></TD>
</TR>
</TABLE></CENTER></P>

</FORM>

</mgitoken>
```

The options for the display of information appear in the Customer Information options in the admin mode of mgiShoppingBasket.

If you choose the Automatic or Selective shopping basket method in the mgiShoppingBasket admin AND if you maintain the field names (post argument names - listed above) of the Automatic/Selective payment, billing, and shipping forms, then custom forms do not require custom mgiConfirmOrder and mgiSendOrder tags.

In Manual mode, the mgiCollectUserInfo tag should not be used. Rather, you should construct a custom form to collect payment, billing and shipping information. If you choose the Manual shopping basket method OR if you use any custom field names (in any mode) in your form, you must customize the mgiConfirmOrder and mgiSendOrder tags.

The following is the display of the custom form above to collect payment, billing, and shipping information.



BILLING ADDRESS

Name*
Company
Address*
City*
State* Select a State
Province
Zip Code*
Country
Phone*
Fax
E-mail*

SHIPPING ADDRESS (if different)

Name
Company
Address
City
State Select a State
Province
Zip Code
Country
Phone
Fax
E-mail

PAYMENT INFORMATION

● Credit Card

Type Select a Card Type
Number
Expiration /
Name

○ Check or Money Order

ADDITIONAL INFORMATION

Comments

Suggested Usage

- Shopping Basket

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiComment Tag

Tag Behavior

Use the mgiComment tag to hide comments or MGI code that you do not wish to be processed. Information in the body of an mgiComment tag is removed completely from the page and is therefore not available in the page source when it is served. (see also mgiDoNotProcess)

Tag Syntax

The mgiComment tag has a beginning tag with no required parameters and no optional parameters, a body, and an ending tag. The tag form is:

```
<mgiComment>
Text, MGI Code, etc.
</mgiComment>
```

Required Parameter:

- **None.**

Optional Parameters:

- **None.**

Example Usage and Output

```
<mgiComment>
Coder: ABSmith
Date Modified: 10-24-2000
Notes: Modified search display.
</mgiComment>
```

The mgiComment tag allows you to make notes about page design, functionality, and other page characteristics without the typical appearance of the comment in the HTML source. In this example, one coder is commenting a database search for others during group development of a site.

Suggested Usage

- Commenting Code
- Site Testing
- Group Site Development

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiConfirmOrder Tag

Tag Behavior

Use the mgiConfirmOrder tag to display a confirmation of the visitor's order, payment, billing, and shipping information during the shopping basket checkout process.

Tag Syntax

The mgiConfirmOrder tag has a beginning tag with two required parameters and eight optional parameters, a body, and an ending tag. The tag form is:

```
<mgiConfirmOrder handle="Configuration Name"
shoppingBasketURL="URL" priceRule="Price Rule Name"
shippingRule="Shipping Rule Name" taxRule="Tax Rule Name"
customerLocation="Location List" displayHiddenInfo="Yes/No"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
Custom Order Confirmation Product Template
</mgiConfirmOrder>
```

Required Parameters:

- **handle -** The handle is the name of the shopping basket configuration to use. Shopping basket configurations are defined using the admin mode of the mgiShoppingBasket tag.
- **shoppingBasketURL -** The shoppingBasketURL is the absolute address to the folder

region containing the visitor's shopping basket (i.e., the region containing the mgiShoppingBasket tag that created the visitor's shopping basket).

Optional Parameters:

- **priceRule** - The priceRule is the name of the price algorithm to apply to products in the final shopping basket confirmation. Price rules included in the mgiConfirmOrder tag must have a scope of "basket" in order to be applied to all products that are purchased. Price rules are defined in the admin mode of the mgiShoppingBasket tag.
- **shippingRule** - The shippingRule rule is the name of the shipping algorithm to apply to products in the final shopping basket confirmation. Shipping rules included in the mgiConfirmOrder tag must have a scope of "basket" in order to be applied to all products that are purchased. Shipping rules are defined in the admin mode of the mgiShoppingBasket tag.
- **taxRule** - The taxRule is the name of the tax algorithm to apply to products in the final shopping basket confirmation. Tax rules included in the mgiConfirmOrder tag must have a scope of "basket" in order to be applied to all products that are purchased. Tax rules are defined in the admin mode of the mgiShoppingBasket tag. If you include the taxRule parameter, the customerLocation parameter is required
- **customerLocation** - The customerLocation is a comma delimited list of the customer's location that is used to determine the customer's tax according to the tax rule. If there is a chain of tax rules, the first location in the list should correspond to the first tax rule, the second location in the list should correspond to the second tax rule, etc. If you include the taxRule parameter, the customerLocation parameter is required.
- **displayHiddenInfo** - The displayHiddenInfo parameter determines whether all post arguments posted to the page containing the mgiConfirmOrder tag are automatically created as hidden text inputs (to carry their values to the next page). If the displayHiddenInfo parameter value is "**Yes**", then the hidden text inputs are created. If the displayHiddenInfo parameter value is "**No**", then the hidden text inputs are not created. The default value is "Yes".
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, shopping basket information will be retrieved from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the

odbcDatasource parameter.

Technical Notes

- **Price, Shipping and Tax Rules -** Price, shipping and tax rules can be configured in the admin mode of mgiShoppingBasket.
  - Shopping basket rules are shared across all shopping basket handles within a region.
  - Only "basket" type rules may be applied in the mgiShoppingBasket tag. Apply "item" type rules in the mgiBuyMe tag.
- **Custom Shopping Basket Display -** The body of the mgiConfirmOrder tag contains a template display that is repeated for each product in the shopping basket. If the body of the mgiConfirmOrder tag is left empty, the default template will be used. To create a custom shopping basket display, use HTML, MGI, text , and the following placeholders in the body of the mgiConfirmOrder tag. The placeholders will be replaced with each product's specific information.
  - **&mgiDBFieldQuantity;** is the quantity of the product.
  - **&mgiDBFieldQuantityMultiplier;** is the quantity multiplier of the product.
  - **&mgiDBFieldProductID;** is the product ID.
  - **&mgiDBFieldName;** is the product name (short description).
  - **&mgiDBFieldDescription;** is the product description
  - **&mgiDBFieldPrice;** is the product price.
  - **&mgiDBFieldShipping;** is the product shipping cost.
  - **&mgiDBFieldQualifier1;** is the first product qualifier.
  - **&mgiDBFieldQualifier2;** is the second product qualifier.
  - **&mgiDBFieldQualifier3;** is the third product qualifier.
  - **&mgiDBFieldQualifier4;** is the fourth product qualifier.
  - **&mgiDBFieldQualifier5;** is the fifth product qualifier.
  - **&mgiDBFieldWeight;** is the product weight.
  - **&mgiSBItemPriceTotal;** is the price of a product multiplied by the quantity purchased and includes any price rules that have been applied.
  - **&mgiSBItemShippingTotal;** is the shipping price of a product after shipping rules have been applied.
  - **&mgiSBItemTotal;** is the total price of a product (price subtotal plus shipping subtotal) after price and shipping rules have been applied.
- **Variables -** The mgiConfirmOrder tag sets the following page variables. The value of these page variables can be displayed anywhere on the page after the

mgiConfirmOrder tag by using an mgiGet tag.

  - **mgiSBSubtotal** is the price of the order after the price rule(s) have been applied.
  - **mgiSBTax** is the tax on the order after the tax rule(s) have been applied.
  - **mgiSBShippingTotal** is the shipping price of the order after the shipping rule(s) have been applied.
  - **mgiSBTotal** is the total price of the order after all rules have been applied.
  - **mgiSBPaymentInfo** is a table of the customer's payment information. The mgiSBPaymentInfo variable is only available in Automatic and Selective modes of the shopping basket.
  - **mgiSBBillingInfo** is a table of the customer's billing information. The mgiSBBillingInfo variable is only available in Automatic and Selective modes of the shopping basket.
  - **mgiSBShippingInfo** is a table of the customer's shipping information. The mgiSBShippingInfo variable is only available in Automatic and Selective modes of the shopping basket.

- **Default Payment, Billing and Shipping Post Arguments** - If you use the mgiCollectUserInfo tag to collect payment, billing, and shipping information, then you can display that information for confirmation by using the following post argument names with the mgiPostArgument tag.
  - **PaymentMethod** - Shopping basket payment method. Values include "Credit Card", "Purchase Order", and "Check".
  - **CardType** - Credit card type. Values include "Visa", "Mastercard", "Disover/Novus", "American Express", "Carte Blanche", "Diner's Club", "Bankcard", and/or "JCB" depending on shopping basket configuration.
  - **CCNumber** - Credit card number.
  - **ExpMo** - Credit card expiration month.
  - **ExpYe** - Credit card expiration year.
  - **CCName** - Name that appears on credit card.
  - **PONumber** - Purchase order number.
  - **POCompany** - Purchase order company.
  - **BName** - Billing name
  - **BCompany** - Billing company
  - **BAddress1** - Billing address
  - **BCity** - Billing city
  - **BState** - Billing state
  - **BProvince** - Billing province

- **BZipCode** - Billing zip code
- **BCountry** - Billing country
- **BPhone** - Billing phone
- **BFax** - Billing fax
- **BEmail** - Billing email
- **SCompany** - Shipping company
- **SAddress1** - Shipping address
- **SCity** - Shipping city
- **SState** - Shipping state
- **SProvince** - Shipping province
- **SZipCode** - Shipping zip code
- **SCountry** - Shipping country
- **SPhone** - Shipping phone
- **SFax** - Shipping fax
- **SEmail** - Shipping email

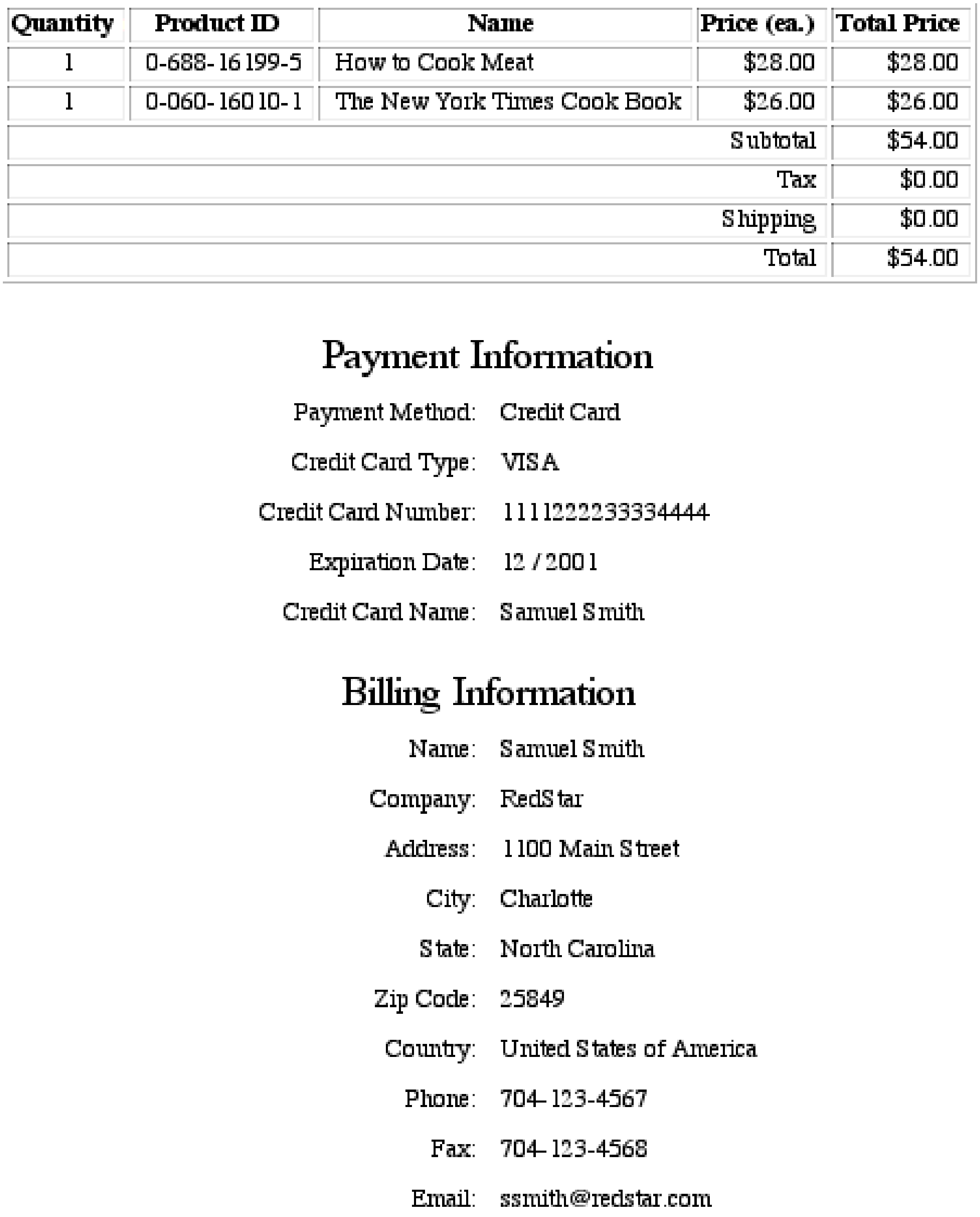
Example Usage and Output

Default Shopping Basket

```
<mgiToken>

<form action="sendorder.mgi" method="post">

<mgiConfirmOrder handle="Books"
shoppingBasketURL="http://www.domain.com/store/">
</mgiConfirmOrder>

<mgiButton name="Process Order">

</form>

</mgiToken>
```

In this example, the mgiConfirmOrder tag will display a confirmation screen containing the customer's order, payment, billing, and shipping information with the options defined in the "Books" configuration. Any information that the customer left blank is not displayed.

The default shopping basket display of the mgiConfirmOrder tag includes the quantity, product ID, product name, product price, extended product price, subtotal, tax, shipping and total. The mgiConfirmorder tag also displays the payment, billing, and shipping information that was entered by the customer.



Quantity	Product ID	Name	Price (ea.)	Total Price
1	0-688-16199-5	How to Cook Meat	$28.00	$28.00
1	0-060-16010-1	The New York Times Cook Book	$26.00	$26.00
			Subtotal	$54.00
			Tax	$0.00
			Shipping	$0.00
			Total	$54.00

Payment Information

Payment Method:	Credit Card
Credit Card Type:	VISA
Credit Card Number:	1111222233334444
Expiration Date:	12 / 2001
Credit Card Name:	Samuel Smith

Billing Information

Name:	Samuel Smith
Company:	RedStar
Address:	1100 Main Street
City:	Charlotte
State:	North Carolina
Zip Code:	25849
Country:	United States of America
Phone:	704-123-4567
Fax:	704-123-4568
Email:	ssmith@redstar.com

Custom Shopping Basket

```
<p><table cellpadding="5" cellspacing="1" border="0">
<tr bgcolor="#EEEECC">
<td><font size="-1" face="helvetica, arial, sans-serif">
```

```
<b>Qty</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Description</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Price Ea.</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Item Total</b></font></td>
</tr>

<mgiConfirmOrder handle="Default"
shoppingBasketURL="http://www.domain.com/">

<tr>
<td>&mgiDBFieldQuantity;</td>
<td><font size="-1" face="helvetica, arial, sans-serif">
&mgiDBFieldName;</font></td>
<td align="center">
<font size="-1" face="helvetica, arial, sans-serif">
$&mgiDBFieldPrice;</font></td>
<td align="center">
<font size="-1" face="helvetica, arial, sans-serif">
$&mgiSBItemPriceTotal;</font></td>
</tr>

</mgiConfirmOrder>

<tr bgcolor="#EEEECC">
<td colspan="3" align="right">
<font size="-1" face="helvetica, arial, sans-serif">
<b>Subtotal</b></font></td>
<td align="center">
<font size="-1" face="helvetica, arial, sans-serif">
<b>$<mgiGet name="mgiSBSubtotal"></b></font></td>
</tr>

<tr bgcolor="#EEEECC">
<td colspan="3" align="right">
<font size="-1" face="helvetica, arial, sans-serif">
<b>Tax</b></font></td>
<td align="center">
<font size="-1" face="helvetica, arial, sans-serif">
<b>$<mgiGet name="mgiSBTax"></b></font></td>
</tr>

<tr bgcolor="#EEEECC">
```

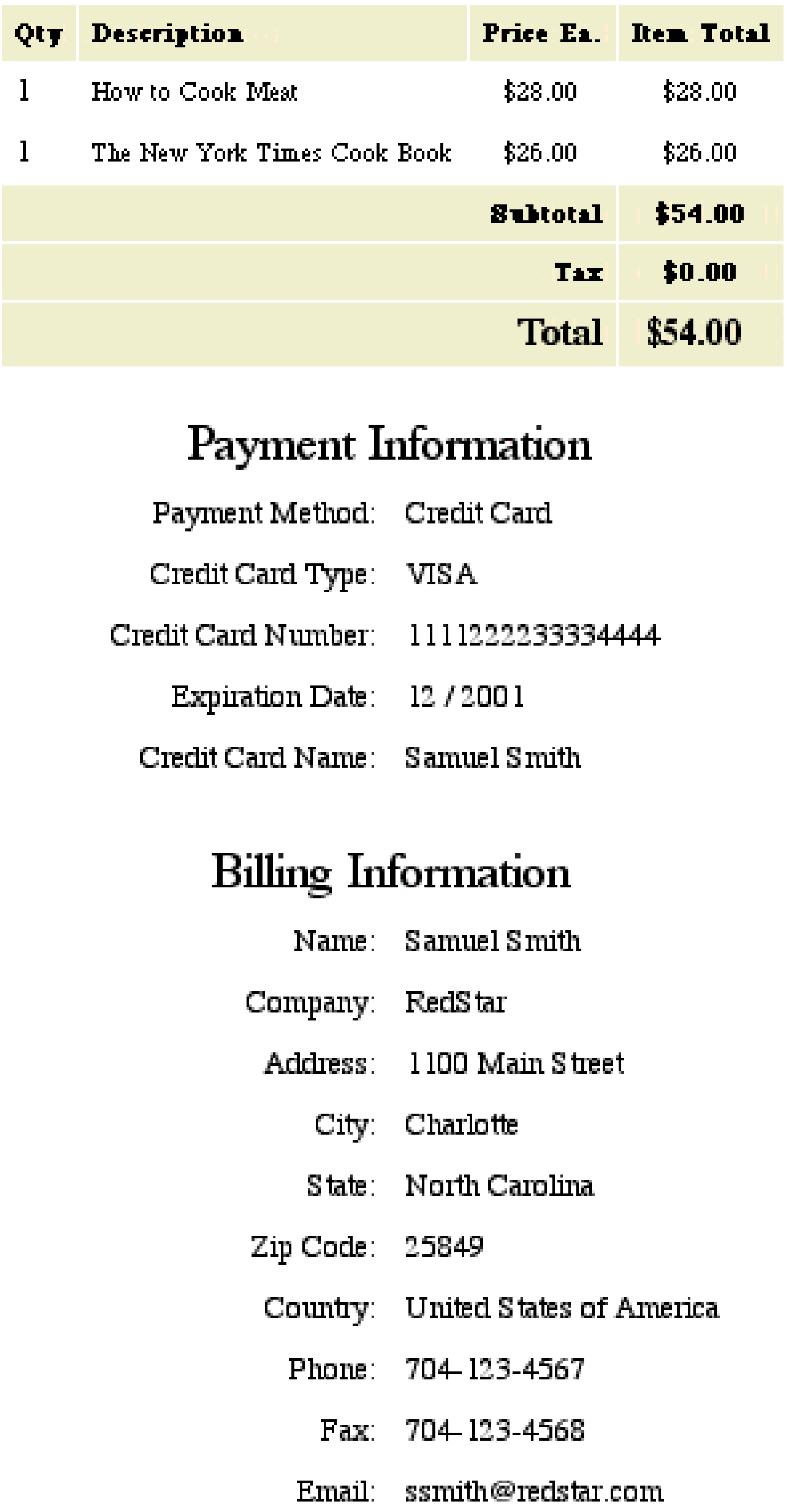
```
<td colspan="3" align="right">
<font size="+1" face="helvetica, arial, sans-serif">
<b>Total</b></font></td>
<td align="center">
<font size="+1" face="helvetica, arial, sans-serif">
<b>$<mgiGet name="mgiSBTotal"></b></font></td>
</tr>

</table>

<p><mgiGet name="mgiSBPaymentInfo">
<p><mgiGet name="mgiSBBillingInfo">
<p><mgiGet name="mgiSBShippingInfo">
```

If the default shopping basket does not fit with your site design, you may customize the shopping basket layout and the content of the shopping basket using the mgiConfirmOrder placeholders and variables. With a custom layout you can add font properties, cell colors, images, etc.

In a custom shopping basket, the code in the body of the mgiConfirmOrder tags is repeated for each item in the shopping basket and the placeholders are replaced with the item's specific information. After the mgiConfirmOrder tags, the shopping basket variables display information calculated from the current shopping basket contents.

Use the shopping basket variables to display the default payment, billing and shipping tables or create your own displays of the information with text, HTML and mgiPostArgument tags. If you use the mgiCollectUserInfo tag to collect payment, billing, and shipping information, then you can display that information in the by using specific post arguments (listed above under Technical Notes). However, if you created a custom form to collect payment, billing and shipping information, then use your post argument names to display that information for confirmation.

The custom shopping basket in this example displays this format:



Qty	Description	Price Ea.	Item Total
1	How to Cook Meat	$28.00	$28.00
1	The New York Times Cook Book	$26.00	$26.00
		Subtotal	**$54.00**
		Tax	**$0.00**
		Total	**$54.00**

Payment Information

Payment Method:	Credit Card
Credit Card Type:	VISA
Credit Card Number:	1111222233334444
Expiration Date:	12 / 2001
Credit Card Name:	Samuel Smith

Billing Information

Name:	Samuel Smith
Company:	RedStar
Address:	1100 Main Street
City:	Charlotte
State:	North Carolina
Zip Code:	25849
Country:	United States of America
Phone:	704-123-4567
Fax:	704-123-4568
Email:	ssmith@redstar.com

Suggested Usage

- Shopping Basket

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

The mgiCounter Tag

Tag Behavior

Use the mgiCounter tag to increment and display a text or graphic counter. Use the web-based administration interface to manage counters.

Tag Syntax

The mgiCounter tag has three modes. Each mode has different required and optional parameters. The modes of mgiCounter are:

- **process** - Displays and/or increments a text or graphic counter.
- **admin** - Creates a web-based interface to manage and reset counters
- **setValue** - Sets a specified counter to a specified value.

Process Mode

The process mode of the mgiCounter tag has one required parameter and eleven optional parameters. The tag form is:

```
<mgiCounter name="Name" mode="process" type="Text/Graphic"
length="Whole Number" display="Yes/No" increment="Yes/No"
font="FontName" fontFolderLocation="Folder Path"
ignoreClientIPAddress="Yes/No" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **name** - The name is the unique name of the counter.

Optional Parameters:

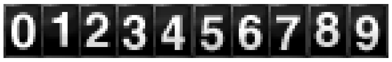
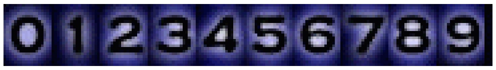
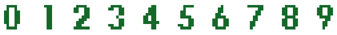
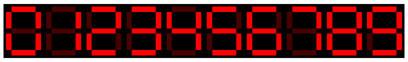
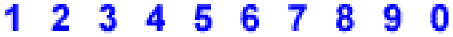
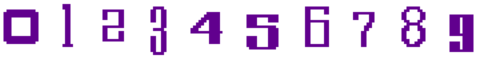
- **mode** - The mode is the function that the mgiCounter tag performs. In "**process**" mode, the mgiCounter tag displays and/or increments the counter value. "Process" is the default mode of the mgiCounter tag.
- **type** - The type parameter determines the appearance of the counter. If the type parameter value is "**Text**", then the counter is displayed as regular HTML text characters. If the type parameter value is "**Graphic**", then the counter is displayed as gif images in a specified font. The default value is "Text". If the type is set to "Graphic", then either the "font" or "fontFolderLocation" parameter is required.

- **length** - The minimum number of digits the counter should display. If the length parameter is not included, the default behavior of the mgiCounter tag is to display the total number of digits for the current counter value (e.g., a counter value of 204 requires the display of 3 characters).
- **display** - The display parameter determines if the counter value displays or is hidden. If the display parameter value is "**Yes**", then the counter displays on the page. If the display parameter value is "**No**", then the counter value does not display on the page. The default value is "Yes".
- **increment** - The increment parameter determines if the counter value increases when processed. If the increment parameter value is "**Yes**", then the counter value increases when processed. If the increment parameter value is "**No**", then the counter value does not increase when processed. The default value is "Yes".
- **font** - The font is the full name of the graphic font included with the mgiCounter tag that displays the counter value. To use the font parameter, you must set the "type" parameter value to "Graphic". The default font is "Odometer". See the list below for additional font names and displays.
- **fontFolderLocation** - The fontFolderLocation is the relative path to the folder containing the ten custom graphic counter images. Custom graphic counter images must be saved in gif format and named with the following syntax:
  - 0.gif
  - 1.gif
  - 2.gif
  - 3.gif
  - 4.gif
  - 5.gif
  - 6.gif
  - 7.gif
  - 8.gif
  - 9.gif
- **ignoreClientIPAddress** - The ignoreClientIPAddress parameter determines if the counter increments for visitor IP numbers that match the previous visitor's IP number. If the ignoreClientIPAddress parameter value is "**Yes**", then the counter increments for any IP number even if it matches the previous visitor's IP number. If the ignoreClientIPAddress parameter value is "**No**", then the counter only increments if the IP number of the current visitor does not match the IP number of the previous visitor. The default value is "Yes". This parameter can be used to create more accurate counts.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, counter information will be stored in the

specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Included Font Names and Displays:

Font Name	Font Display
`Odometer`	
`DigitBlueGlow`	
`Greeny`	
`RedDigital`	
`SmallBlue`	
`SaintFrancis`	

Admin Mode

The admin mode of the mgiCounter tag has one required parameter and four optional parameters. The tag form is:

```
<mgiCounter mode="admin" displayLimit="Integer"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiCounter tag performs. In "**admin**" mode, the mgiCounter tag creates a web-based interface that allows you to add new counters, set or reset counter values and delete counters.

Optional Parameters:

- **displayLimit** - The displayLimit is the maximum number of counters displayed per screen in the administration interface. The default value is 10. The maximum value of the displayLimit parameter is 50.

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, counter information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

SetValue Mode

The setValue mode of the mgiCounter tag has three required parameters and three optional parameters. The tag form is:

```
<mgiCounter mode="setValue" name="Name"
value="Integer" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiCounter tag performs. In "**setValue**" mode, the mgiCounter sets a specified counter to a specified value.
- **name** - The name is the unique name of the counter.
- **value** - The new value of the counter.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, counter information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the

odbcDatasource parameter.

Example Usage and Output

Text Counters

```
<mgiCounter name="index" type="Text" length="4"
increment="Yes" display="Yes">
```

In this example, the counter value will display as regular text with a padding of 4 characters and increment every time the page is accessed. If the previous value of the counter was 257, then the counter will display the following when the page is accessed by any web site visitor:

0258

Graphic Counters

```
<mgiCounter name="about" type="Graphic" increment="Yes"
display="Yes" font="SaintFrancis" ignoreClientIPAddress="No">
```

In this example, the counter value will display the "SaintFrancis" font and increment every time a unique IP number accesses the page. If the previous value of the counter was 300234, then the counter will display the following when the page is accessed by a different web site visitor:

Hidden Counters

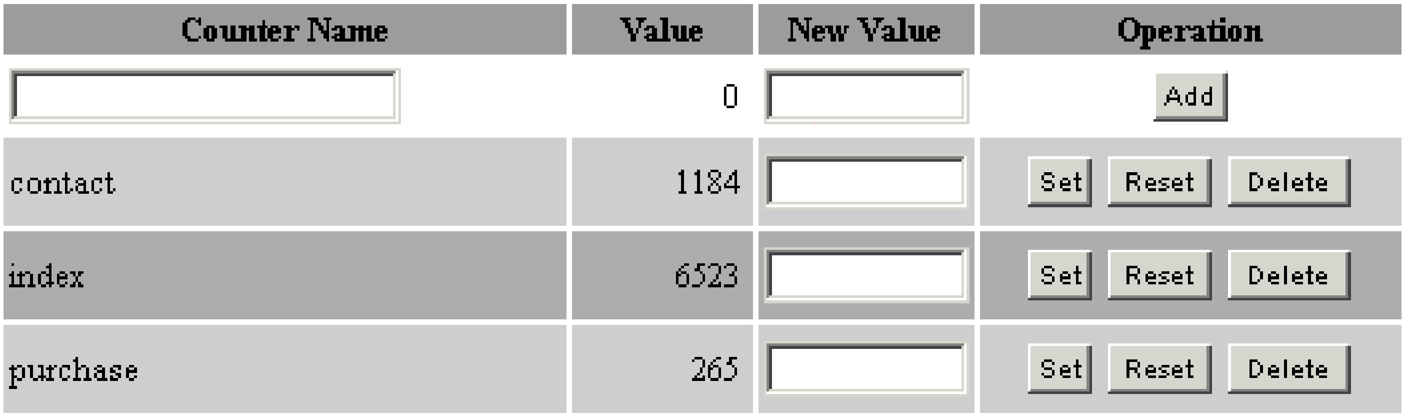
```
<mgiCounter name="home" display="No">
```

In this example, the counter increments every time the page is accessed, but the counter is not displayed. Hidden counters may be used for general page hit statistics.

Counter Admin

```
<mgiCounter mode="admin" displayLimit="25">
```

When you access a page with the mgiCounter tag in admin mode, the counter database administration screen displays. To add a new counter, enter the counter name, new value (if greater than zero) and click the "Add" button (Note: a new counter will be created automatically if you access a page with an mgiCounter tag in process mode and the counter's name does not exist). To set the counter value, enter a counter value beside the counter name in the "New Value" column, then click the "Set" button. To reset the counter value to zero, click the counter's "Reset" button. To delete a counter, click the counter's "Delete" button.



Counter Name	Value	New Value	Operation
	0		Add
contact	1184		Set Reset Delete
index	6523		Set Reset Delete
purchase	265		Set Reset Delete

Set Value

```
<mgiCounter mode="SetValue" name="service" value="0">
```

In this example, the "service" counter is reset to "0".

Suggested Usage

- Page Hit Statistics
- Serialization

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiCreditBank Tag

Tag Behavior

Use the mgiCreditBank tag to store and manage online credits purchased by a user.

Tag Syntax

The mgiCreditBank tag has eight modes. Each mode has different required and optional parameters. The modes of mgiCreditBank are:

- **increment** - Increments a user's account by a specified number of credits.
- **decrement** - Decrements a user's account by a specified number of credits.
- **decrementAndRedirect** - Decrements a user's account by a specified number of credits and redirects to a file.
- **queryCredits** - Displays the number of credits in a user's account.
- **queryUser** - Queries the credit bank database for the existence of a user.
- **addUser** - Adds a new user to the credit bank database.
- **deleteUser** - Deletes a user from the credit bank database.
- **admin** - Creates a web-based interface to manage credit bank users.

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

Increment Mode

The increment mode of the mgiCreditBank tag has three required parameters and three optional parameters. The tag form is:

```
<mgiCreditBank mode="increment" username="Name"
amount="Integer" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiCreditBank tag performs. In "**increment**" mode, the mgiCreditBank tag increments a user's account by a specified amount of credit.
- **username** - The username is the name of the user's account where the credit is applied.

- **amount** - The amount is the number of credits to apply to the user's account.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, credit bank information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

Decrement Mode

The decrement mode of the mgiCreditBank tag has four required parameters and three optional parameters. The tag form is:

```
<mgiCreditBank mode="decrement" username="Name"
amount="Integer" group="Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiCreditBank tag performs. In "**decrement**" mode, the mgiCreditBank tag subtracts a specified number of credits from a user's account.
- **username** - The username is the name of the user's account where the credit is subtracted.
- **amount** - The amount is the number of credits to subtracted from the user's account.
- **group** - The group is the name of the group that is allowed to access the information that the credits are used to purchase.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, credit bank information will be stored in the

specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

DecrementAndRedirect Mode

The decrementAndRedirect mode of the mgiCreditBank tag has seven required parameters and three optional parameters. The tag form is:

```
<mgiCreditBank mode="decrementAndRedirect"
username="Name" amount="Integer" group="Name"
allowedURL="URL" deniedURL="URL" unknownAccountURL="URL"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiCreditBank tag performs. In "**decrementAndRedirect**" mode, the mgiCreditBank tag subtracts a specified number of credits from a user's account and redirects the user to a specified file.
- **username** - The username is the name of the user's account where the credit is subtracted.
- **amount** - The amount is the number of credits to subtracted from the user's account.
- **group** - The group is the name of the group that is allowed to access the information that the credits are used to purchase.
- **allowedURL** - The allowedURL is absolute or relative path to the file where users are redirected when their account is successfully decremented in the DecrementAndRedirect mode.
- **deniedURL** - The deniedURL is is the absolute or relative path to the file where users are redirected when an account cannot be successfully decremented (i.e., the user does not have enough credits) in the DecrementAndRedirect mode.
- **unknownAccountURL** - The unknownAccountURL is is the absolute or relative path to the file where users are redirected when a user name does not exist in the DecrementAndRedirect mode

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, credit bank information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

QueryCredits Mode

The queryCredits mode of the mgiCreditBank tag has two required parameters and three optional parameters. The tag form is:

```
<mgiCreditBank mode="queryCredits" username="Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiCreditBank tag performs. In "**queryCredits**" mode, the mgiCreditBank tag displays the number of credits in a user's account.
- **username** - The username is the name of the user's account that is queried.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, credit bank information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the

odbcDatasource parameter.

- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

QueryUser Mode

The queryUser mode of the mgiCreditBank tag has two required parameters and three optional parameters. The tag form is:

```
<mgiCreditBank mode="queryUser" username="Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiCreditBank tag performs. In "**queryUser**" mode, the mgiCreditBank tag check the credit bank database for the existence of the specified username in any group. If the username exists, the value "**Yes**" is displayed. If the username does not exist, then the value "**No**" is displayed.
- **username** - The username is the name of the user's account that is queried.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, credit bank information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

AddUser Mode

The addUser mode of the mgiCreditBank tag has four required parameters and three optional parameters. The tag form is:

```
<mgiCreditBank mode="addUser" username="Name"
amount="Integer" group="Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiCreditBank tag performs. In "**addUser**" mode, the mgiCreditBank tag add a new username to the credit bank database with the specified credit and specified group.
- **username** - The username is the name of the user's account to create.
- **amount** - The amount is the number of credits to set in the user's account.
- **group** - The group is the name of the access group that the user belongs to.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, credit bank information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

DeleteUser Mode

The deleteUser mode of the mgiCreditBank tag has two required parameters and three optional parameters. The tag form is:

```
<mgiCreditBank mode="deleteUser" username="Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiCreditBank tag performs. In "**deleteUser**" mode, the mgiCreditBank tag deletes an existing user from the credit bank database.

- **username** - The username is the name of the user's account to delete.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, credit bank information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

Admin Mode

The admin mode of the mgiCreditBank tag has one required parameters and four optional parameters. The tag form is:

```
<mgiCreditBank mode="admin" displayLimit="Integer"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiCreditBank tag performs. In "**deleteUser**" mode, the mgiCreditBank tag deletes an existing user from the credit bank database.

Optional Parameters:

- **displayLimit** - The displayLimit is the maximum number of usernames displayed per screen in the administration interface. The default value is 10. The maximum value of the displayLimit parameter is 25.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, credit bank information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the

server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

Example Usage and Output

```
<mgiCreditBank mode="decrementAndRedirect" amount="5"
group="subscribers" username={mgiRequest name="Username"}
allowedURL="http://www.domain.com/allowed.mgi"
deniedURL="http://www.domain.com/denied.mgi"
unknownAccountURL="http://www.domain.com/unknownaccount.mgi">
```

In this example, a user's credit bank account is decremented 5 credits each time they access the page. If the user's account contains at least 5 credits, they are redirected to the allowed.mgi file. If the user's account contains less than 5 credits, they are redirected to denied.mgi. If the user name does not exist in the credit bank database, they are redirected to unknownaccount.mgi.

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

Suggested Usage

- Subscriber-Based Downloads
- Fee-Based Downloads

Return to the mgiCreditBank Mode Menu | Examples and Output | Suggested Usage

The mgiCreditCard Tag

Tag Behavior

Use the mgiCreditCard tag to verify that a given value is a valid credit card number. This tag does **not** authorize credit card purchases.

Tag Syntax

The mgiCreditCard tag has one required parameter and no optional parameters. The tag form is:

```
<mgiCreditCard cardNumber="creditCardNumber">
```

Required Parameters:

- **cardNumber -** The cardNumber is the credit card number to validate. If the card number is valid, the value "**Yes**" is displayed. If the card number is not valid, the value "**No**" is displayed. The card number may be formatted without spaces (4258154612546458), with spaces (4258 1546 1254 6458) or with hyphens (4258-1546-1254-6458). Spaces and hyphens are ignored during the verification process. Valid card number may only contain number and must be between 13 and 16 characters in length.

Optional Parameters:

- **None.**

Example Usage and Output

```
<mgiCreditCard cardNumber="428156214356">
```

In this example, the mgiCreditCard tag will check to see if the number "428156214356" is a valid credit card number. Since the number is invalid, the tag will return the value "No".

Suggested Usage

- Credit Card Validation

The mgiDate Tag

Tag Behavior

Use the mgiDate tag to display the current day, date, month and/or year of the server or offset from GMT (see also mgiTime).

Tag Syntax

The mgiDate tag has no required parameters and four optional parameters. The tag form is:

```
<mgiDate format="Format" gmtOffset="Hours"
html="HTML" text="text">
```

Required Parameters:

- **None.**

Optional Parameters:

- **format -** The format is the format of the date components. If the format parameter is not specified, the default display of the mgiDate tag is the server's date format. You may include multiple format parameters in one mgiDate tag. Valid formats for the mgiDate tag are:
    - **LongWeekDay -** Displays the full week day without abbreviation (e.g. Tuesday).
    - **ShortWeekDay -** Displays the three-letter abbreviation of the week day (e.g. Sat).
    - **NumericWeekday -** Displays a numeric week day between Sunday (1) and Saturday (7).
    - **LongMonth -** Displays the full month without abbreviation (e.g.,November).
    - **ShortMonth -** Displays the three-letter abbreviation of the month (e.g. Jun).
    - **NumericMonth -** Displays a numeric month between January (1) and December (12).
    - **PaddedNumericMonth -** Displays a two-digit numeric month between January (01) and December (12).
    - **Day -** Displays a numeric day of the month as a number between 1 and 31.
    - **PaddedDay -** Displays a two-digit numeric day of the month as a number between 01 and 31.

  - **JulianDayNumber -** Displays the integer julian day (e.g., 31 Jan 2000 is julian day number 2451574).
  - **DayOfYear -** Displays a numeric day between 1 and 366.
  - **PaddedDayOfYear -** Displays a two-digit numeric day between 001 and 366.
  - **WeekOfYear -** Displays the numeric week of the year from 1 to 52.
  - **PaddedWeekOfYear -** Displays the two-digit numeric week of the year from 01 to 52.
  - **LongYear -** Displays the four-digit year as a number between 1000 and 9999.
  - **ShortYear -** Displays the two-digit year as a number between 00 and 99.
- **gmtOffset -** The gmtOffset is the whole or half number of hours the date is offset from Greenwich Mean Time preceeded by a positive (+) or negative (-) sign (e.g., "+5" or "-3.5"). Only half hours may be entered. Other tenths of an hour may not be entered. When the GMTOffset parameter is not present, the local date of the server is displayed. A global GMTOffset can be enabled in the region preferences for your domain. If you do not have access to the region preferences for your domain, contact the server administrator. Including a gmtOffset parameter overrides the gmtOffset in the region preferences.
- **html -** The html is the HTML code and text that is decoded and displayed relative to the specified date formats. Any HTML in the value of the html parameter should already be encoded (e.g., quotation marks should be entered as "&quot;"). You may include multiple html parameters in one mgiDate tag.
- **text -** The text is the string of characters that is decoded and displayed relative to the specified date formats. Text in the value of the text parameter is not decoded prior to display. You may include multiple text parameters in one mgiDate tag.

Example Usage and Output

```
<mgiDate>
```

If the current server date is Sunday, the 9th of March 2001, the default mgiDate tag might display like the following. The actual default display of the date depends on the server's date configuration.

Sunday, March 9, 2001

```
<mgiDate html="Current Month: &lt;font
color=&quot;#006699&quot;&gt;" format="longMonth"
text=" " format="longYear" html="&lt;/font&gt;">
```

If the current server date is Sunday, the 9th of March 2001, the mgiDate tag in this example would display as:

Current Month: March 2001

```
<mgiDate format="longWeekday" text=", " format="day"
text=" " format="longMonth"
text=", " format="longYear" GMTOffset="-8">
```

If the current GMT date is 5 am Sunday, the 11th of March 2001, but you want the local date for Los Angeles, California, then the GMTOffset indicated in this example might display as:

Saturday, 10 March, 2001

Suggested Usage

- Server Side Includes
- Guestbooks
- Form Processing
- Database Submissions
- Emails

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiDoNotProcess Tag

Tag Behavior

Use the mgiDoNotProcess tag to prevent the MGI code within the body from being processed. The code in the body of the mgiDoNotProcess tag is **not** removed from the page before it is served. (see also mgiInlineDoNotProcess and mgiComment)

Tag Syntax

The mgiDoNotProcess tag has a beginning tag with no required parameters and no optional parameters, a body, and an ending tag. The tag form is:

```
<mgiDoNotProcess>
MGI Code and Text
</mgiDoNotProcess>
```

Required Parameters:

- **None**.

Optional Parameters:

- **None**.

Example Usage and Output

```
<mgiDoNotProcess>
<mgiModifyDatabase mode="deleteRecord" databaseName="Scores"
keyfieldName="ID" fieldValue={mgiPostArgument name="ID"}>
</mgiDoNotProcess>
```

The mgiDoNotProcess tag prevents the MGI code in it's body from being processed, BUT the MGI tags are not removed from the page and are therefore viewable in the page source. In this example, the mgiDoNotProcess tag is used to deactivate a portion of MGI code during site development.

The mgiDynamicList Tag

Tag Behavior

Use the mgiDynamicList tag to dynamically build a delimited list of unique values. Multiple sources of values can be used to build one list.

Tag Syntax

The mgiDynamicList tag has five sources. Each source has different required and optional parameters. The sources of mgiDynamicList are:

- **postArguments -** Dynamically creates a delimited list of unique values from post arguments.
- **pathArguments -** Dynamically creates a delimited list of unique values from path arguments.
- **pageVariables -** Dynamically creates a delimited list of unique values from page variables.
- **databaseField -** Dynamically creates a delimited list of unique values from a database field.
- **string -** Dynamically creates a list of delimited unique values from a delimited string of values.

PostArguments, PathArgument and PageVariable Sources

The postArguments, pathArgument and pageVariable sources of mgiDynamicList tag have one required parameters and two optional parameters. The tag form is:

```
<mgiDynamicList source="Source" order="Order"
listDelimiter="Character">
```

Required Parameters:

- **source -** The source is the location of the list values. The "**postArguments**" source creates list values from all post arguments. The "**pathArguments**" source creates list values from all path arguments. The "**pageVariables**" source creates list values from all page variables. Multiple source parameters may be included in one mgiDynamicList tag.

Optional Parameters:

- **order** - The order determines the ordering of the list values. If the order parameter value is "**alphabetical**", then list values are alphabetized from A to Z. If the order parameter value is "**numerical**", then the list values are listed in ascending order from smallest to largest. When using the "numerical" order, all list values must be numbers. If the order parameter value is "**none**", then the list values are left in their original order. The default value is "None".
- **listDelimiter** - The listDelimiter is the character that separates each value in the list. To use escape characters such as carriage returns, line feeds, and tabs as delimiters use the appropriate Escaped String format. The default list delimiter is a comma.

DatabaseField Source

The databaseField source of mgiDynamicList tag has three required parameters and five optional parameters. The tag form is:

```
<mgiDynamicList source="databaseField"
databaseName="Database" fieldName="Field" order="Order"
listDelimiter="Character" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **source** - The source is the location of the list values. The "**databaseField**" source creates list values from all values in a specified database. Multiple source parameters may be included in one mgiDynamicList tag.
- **databaseName** - The databaseName is the name of the database that contains the field values to be included in the list.
- **fieldName** - The fieldName is the name of the database field that contains the values to be included in the list. The database field must be a "text" type field.

Optional Parameters:

- **order** - The order determines the ordering of the list values. If the order parameter value is "**alphabetical**", then list values are alphabetized from A to Z. If the order parameter value is "**numerical**", then the list values are listed in ascending order from smallest to largest. When using the "numerical" order, all list values must be numbers. If the order parameter value is "**none**", then the list values are left in their original order. The default value is "None".
- **listDelimiter** - The listDelimiter is the character that separates each value in the list. To use escape characters such as carriage returns, line feeds, and tabs as delimiters use the appropriate Escaped String format. The default list delimiter is a comma.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be retrieved from the

specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

String Source

The string source of mgiDynamicList tag has three required parameters and two optional parameters. The tag form is:

```
<mgiDynamicList source="String" name="Name" string="Text"
stringDelimiter="Character" order="Order"
listDelimiter="Character">
```

Required Parameters:

- **source** - The source is the location of the list values. The "**string**" source creates list values from all values in a specified string of characters. Multiple source parameters may be included in one mgiDynamicList tag.
- **string** - The string is the delimited text to use as the list values.
- **stringDelimiter** - The stringDelimiter is the character that separates each value in the string. To use escape characters such as carriage returns, line feeds, and tabs, use the appropriate Escaped String format.

Optional Parameters:

- **order** - The order determines the ordering of the list values. If the order parameter value is "**alphabetical**", then list values are alphabetized from A to Z. If the order parameter value is "**numerical**", then the list values are listed in ascending order from smallest to largest. When using the "numerical" order, all list values must be numbers. If the order parameter value is "**none**", then the list values are left in their original order. The default value is "None".
- **listDelimiter** - The listDelimiter is the character that separates each value in the list. To use escape characters such as carriage returns, line feeds, and tabs as delimiters use the appropriate Escaped String format. The default list delimiter is a comma.

Example Usage and Output

PostArguments Source

```
<mgiDynamicList source="postArguments" order="Alphabetical">
```

In this example, categories from a form submission are ordered alphabetically and displayed in a delimited list list.

Architecture,Baby Care,Fashion,Health and Beauty,Science and Technology

PathArguments Source

```
<mgiDynamicList source="pathArguments"
order="Numerical" listDelimiter="-">
```

In this example, available quantities are passed via path arguments and are used to create a list with values in numerical order separated by dashes.

1-10-100-200

PageVariables Source

```
<mgiDynamicList source="pageVariables" listDelimiter="\r">
```

In this example, available product sizes from page variables are used to create a list delimited with carriage returns. The list values are not re-ordered.

```
XL
S
M
L
2X
```

DatabaseField Source

```
<mgiDynamicList source="databaseField" fieldName="Jobs"
databaseName="JobDatabase" order="Alphabetical">
```

In this example, a list of jobs in a database are used to create a list.

C++ Programmer,Database Programmer,LAN Specialist,System Administrator

String Source

```
<mgiDynamicList source="string" stringDelimiter=","
string={mgiPostArgument name="Names"} listDelimiter="/">
```

In this example, a string of names delimited by commas from a form submission is embedded in the string parameter of mgiDynamicList to create a list of names delimited by a slash.

Beth Johnson/Anita Brown/Bradley West/Cole Sanders

Multiple Sources

```
<mgiDynamicList source="postArguments" source="pathArguments"
order="Alphabetical">
```

In this example, values from post arguments and path arguments are used to create a list.

A23345,A26789,BC2234,D99382,LM2345,P92345,X23459

Suggested Usage

- Searchable Databases
- Form Processing

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiDynamicPopup Tag

Tag Behavior

Use the mgiDynamicPopup tag to dynamically build a popup menu (select) of unique values. Multiple sources of values can be used to build one popup menu.

Tag Syntax

The mgiDynamicPopup tag has five sources. Each source has different required and optional parameters. The sources of mgiDynamicPopup are:

- **postArguments -** Dynamically creates a popup of unique values from post arguments.
- **pathArguments -** Dynamically creates a popup of unique values from path arguments.
- **pageVariables -** Dynamically creates a popup of unique values from page variables.
- **databaseField -** Dynamically creates a popup of unique values from a database field.
- **string** - Dynamically creates a popup of unique values from a delimited string of values.

PostArguments, PathArgument and PageVariable Sources

The postArguments, pathArgument and pageVariable sources of mgiDynamicPopup tag have two required parameters and three optional parameters. The tag form is:

```
<mgiDynamicPopup source="Source" name="Name" order="Order"
selectedItem="Item Name" selectedValue="Item Value">
```

Required Parameters:

- **source** - The source is the location of the popup menu values. The "**postArguments**" source creates popup menu values from all post arguments. The "**pathArguments**" source creates popup menu values from all path arguments. The "**pageVariables**" source creates popup menu values from all page variables. Multiple source parameters may be included in one mgiDynamicPopup tag.
- **name** - The name is the name of the popup menu (select) that is created.

Optional Parameters:

- **order** - The order determines the ordering of the popup menu values. If the order parameter value is "**alphabetical**", then popup menu values are alphabetized from A to Z. If the order parameter value is "**numerical**", then the popup menu values are listed in ascending order from smallest to largest. When using the "numerical" order, all popup menu values must be numbers. If the order parameter value is "**none**", then the popup menu values are left in their original order. The default value is "None".

- **selectedItem** - The initial item in the popup menu that will be selected by default.
- **selectedValue** - The initial value in the popup menu that will be selected by default.

DatabaseField Source

The databaseField source of mgiDynamicPopup tag has four required parameters and six optional parameter. The tag form is:

```
<mgiDynamicPopup source="databaseField" name="Name"
databaseName="Database" fieldName="Field" order="Order"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password" selectedItem="Item Name"
selectedValue="Item Value">
```

Required Parameters:

- **source** - The source is the location of the popup menu values. The "**databaseField**" source creates popup menu values from all values in a specified database field. Multiple source parameters may be included in one mgiDynamicPopup tag.
- **name** - The name is the name of the popup menu (select) that is created.
- **databaseName** - The databaseName is the name of the database that contains the field values to be included in the popup menu.
- **fieldName** - The fieldName is the name of the database field that contains the values to be included in the popup menu. The database field must be a "text" type field.

Optional Parameters:

- **order** - The order determines the ordering of the popup menu values. If the order parameter value is "**alphabetical**", then popup menu values are alphabetized from A to Z. If the order parameter value is "**numerical**", then the popup menu values are listed in ascending order from smallest to largest. When using the "numerical" order, all popup menu values must be numbers. If the order parameter value is "**none**", then the popup menu values are left in their original order. The default value is "None".
- **selectedItem** - The initial item in the popup menu that will be selected by default.
- **selectedValue** - The initial value in the popup menu that will be selected by default.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be retrieved from the

specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

String Source

The string source of mgiDynamicPopup tag has four required parameters and three optional parameters. The tag form is:

```
<mgiDynamicPopup source="String" name="Name" string="Text"
stringDelimiter="Character" order="Order"
selectedItem="Item Name" selectedValue="Item Value">
```

Required Parameters:

- **source** - The source is the location of the popup menu values. The "**String**" source creates popup menu values from all values in a specified string of characters. Multiple source parameters may be included in one mgiDynamicPopup tag.
- **name** - The name is the name of the popup menu (select) that is created.
- **string** - The string is the delimited text to use as the popup menu values.
- **stringDelimiter** - The stringDelimiter is the character that separates each popup menu value in the string. To use escape characters such as carriage returns, line feeds, and tabs, use the appropriate Escaped String format.

Optional Parameters:

- **order** - The order determines the ordering of the popup menu values. If the order parameter value is "**alphabetical**", then popup menu values are alphabetized from A to Z. If the order parameter value is "**numerical**", then the popup menu values are listed in ascending order from smallest to largest. When using the "numerical" order, all popup menu values must be numbers. If the order parameter value is "**none**", then the popup menu values are left in their original order. The default value is "None".
- **selectedItem** - The initial item in the popup menu that will be selected by default.
- **selectedValue** - The initial value in the popup menu that will be selected by default.

Example Usage and Output

PostArguments Source

```
<mgiDynamicPopup source="postArguments" name="Categories"
order="Alphabetical">
```

In this example, categories from a form submission are ordered alphabetically and displayed in a popup menu.

Select Item

PathArguments Source

```
<mgiDynamicPopup source="pathArguments" name="Quantity"
order="Numerical" selectedItem="-- Number --">
```

In this example, available quantities are passed via path arguments and are used to create a popup menu with values in numerical order.

-- Number --

PageVariables Source

```
<mgiDynamicPopup source="pageVariables" name="Sizes"
selectedItem="-- Size --" selectedValue="none">
```

In this example, available product sizes from page variables are used to create a popup menu. The popup menu values are not re-ordered.

-- Size --

DatabaseField Source

```
<mgiDynamicPopup source="databaseField" fieldName="Jobs"
databaseName="JobDatabase" name="Positions"
order="Alphabetical">
```

In this example, a list of jobs in a database are used to create a popup menu.

Select Item

String Source

```
<mgiDynamicPopup source="string" stringdelimiter=","
string={mgiPostArgument name="Names"} name="Nominees">
```

In this example, a string of names delimited by commas from a form submission is embedded in the string parameter of mgiDynamicPopup to create a popup menu of names.

Select Item

Multiple Sources

```
<mgiDynamicPopup source="postArguments" source="pathArguments"
order="Alphabetical" name="Items">
```

In this example, values from post arguments and path arguments are used to create a popup menu.

Select Item

Suggested Usage

- Searchable Databases
- Form Processing

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiEditDatabase Tag

Tag Behavior

Use the mgiEditDatabase tag to create, modify and delete custom databases via a web-based interface.

Tag Syntax

The mgiEditDatabase tag has no required parameters and nine optional parameters. The tag form is:

```
<mgiEditDatabase mode="Mode" databaseName="Database"
showMGIDatabases="Yes/No" advancedSearch="On/Off"
mirrorButtons="Yes/No" uniqueIDFieldName="Field"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **None.**

Optional Parameters:

- **mode** - The mode is the function that the mgiEditDatabase tag performs. In "**manageFields**" mode, the mgiEditDatabase tag displays the web-based interface for creating and deleting database fields. In "**manageRecords**" mode, the mgiEditDatabase tag displays the web-based interface for creating, modifying and deleting database records. If the mode parameter is not included, then buttons for managing fields and managing records are displayed in the interface. The databaseName parameter is required if you include the mode parameter.
- **databaseName** - The databaseName is the name of the database to manage. If the databaseName parameter is not included, then all databases in the region are displayed. The databaseName parameter is required if you include the mode parameter.
- **showMGIDatabases** - The showMGIDatabases parameter determines if databases used internally by MGI are displayed. If the showMGIDatabases parameter value is "**Yes**", then internal MGI databases are displayed. If the showMGIDatabases parameter value is "**No**", then internal MGI databases are not displayed. The default value is "No".
- **advancedSearch** - The advancedSearch parameter determines if the advanced search field displays on the search screen of the edit database interface. If the advancedSearch

parameter value is "**On**", then advanced search field displays. If the advancedSearch parameter value is "**Off**", then the advanced search field does not display. The default value is "Off". Search strings should contain field name and search criteria in the format keyFieldName='criteria'. Field name and search criteria pairs should be separated by the search operator "AND", "OR", or "NOT". To order the results from a search string, include the order by field in the format ORDER BY keyFieldName. To reverse the order of results in a search string, specify descending order by including DESC. The following are example search strings:

```
"LastName='J*' OR LastName='L'
AND Registration='Yes' ORDER BY CustID DESC"

"FirstName='Ken' AND NOT LastName='Barker'
```

- **mirrorButtons** - The mirrorButtons parameter determines if the navigation and function buttons of the edit database interface are displayed at the top of the edit database screens. If the mirrorButtons parameter value is "**Yes**", then the buttons are displayed at the top and bottom of the edit database screens. If the mirrorButtons parameter value is "**No**", then buttons are displayed only at the bottom of the edit database screens. The default value is "No".
- **uniqueIDFieldName** - The uniqueIDFieldName parameter is the name of the ODBC database field to use for a unique record value.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, databases will be created and managed using the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Technical Notes

- **Field Name Length Limit** - The length of field names in the internal MGI database is limited according to the type of field and operating system. All field names that are not indexed are limited to 63 characters. The limit for non-indexed field names applies to all operating systems. Indexed field names on Macintosh operating systems 9.04 and prior are limited to 25 characters. Indexed field names on all supported Windows operating systems are limited to 63 characters.

- **Field Name Characters** - The characters allowed in database field names for the built-in MGI database are alpha-numeric characters (i.e., letters and numbers), underscores (_), hyphens (-), and spaces. When adding a field, any non-allowed characters in the field name are automatically replaced with an underscore.
- **Field Types** - Database fields types are specified when you create each field. The following is a description of each field type. Fields can be designated as indexed and/or unique. A field must be indexed in order to search for information in the field (in the mgiEditDatabase admin or using mgiSearchDatabase) or order search results by the field. A unique field will not allow two records to contain the same value in that field (even if the value is a blank field).
    - **Whole Number (Integer)** - A whole number field can contain only positive and negative whole numbers without decimals.
    - **Positive Number (UInteger)** - A positive number field can contain only positive whole numbers.
    - **Multi-precision Number** - A multi-precision number field can contain up to 50 significant digits (including decimal positions) and supports the concept of a NULL number (a blank number that is not defaulted to 0). When this field is created, the scale (i.e., the number of digits to the right of the decimal point) is specified. The maximum scale is 25. The multi-precision field type does not suffer from the usual rounding errors that are common when dealing with floating point numbers.
    - **True/False (Boolean)** - A boolean field can contain the values "True" or "False" to discriminate records.
    - **Text** - A text field can contain up to 250 alpha-numeric characters. The maximum length can be specified during field creation, but defaults to 25 if not specified.
    - **Long Text** - A long text field can contain text that is greater than 250 alpha-numeric. Search results cannot be ordered by long text fields.
- **Indexed Field Limit** - You may create unlimited fields, but only 31 fields may be indexed.
- **Record Data Limit** - Each database record can hold a maximum of 64 KB of data across all fields.

Example Usage and Output

```
<mgiEditDatabase>
```

In this example, the mgiEditDatabase tag would display the web-based database interface with all databases in the region listed and with options to modify fields and modify records.

Suggested Usage

- Databases

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiEncryptURL Tag

Tag Behavior

Use the mgiEncryptURL tag to encrypt the link to specified files.

Tag Syntax

The mgiEncryptURL tag has one required parameter and two optional parameters. The tag form is:

```
<mgiEncryptURL url="File Path" expiration="Julian Date"
mimeType="Mime Type">
```

Required Parameters:

- **url -** The url is the absolute path from the root folder to the file to encrypt (e.g., Root/Folder/File.jpg).

Optional Parameters:

- **expiration -** The expiration is the Julian date that you want the encryption link to expire. (see mgiJulianDay) The default expiration is no expiration (i.e., the encrypted link will never expire).
- **mimeType -** The mimeType is the MIME type that should be used to send the data in the file (e.g., "image/jpeg"). If no MIME type is selected, an appropriate header will be sent according to the extension of the file.

Example Usage and Output

```
<mgiEncryptURL url="portrait.jpg"
expiration={mgiJulianDay month="10" day="25"
year="2000" dateonly="yes"}>
```

In this example, the link to the file "portrait.jpg" is encrypted and the encrypted link expires on October 25, 2000. Encrypted file links protect your images, downloads, and other files.

Suggested Usage

- File Downloads
- Online Image Sales
- Software Sales

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiGet Tag

Tag Behavior

Use the mgiGet tag to display the value of a page or site variable (see also mgiSet). Variables are containers for text, numbers, calculations, results, etc. that can be used after they are created (i.e., set) throughout a single page (page variables) or throughout an entire web site (site variables).

Tag Syntax

The mgiGet tag has one required parameter and seven optional parameters. The tag form is:

```
<mgiGet name="Name" scope="Page/Site" defaultValue="Text"
mode="Function" delimiter="String" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **name** - The name is the name of the variable to display. Variable names are unique within a scope. If you include the mode parameter, the name parameter is not required.

Optional Parameters:

- **scope** - The scope is the type of variable to display. "**Page**" variables are available to display from the location they are set on a page to the end of the page. "**Site**" variables are available to display after they are set on any page in a web site. The default scope is "Page".
- **defaultValue -** The defaultValue is the text that displays if the variable is not found or if the variable is blank. If a default value is not specified, an empty string (blank) is displayed.
- **mode** - The mode is the function that the mgiGet tag performs. In "**returnAllNames**" mode, the names of all variables from the specified scope are displayed. In "**returnAllValues**" mode, the values of all variables from the specified scope are displayed. In "**returnAll**" mode, the names and values of all variables from the specified scope are displayed in the format below. If you include the mode parameter, the name parameter is not required.

  Name1: Value1

  Name2: Value2

  Name3: Value3

  NameX: ValueX

- **delimiter** - The delimiter is the character (e.g., "_", tabs, etc.) that is used to separate the names and values in returnAllNames, returnAllValues, and returnAll modes. The default delimiter for the "returnAllNames" and "returnAllValues" modes is a comma . The default delimiter for the "returnAll" mode is a carriage return and line feed (CRLF). In order to use special reserved characters such as tabs and backslashes as delimiters, you must use the appropriate Escaped String format.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, site variable information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Example Usage and Output

Page Variable

```
<mgiSet name="QuizScore">
<mgiMath resultPrecision="1">
<mgiPostArgument name="Question1">
+<mgiPostArgument name="Question2">
+<mgiPostArgument name="Question3">
+<mgiPostArgument name="Question4">
</mgiMath>
</mgiSet>

Your quiz score is <mgiGet name="QuizScore">
```

Variables are so versatile that you could use them in combination with almost any other MGI tag. In this example, a student answers a simple 4 question quiz in a form. The result of a quiz score is calculated and set in a page variable. The mgiGet tag is then used to display the quiz score from the page variable to the student.

Site Variable

```
Now serving <mgiGet name="Subscribers" scope="Site">
```

```
Subscribers! Join us today!
```

Once they are set, site variables can be displayed on any page in a site. In this example, the number of subscribers to an online service is update during each subscription and set in a site variable named "Subscribers". This information is then available to display anywhere on the site using mgiGet and the scope parameter.

Modes

```
<mgiLoop itemList={mgiGet mode="returnAllValues"}>
  <mgiSendMail to="&mgiLoopIndex;" from="sales@domain.com"
  subject="Sale" mailserver="mail.domain.com">

  Dear Customer,

  Shop our store during the month of October
  and save 10% off all new fall merchandise!

  </mgiSendMail>
</mgiLoop>
```

In returnAllNames and returnAllValues mode, you can easily create a loop to perform a function on each name or value. In this example, a list of email addresses have each been set in a variable after a search. The comma-delimited list of variable values is then embedded in a loop that emails a form letter to each address.

```
<mgiGet mode="returnAll" scope="Site">
```

In returnAll mode, the variable names and values are formatted with colons in a list. This feature is useful for debugging purposes because you can display a comprehensive list of variables rather than listing an mgiGet tag for each variable. In this example, the site variables would display in this format:

```
Header: Fall is here. See our selection of
winterizing products.
Phone: 667-223-4567
Version: 2.4
```

Suggested Usage

- Storage of Information
- Combining Information from Multiple Sources (Post Arguments, Path Arguments, Variables, etc.)
- Embedding

The mgiGetCookie Tag

Tag Behavior

Use the mgiGetCookie tag to display the value of a cookie that has been saved on a visitor's computer (see also mgiSetCookie).

Tag Syntax

The mgiGetCookie tag has one required parameter and two optional parameters. The tag form is:

```
<mgiGetCookie name="Name" useEncryption="Yes/No"
defaultValue="defaultString">
```

Required Parameters:

- **name -** The name is the name of the cookie to retrieved.

Optional Parameters:

- **useEncryption -** The useEncryption parameter specifies if the cookie value should be decrypted. If the useEncryption parameter value is "**Yes**", then the cookie value is decrypted. If the useEncryption parameter value is "**No**", then the cookie value is not decrypted. The default value is "No."
- **defaultValue -** The defaultValue is the text that displays if the cookie is not found or if the cookie value is blank. If a default value is not specified, an emtpy string (blank) is displayed.

Example Usage and Output

```
Hello <mgiGetCookie name="Name" defaultValue="Customer">.
Welcome back.
```

The mgiGetCookie in this example would display the value of the "Name" cookie at the location of the mgiGetCookie tag.

```
Hello Jim Sampson. Welcome back.
```

Suggested Usage

- Repeat Customer Information
- Login Information

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiGetErrorParameter Tag

Tag Behavior

Use the mgiGetErrorParameter tag to create a custom error page for your domain (or for specific folders in your domain). Custom error pages are enabled in the region preferences for your domain. If you do not have access to the region preferences for your domain, contact the server administrator.

Tag Syntax

The mgiGetErrorParameter tag has one required parameter and no optional parameters. The tag form is:

```
<mgiGetErrorParameter name="Error Parameter">
```

Required Parameters:

- **name** - The name is the name of the error parameter to display. Valid parameter names are:
  - **type** - The type of error that occurred.
  - **explanation** - The explanation of the error that occurred.
  - **correctiveAction** - Suggestions for correcting the error.
  - **tagName** - The name of the tag where the error occurred.
  - **completeTag** - The complete tag where the error occurred.

Optional Parameters:

- **None**.

Example Usage and Output

```
<html>

<head>
  <title>Processing Error</title>
</head>

<body background="white">
```

```
<H2 align="center">Processing Error</H2>

<p align="center">
<font color="#ff0000">
Error: <mgiGetErrorParameter name="explanation">
</font></p>

</body>
</html>
```

In this example, a custom error page displays the error explantion only. The page might look like:

Processing Error

The "fileLocation" parameter is required for the <mgiIncludeFile> tag.

Suggested Usage

- Custom Error Pages

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiGetFileInfo Tag

Tag Behavior

Use the mgiGetFileInfo tag to display information about a file.

Tag Syntax

The mgiGetFileInfo tag has one required parameter and five optional parameters. The tag form is:

```
<mgiGetFileInfo info="Type" fileLocation="File Path"
format="Format" html="HTML" text="String" units="Units">
```

Required Parameters:

- **info** - Info is the type of file information to display. Valid info types are:
  - **creationDate** - The creation date of the specified file.
  - **creationTime** - The creation time of the specified file.
  - **lastModifiedDate** - The last modified date of the specified file.
  - **lastModifiedTime** - The last modified date of the specified file.
  - **size** - The size of the specified file.
  - **exists** - Whether the specified file exists. If the file does exist, the value "True" displays. If the file does not exists, the value "False" displays.
  - **isAFolder** - Whether the specified file name is a folder. If the specified name is a folder, the value "True" displays. If the specified name is not a folder, the value "False" displays.

Optional Parameters:

- **fileLocation** - The fileLocation is the relative path to the file to query for information. If the fileLocation parameter is not included, the file information of the page containing the mgiGetFileInfo tag is displayed.
- **format** - The format is the type of date and time information to display. If the creationDate or lastModifiedDate is displayed, then any date format from the mgiDate tag may be used. If the creationTime or lastModifiedTime parameter is displayed, then any time format from the mgiTime tag may be used. The format parameter has no effect if the size is displayed. You may include multiple format parameters in one mgiGetFileInfo tag.

- **html** - The html is the HTML code and text that is decoded and displayed relative to the specified date or time formats. Any HTML in the value of the html parameter should already be encoded (e.g., quotation marks should be entered as "&quot;"). You may include multiple html parameters in one mgiGetFileInfo tag.
- **text** - The text is the string of characters that is decoded and displayed relative to the specified date or time formats. Text in the value of the text parameter is not decoded prior to display. You may include multiple text parameters in one mgiGetFileInfo tag.
- **units** - The units is the unit of measurement displayed for the file size. The default value is "bytes". Valid units values are:
    - b
    - bytes
    - k
    - kilo
    - kilobytes
    - m
    - mega
    - megabytes
    - g
    - giga
    - gigabytes
    - t
    - tera
    - terabytes

Example Usage and Output

```
<mgiGetFileInfo info="size" units="k">
```

In this example, the size of the HTML file containing this mgiGetFileInfo tag is displayed in kilo-bytes.

```
<mgiGetFileInfo fileLocation="../contact.html"
info="lastModifiedDate">
```

In this example, the last modified date of the "contact.html" file is displayed.

Suggested Usage

- Document, Image and Software Downloads

- Content Updates

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiGuestBook Tag

Tag Behavior

Use the mgiGuestbook tag to display formatted form submissions at a designated location on a page.

Tag Syntax

The mgiGuestbook tag has a beginning tag with one required parameter and no optional parameters, a body, and an ending tag. The tag form is:

```
<mgiGuestbook fileLocation="File Path">
MGI Tags and Text
</mgiGuestbook>
```

The form of the guestbook insertion marker is:

```
<!-- Guestbook Insertion Marker -->
```

Required Parameters:

- **fileLocation -** The fileLocation is the relative path to the file containing the Guestbook Insertion Marker (i.e., where the guestbook entry is written.)

Optional Parameters:

- **None.**

Guestbook Insertion Marker:

- The guestbook insertion marker is placed in a separate file at the location where each guestbook entry should be written. Guestbook entries are written at the beginning of the insertion marker such that the most recent entries appear at the top of the list.

Example Usage and Output

```
<mgiGuestbook fileLocation="GuestbookLog.html">
<table width="450" border="1">
<tr>
<td>
<P><mgiPostArgument name="name"> wrote to us and said:</P>
```

```
<P><font color="#006600">
<mgiPostArgument name="comments"></font></P>
<P><mgiPostArgument name="name"> can be reached at
<B><mgiPostArgument name="email"></B>
</td></tr>
</table>
</mgiGuestbook>
```

In this example, guestbook entries are written to the GuestbookLog.html file and each entry will appear in the following format at the location of the guestbook insertion marker:

John Doe wrote to us and said: I really like your guestbook. John Doe can be reached at **doe@someplace.com**

When the next entry is submitted, it will appear above the previous entry:

Sarah Smith wrote to us and said: I think your guestbook is great, too! Sarah Smith can be reached at **ssmith@domain.com**
John Doe wrote to us and said: I really like your guestbook. John Doe can be reached at **doe@someplace.com**

Suggested Usage

- Guestbooks
- Appending Information to a File

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiGuestbookDB Tag

Tag Behavior

Use the mgiGuestbookDB tag to create a guestbook with entries stored in a database.

Tag Syntax

The mgiGuestbookDB tag has three modes. Each mode has different required and optional parameters. The modes of mgiGuestbookDB are:

- **submit** - Submits the guestbook entry to the database.
- **display** - Displays the guestbook entries from the database.
- **submitAndDisplay** - Submits the guestbook entry to the database and displays the guestbook entries from the database.

Submit Mode

The submit mode of the mgiGuestBookDB tag has a beginning tag with two required parameters and three optional parameters, a body, and an ending tag. The tag form is:

```
<mgiGuestbookDB name="myGuestbook" mode="submit"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
MGI Tags and Text
</mgiGuestbookDB>
```

Required Parameters:

- **name** - The name of the guestbook.
- **mode** - The mode is the function that the mgiGuestbookDB tag performs. In "**submit**" mode, the mgiGuestbookDB tag adds the guestbook entry to the guestbook database. In submit mode, the body of the mgiGuestbookDB tag is required and cannot be empty.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, counter information will be stored in the

specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Display Mode

The display mode of the mgiGuestBookDB tag has a beginning tag with one required parameter and seven optional parameters, and an ending tag. The tag form is:

```
<mgiGuestbookDB name="myGuestbook" mode="display"
resultsPerPage="Whole Number" page="Whole Number"
resultsVariableName="Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
</mgiGuestbookDB>
```

Required Parameters:

- **name** - The name of the guestbook.

Optional Parameters:

- **mode** - The mode is the function that the mgiGuestbookDB tag performs. In "**display**" mode, the mgiGuestbookDB tag displays guestbook entries from the guestbook database at the location of the mgiGuestbookDB tag. "Display" is the default mode of the mgiGuestbookDB tag.
- **resultsPerPage** - The resultsPerPage is the number of guestbook entries to display. The default value is 25.
- **page** - The page is the set of guestbook entries to display. For example, if the resultsPerPage is set to "25", then page 1 displays entries 1 to 25 and page 2 displays entries 26 to 50. The default value is 1.
- **resultVariableName** - The resultVariableName is the name prepended to the guestbook result variables that contain information about the guestbook results. You may choose any name for the result variable prefix. To display guestbook result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiGuestbookDB tag.
    - **resultvar_ResultCount -** A page variable with the total number of entries in the guestbook.

    - **resultvar_FirstIndex** - A page variable with the index number of the first entry shown on the page.
    - **resultvar_LastIndex** - A page variable with the index number of the last entry shown on the page.
    - **resultvar_PrevPage** - A page variable with the number of the page preceding the current page. The value of this variable is zero (0) if the current page is 1.
    - **resultvar_Page** - A page variable with the number of the page currently displaying.
    - **resultvar_NextPage** - A page variable with the number of the page following the current page. The value of this variable is zero (0) if the current page is the last page.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, counter information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

SubmitAndDisplay Mode

The submitAndDisplay mode of the mgiGuestBookDB tag has a beginning tag with two required parameters and six optional parameters, a body, and an ending tag. The tag form is:

```
<mgiGuestbookDB name="myGuestbook" mode="display"
resultsPerPage="Whole Number" page="Whole Number"
resultsVariableName="Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
MGI Tags and Text
</mgiGuestbookDB>
```

Required Parameters:

- **name** - The name of the guestbook.
- **mode** - The mode is the function that the mgiGuestbookDB tag performs. In "**submitAndDisplay**" mode, the mgiGuestbookDB tag adds the guestbook entry to the guestbook database and displays guestbook entries from the guestbook database at the location of the mgiGuestbookDB tag. In submitAndDisplay mode, the body of the

mgiGuestbookDB tag is required and cannot be empty.

Optional Parameters:

- **resultsPerPage** - The resultsPerPage is the number of guestbook entries to display. The default value is 25.
- **page** - The page is the set of guestbook entries to display. For example, if the resultsPerPage is set to "25", then page 1 displays entries 1 to 25 and page 2 displays entries 26 to 50. The default value is 1.
- **resultVariableName** - The resultVariableName specifies the type of information that is saved in a page variable. If the resultVariableName parameter value is "**resultvar**", then the following page variables are created and available anywhere on the page after the mgiGuestbookDB tag (Use the mgiGet tag to display variable information.)
    - **resultvar_ResultCount** - A page variable with the total number of entries in the guestbook.
    - **resultvar_FirstIndex** - A page variable with the index number of the first entry shown on the page.
    - **resultvar_LastIndex** - A page variable with the index number of the last entry shown on the page.
    - **resultvar_PrevPage** - A page variable with the number of the page preceding the current page. The value of this variable is zero (0) if the current page is 1.
    - **resultvar_Page** - A page variable with the number of the page currently displaying.
    - **resultvar_NextPage** - A page variable with the number of the page following the current page. The value of this variable is zero (0) if the current page is the last page.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, counter information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Example Usage and Output

Submit Mode

```
<mgiGuestbookDB name="myGuestbook" mode="submit">
<p><b><mgiPostArgument name="Name"> said:</b></p>
<p><mgiPostArgument name="Comments"></p>
<p><hr></p>
</mgiGuestbookDB>
```

In this example, a guestbook entry is added to the guestbook database from a form submission.

Display Mode

```
<mgiGuestbookDB name="myGuestbook" mode="display"
resultsPerPage="20" page="2">
</mgiGuestbookDB>
```

In this example, the guestbook entries from 21 to 40 are displayed. If the guestbook entries were submitted as in the submit mode example, they would appear as follows:

Ralph Barber said:

I prefer your products because they are easy to learn and use

---

Marcia Simpson said:

Does your product come in blue?

SubmitAndDisplay Mode

```
<mgiGuestbookDB name="myGuestbook" mode="submitAndDisplay"
resultsPerPage="20" page="2">
<p><b><mgiPostArgument name="Name"> said:</b></p>
<p><mgiPostArgument name="Comments"></p>
<p><hr></p>
</mgiGuestbookDB>
```

In this example, new guestbook entries from a form submission are submitted to the guestbook database as in the submit mode example and displayed as in the display mode example.

Suggested Usage

- Database-Driven Guestbooks

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiIf Tag

Tag Behavior

Use the mgiIf tag to perform a conditional comparison and display a result based upon the outcome of the comparison (see also mgiInlineIf and mgiSwitch).

Tag Syntax

The mgiIf tag has two forms. The mgiIf tag can be used as a beginning tag with three required parameters and two optional parameters, a "true" body, an mgiElse tag, a "false" body, and an ending mgiIf tag OR as a beginning tag with three required parameters and two optional parameters, a "true" body, and an ending tag. The tag forms are:

```
<mgiIf lhs="Comparison Value" relationship="Relationship"
rhs="Comparison Value" order="Order"
caseSensitive="Yes/No">
Value if comparison is true
<mgiElse>
Value if comparison is false
</mgiIf>
```

and

```
<mgiIf lhs="Comparison Value" relationship="Relationship"
rhs="Comparison Value" order="Order"
caseSensitive="Yes/No">
Value if comparison is true
</mgiIf>
```

Required Parameters:

- **lhs** - The lhs is the first value tested in the conditional comparison (i.e., the "left hand side" of the comparison). The lhs value must be a number for the greaterThan, greaterThanOrEqualTo, lessThan, and lessThanOrEqualTo relationships. In order to use special reserved characters such as tabs, backslashes, and ASCII characters, you must use the appropriate Escaped String format.
- **relationship** - The relationship is the type of comparison to perform. You may use the relationship name or symbol. Valid relationships include:
  - **lessThan (<)**
  - **lessThanOrEqualTo (<=)**

    - **greaterThan (>)**
    - **greaterThanOrEqualTo (>=)**
    - **equals (=)**
    - **doesNotEqual (<>)**
    - **contains**
    - **doesNotContain**
    - **beginsWith**
    - **doesNotBeginWith**
    - **endsWith**
    - **doesNotEndWith**
    - **isEmpty -** The rhs parameter is not required for this relationship.
    - **isNotEmpty -** The rhs parameter is not required for this relationship.
- **rhs -** The rhs is the second value tested in the conditional comparison (i.e., the "right hand side" of the comparison). In order to use special reserved characters such as tabs, backslashes, and ASCII characters, you must use the appropriate Escaped String format.

Optional Parameters:

- **order -** The order determines the order of values in the conditional comparison. If the order parameter value is "**alphabetical**", then values are compared alphabetically as ascii strings (i.e., characters appear alphabetically in the order of non-printable characters, punctuation, numbers, capital letters, and lower-case letters). Therefore, if two numbers are compared alphabetically as strings, they are ordered from left to right alphabetically (e.g., "550" appears alphabetically before "60"). If the order parameter value is "**numerical**", then number values are compared numerically from the decimal. If either value is not a number in a numerical order, then the values are compared alphabetically as strings. The default value is "numerical".
- **caseSensitive -** The caseSensitive parameter specifies whether the lhs and rhs values are checked for case-sensitivity. If the caseSensitive parameter value is "**Yes**", the lhs and rhs values are checked for case-sensitivity. If the caseSensitive parameter values is "**No**", the lhs and rhs values are not checked for case-sensitivity. The default value is "No".

Example Usage and Output

```
<mgiIf lhs={mgiGet name="Score"}
relationship="greaterThanOrEqualTo" rhs="11">
```

```
Your quiz score indicates that you are a Growth Investor.
Growth Investors are willing to take risks and prefer short
term gain to long term gain.

<mgiElse>

<mgiIf lhs={mgiGet name="Score"}
relationship="greaterThanOrEqualTo" rhs="7">

Your quiz score indicates that you are a Moderate Investor.
Moderate Investors will take well-researched risks and can
handle fluctuations in the market.

<mgiElse>

Your quiz score indicates that you are a Conservative Investor.
Conservative Investors prefer minimal risk options and growth
over a long term.

</mgiIf>

</mgiIf>
```

In this example, a form with quiz questions has been created. The value of each quiz answer is a number. On the processing page of the form, an mgiMath tag is used to calculate the sum of all 5 questions and that value is set in a page variable named "Score". The variable value is then compared to the values in the "rhs" parameters and the result of the conditional comparison displays.

If a visitor's score is greater than or equal to 11, then the following text will display:

> Your quiz score indicates that you are a Growth Investor. Growth Investors are willing to take risks and prefer short term gain to long term gain.

If a visitor's score is greater than or equal to 7, then the following text will display:

> Your quiz score indicates that you are a Moderate Investor. Moderate Investors will take well-researched risks and can handle fluctuations in the market.

If a visitor's score is greater than or equal to 5, than the following text will display:

> Your quiz score indicates that you are a Conservative Investor. Conservative Investors prefer minimal risk options and growth over a long term.

Suggested Usage

- Conditional Comparisons

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiIncludeFile Tag

Tag Behavior

Use the mgiIncludeFile tag to insert information from another file into the current page. (see also mgiIncludeHTTP)

Tag Syntax

The mgiIncludeFile tag has one required parameter and one optional parameter. The tag form is:

```
<mgiIncludeFile fileLocation="File Path"
verifyFileExistence="yes/no">
```

Required Parameters:

- **fileLocation** - The fileLocation is the relative path of the file to display.

Optional Parameters:

- **verifyFileExistence** - Whether or not to display an error page if the specified file does not exist. Valid values are "yes" and "no". The default value is "yes".

Example Usage and Output

```
<mgiIncludeFile fileLocation="file2.html">
```

In this example, file2.html will be inserted into the file containing the mgiIncludeFile tag at the location of the mgiIncludeFile tag.

```
<mgiIncludeFile fileLocation="notafile.txt"
verifyFileExistence="no">
```

In this example, notafile.txt does not exist. Since the verifyFileExistence parameter is set equal to "no", an error page will not be displayed.

Suggested Usage

- Common Headers and Footers

- Navigation

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiIncludeHTTP Tag

Tag Behavior

Use the mgiInclude HTTP tag to retrieve and display web page headers and content via HTTP. The mgiIncludeHTTP supports both the GET and POST methods. (see also mgiIncludeFile)

Tag Syntax

The mgiIncludeHTTP tag has one required parameter and six optional parameters. The tag form is:

```
<mgiIncludeHTTP url="URL" data="Type" method="Method"
postArgumentName="Name" postArgumentValue="Value"
username="Login" password="Password">
```

Required Parameters:

- **url** - The url is the relative or absolute URL to the web page that you wish to include. The mgiIncludeHTTP tag cannot include data from a secure URL (e.g., URLs beginning with HTTPS).

Optional Parameters:

- **data** - The data is the type of information to display from the included page. If the data parameter value is "**headers**", then the HTTP headers from the included page are displayed. If the data parameter value is "**content**", then the content of the included page is displayed. If the data parameter value is "**headersAndContent**", then the headers and the content of the included page are displayed. The default value is content.
- **method** - The method is the method used to retrieve the page. If the method parameter value is "get", then the get method is used to retrieve the page. If the method parameter value is "post", then the post method is used to retrieve the page. The default value is get.
- **postArgumentName** - The postArgumentName is the name of the post argument sent with the page request. You may include multiple postArgumentName parameters in one mgiIncludeHTTP tag. Each postArgumentName parameter must have a corresponding postArgumentValue parameter that is entered after the postArgumentName parameter.
- **postArgumentValue** - The postArgumentValue is the value of the post argument sent

with the page request. You may include multiple postArgumentValue parameters in one mgiIncludeHTTP tag. Each postArgumentValue parameter must have a corresponding postArgumentName parameter that is entered before the postArgumentValue parameter.

- **username** - The username is the username to send with the page request if the data is protected by Basic HTTP authorization.
- **password** - The password is the password to send with the page request if the data is protected by Basic HTTP authorization.

Example Usage and Output

Basic Include

```
<mgiIncludeHTTP url="http://www.nytimes.com" data="content">
```

In this example, the content from the New York Time's home page is displayed. Specific information can then be extracted using mgiString tags, for example.

Base HREF

```
<html>
<head>
<title>Page Title</title>
<base href="http://www.cnn.com/">
</head>
<body>
<mgiIncludeHTTP url="http://www.cnn.com" data="content">
</body>
</html>
```

When you include the content of a page that is not located in the same directory as the page containing the mgiIncludeHTTP tag, any elements of the page with relative file references (e.g., images) will likely break. To "fix" those elements, include the base HTML tag in the head of your page. In the href parameter of the base tag, list the URL to pre-pend to the relative file references in the page. In this example, CNN's home page elements with a relative file reference are corrected with the base tag.

Suggested Usage

- Including Files Located Outside the Server

The mgiIncrement Tag

Tag Behavior

Use the mgiIncrement tag to increment or decrement a numeric variable by a specified value.

Tag Syntax

The mgiIncrement tag has one required parameter and two optional parameters. The tag form is:

```
<mgiIncrement variableName="Name" scope="Scope"
value="Number">
```

Required Parameters:

- **variableName -** The variableName is the name of the variable to increment. If the variable has already been created, it must be blank or contain a number. If the variable has not been created, the mgiIncrement tag will create a new variable with the variableName and set the specified value.

Optional Parameters:

- **scope -** The scope is the scope of the variable to increment. If the scope parameter is set to "**page**", then a page variable with the specified name will be incremented by the specified value. If the scope is set to "**site**", then a site variable with the specified name will be incremented by the specified value. The default scope is "page".
- **value -** The value is amount to increment the variable. Use a negative number to decrement the value.

Example Usage and Output

```
<mgiIncrement variableName="Count" scope="Page" value="1">
```

In this example, a page variable is incremented by one. If the current value of the "Count" variable is "33", the mgiIncrement tag will set the value to "34" when it is processed.

Suggested Usage

- Loops
- Conditional Comparisons

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiInfoDumper Tag

Tag Behavior

Use the mgiInfoDumper tag for testing purposes to view MGI and page header information.

Tag Syntax

The mgiInfoDumper tag has no required parameters and no optional parameters. The tag form is:

```
<mgiInfoDumper>
```

Required Parameters:

- **None.**

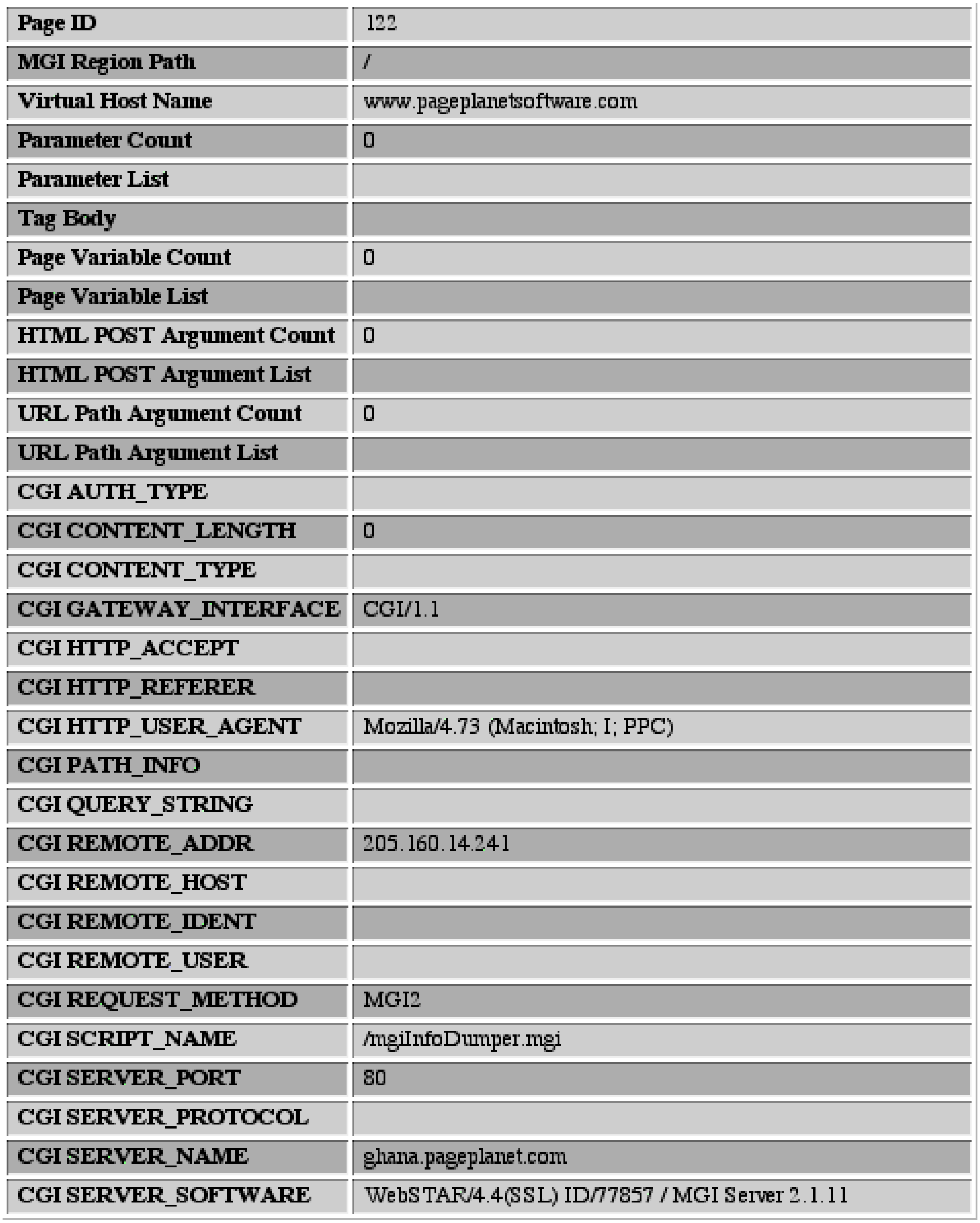
Optional Parameters:

- **None.**

Example Usage and Output

```
<mgiInfoDumper>
```

The mgiInfoDumper tag creates a table of MGI and page header information:



Page ID	122
MGI Region Path	/
Virtual Host Name	www.pageplanetsoftware.com
Parameter Count	0
Parameter List	
Tag Body	
Page Variable Count	0
Page Variable List	
HTML POST Argument Count	0
HTML POST Argument List	
URL Path Argument Count	0
URL Path Argument List	
CGI AUTH_TYPE	
CGI CONTENT_LENGTH	0
CGI CONTENT_TYPE	
CGI GATEWAY_INTERFACE	CGI/1.1
CGI HTTP_ACCEPT	
CGI HTTP_REFERER	
CGI HTTP_USER_AGENT	Mozilla/4.73 (Macintosh; I; PPC)
CGI PATH_INFO	
CGI QUERY_STRING	
CGI REMOTE_ADDR	205.160.14.241
CGI REMOTE_HOST	
CGI REMOTE_IDENT	
CGI REMOTE_USER	
CGI REQUEST_METHOD	MGI2
CGI SCRIPT_NAME	/mgiInfoDumper.mgi
CGI SERVER_PORT	80
CGI SERVER_PROTOCOL	
CGI SERVER_NAME	ghana.pageplanet.com
CGI SERVER_SOFTWARE	WebSTAR/4.4(SSL) ID/77857 / MGI Server 2.1.11

Suggested Usage

- Testing

The mgiInlineDoNotProcess Tag

Tag Behavior

Use the mgiInlineDoNotProcess tag to prevent the MGI code within post arguments, path arguments, variables, and includes from being processed. The code inserted by a post argument, path argument, variable or include in the data parameter of the mgiInlineDoNotProcess tag is **not** removed from the page before it is served. (see also mgiDoNotProcess and mgiComment)

Tag Syntax

The mgiInlineDoNotProcess tag has one required parameter and no optional parameters,. The tag form is:

```
<mgiInlineDoNotProcess data="Data">
```

Required Parameters:

- data - The data is the text, HTML, or MGI code that should not be processed. Any tag embedded in the data parameter will be processed, but the result of that tag (e.g., the value of the post argument or the data from an included file) will not be re-processed by MGI.

Optional Parameters:

- **None**.

Example Usage and Output

```
<mgiInlineDoNotProcess data={mgiPostArgument name="Comments"}>
```

The mgiInlineDoNotProcess tag prevents any MGI code in the "Comments" post argument from being processed.

Suggested Usage

- Dynamic Forms

- Conditional Comparisons
- Site Development and Testing
- Site Security

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiInlineIf Tag

Tag Behavior

Use the mgiInlineIf tag to perform a conditional comparison and display a result based upon the outcome of the comparison (see also mgiIf and mgiSwitch).

Tag Syntax

The mgiInlineIf tag has four required parameters and three optional parameter. The tag form is:

```
<mgiInlineIf lhs="Comparison Value" relationship="Relationship"
rhs="Comparison Value" then="True" else="False" order="Order"
caseSensitive="Yes/No">
```

Required Parameters:

- **lhs** - The lhs is the first value tested in the conditional comparison (i.e., the "left hand side" of the comparison). The lhs value must be a number for the greaterThan, greaterThanOrEqualTo, lessThan, and lessThanOrEqualTo relationships. In order to use special reserved characters such as tabs, backslashes, and ASCII characters, you must use the appropriate Escaped String format.
- **relationship** - The realtionship is the type of comparison to perform. You may use the relationship name or symbol. Valid relationships include:
    - **lessThan (<)**
    - **lessThanOrEqualTo (<=)**
    - **greaterThan (>)**
    - **greaterThanOrEqualTo (>=)**
    - **equals (=)**
    - **doesNotEqual (<>)**
    - **contains**
    - **doesNotContain**
    - **beginsWith**
    - **doesNotBeginWith**
    - **endsWith**
    - **doesNotEndWith**
    - **isEmpty** - The rhs parameter is not required for this relationship.
    - **isNotEmpty** - The rhs parameter is not required for this relationship.

- **rhs** - The rhs is the second value tested in the conditional comparison (i.e., the "right hand side" of the comparison). In order to use special reserved characters such as tabs, backslashes, and ASCII characters, you must use the appropriate Escaped String format.
- **then** - Then then value is the text that displays if the conditional comparison is true.

Optional Parameters:

- **else** - Then else value is the text that displays if the conditional comparison is false.
- **order** - The order determines the order of values in the conditional comparison. If the order parameter value is "**alphabetical**", then values are compared alphabetically as ascii strings (i.e., characters appear alphabetically in the order of non-printable characters, punctuation, numbers, capital letters, and lower-case letters). Therefore, if two numbers are compared alphabetically as strings, they are ordered from left to right alphabetically (e.g., "550" appears alphabetically before "60"). If the order parameter value is "**numerical**", then number values are compared numerically from the decimal. If either value is not a number in a numerical order, then the values are compared alphabetically as strings. The default value is "numerical".
- **caseSensitive** - The caseSensitive parameter specifies whether the lhs and rhs values are checked for case-sensitivity. If the caseSensitive parameter value is "**Yes**", the lhs and rhs values are checked for case-sensitivity. If the caseSensitive parameter values is "**No**", the lhs and rhs values are not checked for case-sensitivity. The default value is "No".

Example Usage and Output

```
<mgiInlineIf lhs={mgiGetCookie name="Subscription"}
relationship="contains" rhs={mgiDate format="longMonth"}
then="Renew your subscription today!"
else="Thank you for your support.">
```

In this example, a visitor's subscription cookie is checked for the current date. If their subscription contains the current month, then the visitor is prompted to renew, otherwise a thank you message is displayed.

Suggested Usage

- Conditional Comparisons
- Quizzes

The mgiInlineString Tag

Tag Behavior

Use the mgiInlineString tag to perform operations on text. (see also mgiString).

Tag Syntax

The mgiInlineString tag has twenty modes. Each mode has different required and optional parameters. The modes of mgiInlineString are:

- **urlEncode** - The urlEncode mode replaces URL reserved characters with encoded versions of those characters.
- **urlDecode** - The urlDecode mode replaces encoded URL characters with URL reserved characters.
- **htmlEncode** - The htmlEncode mode replaces HTML reserved characters with an encoded version of those characters.
- **htmlDecode** - The htmlDecode mode replaces encoded HTML characters with HTML reserved characters.
- **toUpper** - The toUpper mode changes the string to all capital letters.
- **toLower** - The toLower mode changes the string to all lowercase letters.
- **substring** - The substring mode extracts and displays a string of characters.
- **replace** - The replace mode finds a string and replaces it with another string.
- **insert** - The insert mode inserts a new string of text at a specified location in the string.
- **append** - The append mode adds characters to the end of the string.
- **prepend** - The prepend mode adds characters to the beginning of the string.
- **trimLeft** - The trimLeft mode removes whitespace from the beginning of a string.
- **trimRight** - The trimRight mode removes whitespace from the end of a string.
- **trim** - The trim function removes whitespace from the beginning and end of a string.
- **getLength** - The getLength mode displays the total length of the string.
- **split** - The split mode extracts values with a delimiter.
- **regularExpression** - The regularExpression mode searches a string for a regular expression pattern.

- **getWordCount** - The getWordCount mode displays the total number of words in a string.
- **getCharacterCount** - The getCharacterCount mode displays the total number of characters in a string that are not carriage returns or linefeeds.
- **noOperation** - The noOperation mode does not change the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

URLEncode Mode

The urlEncode mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="urlEncode"
quoteResult="Yes/No">
```

Required Parameter:

- **string** - The string is the string of text to encode.
- **mode** - The mode is the function that the mgiInlineString tag performs. In "**urlEncode**" mode, the mgiInlineString tag replaces all URL reserved characters with an encoded version of the character (e.g., spaces in the URL are replaced with %20 characters).

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

URLDecode Mode

The urlDecode mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="urlDecode"
quoteResult="Yes/No">
```

Required Parameter:

- **string** - The string is the string of text to decode.
- **mode** - The mode is the function that the mgiInlineString tag performs. In "**urlDecode**" mode, the mgiInlineString tag replaces all encoded characters with URL reserved characters (e.g., %20 characters are replaced with spaces).

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

HTMLEncode Mode

The htmlEncode mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="htmlEncode"
quoteResult="Yes/No">
```

Required Parameter:

- **string -** The string is the string of text to encode.
- **mode -** The mode is the function that the mgiInlineString tag performs. In "**htmlEncode**" mode, the mgiInlineString tag replaces all HTML reserved characters with an encoded version of the character (e.g., quotation marks are replaced with &quot; characters).

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

HTMLDecode Mode

The htmlDecode mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="htmlDecode"
quoteResult="Yes/No">
```

Required Parameter:

- **string -** The string is the string of text to decode.
- **mode -** The mode is the function that the mgiInlineString tag performs. In "**htmlDecode**" mode, the mgiInlineString tag replaces all encoded characters with HTML reserved characters (e.g., &quot; characters are replaced with quotation marks).

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

ToUpper Mode

The toUpper mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="toUpper"
quoteResult="Yes/No">
```

Required Parameter:

- **string -** The string is the string of text to change.
- **mode -** The mode is the function that the mgiInlineString tag performs. In "**toUpper**" mode, the mgiInlineString tag changes the string to all capital letters.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

ToLower Mode

The toLower mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="toLower"
quoteResult="Yes/No">
```

Required Parameter:

- **string -** The string is the string of text to change.
- **mode -** The mode is the function that the mgiInlineString tag performs. In "**toLower**" mode, the mgiInlineString tag changes the string to all lowercase letters.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

Substring Mode

The substring mode of the mgiInlineString tag has four required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="substring"
beginningIndex="Number" length="Number" quoteResult="Yes/No">
```

Required Parameter:

- **string -** The string is the string of text to search.
- **mode -** The mode is the function that the mgiInlineString tag performs. In "**substring**" mode, the mgiInlineString tag extracts and displays a string of characters specified by the numeric location of the first character in the string and the length of the string.
- **beginningIndex -** The beginningIndex is an integer that indicates the numeric location of the first character to extract and display. The value of the beginningIndex value must be less than the total length of the string. For example, to extract the numeric month from the string "19991025", the beginningIndex would be "5".
- **length -** The length is an integer that indicates the length in characters of the string including the beginningIndex. For example, to extract the numeric month from the string "19991025", the beginningIndex would be "5" and the length would be "2".

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

Replace Mode

The replace mode of the mgiInlineString tag has four required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="replace"
stringToReplace="String" replacementString="String"
quoteResult="Yes/No">
```

Required Parameter:

- **string** - The string is the string of text to search.
- **mode** - The mode is the function that the mgiInlineString tag performs. In "**replace**" mode, the mgiInlineString tag finds a string and replaces it with another string.
- **stringToReplace** - The stringToReplace is the text that will be found and replaced by the replacement string. To find the escape characters carriage returns, line feeds, and tabs, use the appropriate Escaped String format.
- **replacementString** - The replacementString is the text that will replace any string found by the stringToReplace parameter. To find the escape characters carriage returns, line feeds, and tabs, use the appropriate Escaped String format.

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

Insert Mode

The insert mode of the mgiInlineString tag has four required parameters and one optional parameterg. The tag form is:

```
<mgiInlineString string="String"  mode="insert"
stringToInsert="String" index="Integer" quoteResult="Yes/No">
```

Required Parameter:

- **string** - The string is the string of text to change.
- **mode** - The mode is the function that the mgiInlineString tag performs. In "**insert**" mode, the mgiInlineString tag inserts string of characters at a specified location in the string.
- **stringToInsert** - The stringToInsert is the text that is inserted at the location specified in the index parameter. To insert escape characters such as carriage returns, line feeds, and tabs, use the appropriate Escaped String format.
- **index** - The index is the integer of the character location where the string is inserted. For example, in the string "abcdef", "c" is at the 3 index position.

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

Append Mode

The append mode of the mgiInlineString tag has three required parameters and one optional parameterg. The tag form is:

```
<mgiInlineString string="String" mode="insert"
stringToAppend="String" quoteResult="Yes/No">
```

Required Parameter:

- **string** - The string is the string of text to change.
- **mode** - The mode is the function that the mgiInlineString tag performs. In "**append**" mode, the mgiInlineString tag adds characters to the end of the string.
- **stringToAppend** - The stringToAppend is the text that is appended to the end of the string. To insert escape characters such as carriage returns, line feeds, and tabs, use the appropriate Escaped String format.

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

Prepend Mode

The prepend mode of the mgiInlineString tag has three required parameters and one optional parameterg. The tag form is:

```
<mgiInlineString string="String" mode="insert"
stringToPrePend="String" quoteResult="Yes/No">
```

Required Parameter:

- **string** - The string is the string of text to change.
- **mode** - The mode is the function that the mgiInlineString tag performs. In "**prepend**" mode, the mgiInlineString tag adds characters to the beginning of the string.
- **stringToPrepend** - The stringToPrepend is the text that is prepended to the beginning of the string. To insert escape characters such as carriage returns, line feeds, and tabs, use the appropriate Escaped String format.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

TrimLeft Mode

The trimLeft mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="trimLeft"
quoteResult="Yes/No">
```

Required Parameter:

- **string -** The string is the string of text to trim.
- **mode -** The mode is the function that the mgiInlineString tag performs. In "**trimLeft**" mode, the mgiInlineString tag removes all whitespace (including spaces, tabs and returns) from the beginning of a string.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

TrimRight Mode

The trimRight mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="trimRight"
quoteResult="Yes/No">
```

Required Parameter:

- **string -** The string is the string of text to trim.
- **mode -** The mode is the function that the mgiInlineString tag performs. In "**trimRight**" mode, the mgiInlineString tag removes all whitespace (including spaces, tabs and returns) from the end of a string.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

Trim Mode

The trim mode of the mgiInlineString tag has a beginning tag with two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="trim"
quoteResult="Yes/No">
```

Required Parameter:

- **string** - The string is the string of text to trim.
- **mode** - The mode is the function that the mgiInlineString tag performs. In "**trim**" mode, the mgiInlineString tag removes all whitespace (including spaces, tabs and returns) from the beginning and end of a string.

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

GetLength Mode

The getLength mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="getLength"
quoteResult="Yes/No">
```

Required Parameter:

- **string** - The string is the string of text to search.
- **mode** - The mode is the function that the mgiInlineString tag performs. In "**getLength**" mode, the mgiInlineString tag displays the total length of the string.

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is

"No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

Split Mode

The split mode of the mgiInlineString tag has a beginning tag with four required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="split"
delimiter="Character" index="Integer" quoteResult="Yes/No">
```

Required Parameter:

- **string** - The string is the string of text to search.
- **mode** - The mode is the function that the mgiInlineString tag performs. In "**split**" mode, the mgiInlineString tag extracts values with a delimiter.
- **delimiter** - The delimiter is the character that separates each part of the string. To specify escape characters such as carriage returns, line feeds, and tabs as delimiters use the appropriate Escaped String format.
- **index** - The index is the number of the value to extract. For example, in the following comma-delimited list, an index of "2" will extract the second value of "Smith".

  ```
  Bob,Smith,919-225-6329,919-225-6330
  ```

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

RegularExpression Mode

The regularExpression mode of the mgiInlineString tag has a beginning tag with four required parameters and two optional parameterg. The tag form is:

```
<mgiInlineString string="String" mode="regularExpression"
regularExpression="RE Value" beginningIndex="Integer"
quoteResult="Yes/No" resultVariableName="Name">
```

Required Parameter:

- **string** - The string is the string of text to search.
- **mode** - The mode is the function that the mgiInlineString tag performs. In "**regularExpression**" mode, the mgiInlineString tag searches a string for a regular expression pattern.

- **regularExpression** - The regularExpression is the regular expression pattern to search for in the string. Create a regular expression pattern using regular expression symbols.
- **beginningIndex** - The beginningindex is an integer that indicates the numeric location in the string to begin searching. The value of the beginningIndex value must be less than the total length of the string. For example, a beginningIndex of "5" would start searching at the "e" in the string "abcdefghij".

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.
- **resultVariableName** - The resultVariableName is the name prepended to result variables that contain information about the search. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName prefix is "**resultvar**". For example, if your resultVariableName is "Email", then the page variable that contains the length of the full match would be named "Email_0_length". Result variables are created and available anywhere on the page after the mgiInlineString tag. The pound signs (#) represent the regular expression subgroup number (from 1 to 9). Zero (0) is the main subgroup that is matched. (Use the mgiGet tag to display variable information.)
    - **resultvar** - The full match returned by the regular expression.
    - **resultvar_0_index** - The beginning index of the full match.
    - **resultvar_0_length** - The length of the full match.
    - **resultvar_#** - The # subgroup of the match. # can be 1 through 9.
    - **resultvar_#_index** - The beginning index of the # subgroup match.
    - **resultvar_#_length** - The length of the # subgroup match.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

GetWordCount Mode

The getWordCount mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="getWordCount"
quoteResult="Yes/No">
```

Required Parameter:

- **string** - The string is the string of text to search.
- **mode** - The mode is the function that the mgiInlineString tag performs. In

"**getWordCount**" mode, the mgiInlineString tag displays the total number of words in a string

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

GetCharacterCount Mode

The getCharacterCount mode of the mgiInlineString tag has a beginning tag with two required parameters and one optional parameterg. The tag form is:

```
<mgiInlineString string="String" mode="getCharacterCount"
quoteResult="Yes/No">
```

Required Parameter:

- **string -** The string is the string of text to search.
- **mode -** The mode is the function that the mgiInlineString tag performs. In "**getCharacterCount**" mode, the mgiInlineString tag displays the total number of characters in a string that are not carriage returns or linefeeds.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

NoOperation Mode

The noOperation mode of the mgiInlineString tag has two required parameters and one optional parameter. The tag form is:

```
<mgiInlineString string="String" mode="noOperation"
quoteResult="Yes/No">
```

Required Parameter:

- **string -** The string is the string of text to use.
- **mode -** The mode is the function that the mgiInlineString tag performs. In "**noOperation**" mode, the mgiInlineString tag does not change the string. The

noOperation function can be used in conjunction with the quoteResult parameter to only enter quote marks around the string.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

Example Usage and Output

URLDecode Mode

```
<mgiInlineString string={mgiPathArgument name="Address"}
mode="urlDecode">
```

In this example, a path argument value is URL decoded for special characters.

Split Mode

```
<mgiInlineString string="blue,orange,green,purple,red"
mode="split" delimiter="," index="3">
```

In this example, the third value in a comma-delimited list of values is displayed. The third value from the list in this example is "green".

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

Suggested Usage

- Form Processing
- Dynamic Content

Return to the mgiInlineString Mode Menu | Examples and Output | Suggested Usage

The mgiInlineToken Tag

Tag Behavior

Use the mgiInlineToken tag to append a token to a URL. A token number can be used as a visitor's unique shopping basket identification. A token is composed of the visitor's IP number and time interval so that each token is absolutely unique.

Tag Syntax

The mgiInlineToken tag has one required parameter and three optional parameters. The tag form is:

```
<mgiInlineToken url="URL" mode="Mode" pathArgumentName="Name"
pathArgumentValue="Value">
```

Required Parameters:

- **url -** The url is the URL to tokenize.

Optional Parameters:

- **mode -** The mode is the function that the mgiToken tag performs. In "**addTokenOnly**" mode, the mgiToken tag appends the mgiToken path argument and unique token value to URLs. In "**addTokenWithPathArguments**" mode, the mgiToken tag appends the token and all path arguments to URLs. In "**addPathArgumentsOnly**" mode, the mgiToken tag appends all path arguments to URLs. The default value is "addTokenOnly".
- **pathArgumentName -** The pathArgumentName is the name of a path argument that is always appended to tokenized URLs. If this parameter is not followed by a pathArgumentValue parameter in the tag, it is ignored.
- **pathArgumentValue -** The pathArgumentValue is the value of the specified path argument that is always appended to tokenized URLs. If this parameter is not preceded by a pathArgumentName parameter in the tag, it is ignored.

Example Usage and Output

```
<mgiInlineToken url={mgiRequest name="Referrer"}>
```

In this example, a token is appended to the page referrer address.

Suggested Usage

- Shopping Baskets
- Affiliate Systems

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiJulianDay Tag

Tag Behavior

Use the mgiJulianDay tag to convert a date and time to a decimal Julian day or to convert a decimal Julian day to a specified date and/or time format. Julian dates can be used to sort and order date values.

Tag Syntax

The mgiJulianDay tag has no required parameters and thirteen optional parameters. The tag form is:

```
<mgiJulianDay month="Month" day="Day" year="Year"
hour="Hour" minute="Minute" second="Second" dateOnly="Yes/No"
relativeStep="Integer" relativeUnits="Units">
```

or

```
<mgiJulianDay julianDay="Julian Day" format="Format"
html="HTML" text="text">
```

Required Parameters:

- **None.**

Optional Parameters:

- **month** - The month is the months to convert to a julian date. The month format may be the full month name, the three letter abbreviation of the month or the numeric month value.
- **day** - The day is the day of the month to convert to a julian date.
- **year** - The year is the year to convert to a julian date
- **hour** - The hour is the military hour to convert to a julian date
- **minute** - The minute is the minutes to convert to a julian date.
- **second** - The second is the seconds to convert to a julian date.
- **ampm** - Adjusts the specified hour based on an "am" or "pm" value. If the parameter value is "**am**", the hour is converted to a julian date as entered. If the parameter value is "**pm**", 12 hours are added during the julian date conversion. The "hour" parameter is required and must contain a value between "0" and "11". The default value is "am".
- **dateOnly** - The dateOnly parameter specifies if only the date component of the julian

day number is displayed or if the date and time components of the julian day number are displayed. If the dateOnly parameter value is "Yes", then only the date component of the julian day number is displayed. If the dateOnly parameter value is "No", then the date and time components of the julian day number are displayed. The default value is "no".

- **relativeStep** - The relativeStep is the number of relative units to increase the julian day calculation. The default relative step is "0" (the current time.) If a relative step greater than 0 is specified, the relativeUnits parameter is required.
- **relativeUnits** - The relativeUnits is the type of increase to make in the julian day. Valid relative units are "**seconds**", "**minutes**", "**hours**", and "**days**".
- **julianDay** - The julianDay is the julian day number to convert to a standard date format.
- **format** - The format is the format of the date and time that is converted from a julian day number. You may include multiple format parameters in one mgiJulianDay tag. Valid formats are:
    - **LongWeekDay** - Displays the full week day without abbreviation (e.g. Tuesday).
    - **ShortWeekDay** - Displays the three-letter abbreviation of the week day (e.g. Sat).
    - **NumericWeekday** - Displays a numeric week day between Sunday (1) and Saturday (7).
    - **LongMonth** - Displays the full month without abbreviation (e.g.,November).
    - **ShortMonth** - Displays the three-letter abbreviation of the month (e.g. Jun).
    - **NumericMonth** - Displays a numeric month between January (1) and December (12).
    - **PaddedNumericMonth** - Displays a two-digit numeric month between January (01) and December (12).
    - **Day** - Displays a numeric day of the month as a number between 1 and 31.
    - **PaddedDay** - Displays a two-digit numeric day of the month as a number between 01 and 31.
    - **DaySuffix** - Displays the appropriate suffix for each numerical day (i.e., "st", "nd", "rd", or "th").
    - **JulianDayNumber** - Displays the integer julian day (e.g., 31 Jan 2000 is julian day number 2451574).
    - **DayOfYear** - Displays a numeric day between 1 and 366.
    - **PaddedDayOfYear** - Displays a two-digit numeric day between 001 and 366.
    - **WeekOfYear** - Displays the numeric week of the year from 1 to 52.
    - **PaddedWeekOfYear** - Displays the two-digit numeric week of the year from 01 to 52.

  - **LongYear** - Displays the four-digit year as a number between 1000 and 9999.
  - **ShortYear** - Displays the two-digit year as a number between 00 and 99.
  - **Date** - Displays the server's standard date format.
  - **Hour** - Displays the numeric hour between 1 and 12.
  - **PaddedHour** - Displays the numeric hour between 01 and 12.
  - **MilitaryHour** - Displays the numeric hour between 00 and 23.
  - **Minute** - Displays the minutes of an hour between 00 and 59.
  - **Second** - Displays the seconds of a minute between 00 and 59.
  - **AmPm** - Displays an "am" or "pm" label.
  - **Time** - Displays the server's standard time format.
- **html** - The html is the HTML code and text that is decoded and displayed relative to the specified date formats. Any HTML in the value of the html parameter should already be encoded (e.g., quotation marks should be entered as "&quot;"). You may include multiple html parameters in one mgiJulianDay tag.
- **text** - The text is the string of characters that is decoded and displayed relative to the specified date formats. Text in the value of the text parameter is not decoded prior to display. You may include multiple text parameters in one mgiJulianDay tag.

Example Usage and Output

<mgiJulianDay>

In this example, the current julian day with the time fraction is displayed: 2451845.96118056

```
<mgiJulianDay dateOnly="yes">
```

In this example, the current julian day without the time fraction is displayed: 2451845

```
<mgiJulianDay month="12" day="25" year="1999"
dateOnly="yes">
```

In this example, the julian day of December 25, 1999 is displayed without the time fraction: 2451537

```
<mgiJulianDay month="12" day="25" year="1999"
hour="17" minute="34" second="58">
```

In this example, the julian day of December 25, 1999 at 5:34:58 PM is displayed: 2451537.73261574

```
<mgiJulianDay julianDay="2455343" format="longMonth">
<mgiJulianDay julianDay="2455343" format="longYear">
```

In this example, the month and year of julian day 2455343 are displayed: May 2010

```
<mgiJulianDay julianDay="2455343.33439894789"
format="militaryHour">
```

In this example, the hour of julian date 2455343.33439894789 is displayed: 08

```
<mgiJulianDay relativeStep="27" relativeUnits="minutes">
```

In this example, the julian day 27 minutes from the time the mgiJulianDay tag was processed is displayed: 2451845.97993056

```
<mgiJulianDay relativeStep="44" relativeUnits="hours">
```

In this example, the jullian day 44 hours from the time the mgiJulianDay tag was processed is displayed: 2451847.79451389

Suggested Usage

- Creating Julian Day Values for Embedding
- Sorting by Dates

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiLink Tag

Tag Behavior

Use the mgiLink tag to create text or graphic links to a specific URL or referral URL (e.g. a dynamic "back" button).

Tag Syntax

The mgiLink tag has no required parameters and three optional parameters. The tag form is:

```
<mgiLink text="Link Text" image="Image Path" url="URL">
```

Required Parameters:

- **None.**

Optional Parameters:

- **text** - The text is the text that is displayed as the link. If neither text nor image is specified, the default text is "Go" when the url parameter is included and the default text is "Back" if the url parameter is not included.
- **image** - The image is the image that is displayed as the link. If the image and text parameters are included, the image will be displayed.
- **url** - The url is the target page of the link. The default URL is the referral URL.

Tag Behaviors

If you include no parameters, the text "Back" is displayed and linked to a referral URL.

1. If you only include the **text** parameter, the text is displayed and linked to the referral URL.
2. If you only include the **image** parameter, the image is displayed and linked to the referral URL.
3. If you include both the **text** and **url** parameters, the text is displayed and linked to the specified URL.
4. If you include both the **image** and **url** parameters, the image is displayed and linked to the specified URL.

Example Usage and Output

```
<mgiLink text="Guide Index" url="../index.html">
```

The mgiLink tag in this example will display a text link to the index page of the user guide.

Guide Index

Suggested Usage

- Dynamic "Back" Buttons

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiListFolder Tag

Tag Behavior

Use the mgiListFolder tag to list all files and folders located in a specified folder. The mgiListFolder tag does not include hidden files or aliases in the file list.

Tag Syntax

The mgiListFolder tag has no required parameters and four optional parameter. The tag form is:

```
<mgiListFolder folderLocation="File Path" delimiter="Character"
filter="Extension" resultVariableName="Name">
```

Required Parameters:

- **None**.

Optional Parameters:

- **folderLocation -** The folderLocation is the relative path to the folder to list. If the folderLocation is not provided, the folder containing the mgiListFolder tag is listed by default.
- **delimiter** - The delimiter is the character that separates each value returned file list. To use escape characters such as carriage returns, line feeds, and tabs as delimiters use the appropriate Escaped String format. The default delimiter is a comma.
- **filter** - The extension of the files to list. Each filter **must** begin with a wildcard, the asterisk (*). For example, the filter "*.jpg" will list all files with the .jpg extension and the filter "*.html" will list all files with the HTML extension. Multiple filter parameters may be included in one mgiListFolder tag. If the filter parameter is not included, all files and folders will be listed.
- **resultVariableName -** The resultVariableName is the name prepended to result variables that contain information about the files and folders that are listed. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiListFolder tag.
    - **resultvar_Length-** A page variable with the total number of files and folders in the list.

Example Usage and Output

```
<mgiListFolder>
```

In this example, the files and folders on the same level of the file system as the page containing the mgiListFolder tag are listed in comma-delimited format.

```
<mgiListFolder folderLocation="Images">
```

In this example, the files and folders in the "Images" folder are listed in comma-delimited format.

```
<mgiListFolder folderLocation="Articles" Delimiter=":">
```

In this example, the files and folders in the "Articles" folder are listed with a colon delimiter.

```
<pre>
<mgiListFolder folderLocation="PDFs" Delimiter="\r"
resultVariableName="Results">
</pre>

<p>Total Items: <mgiGet name="Results_Length">
```

In this example, the files and folders in the "PDFs" folder are separated by a carriage return in the list. The total number of items in the list is displayed after the "Total Items" label.

```
<mgiListFolder filter="*.gif">
```

In this example, all files from the folder containing the mgiListFolder tag with the .GIF extension in the name will be listed in comma-delimited format.

Suggested Usage

- Loops
- Image Displays

The mgiLoop Tag

Tag Behavior

Use the mgiLoop tag to repeat an action or display for a specified range of numbers or a specified list of values.

Tag Syntax

The mgiLoop tag has a beginning tag with no required parameters and seven optional parameters, a body and an ending tag. The tag forms are:

```
<mgiLoop first="Integer" last="Integer" step="Integer"
indexName="Set Name">
Information to Repeat and &mgiLoopIndex;
</mgiLoop>
```

or

```
<mgiLoop item="Item" item="Item" indexName="Set Name">
Information to Repeat and &mgiLoopIndex;
</mgiLoop>
```

or

```
<mgiLoop itemList="List" indexName="Set Name"
listDelimiter="Character">
Information to Repeat and &mgiLoopIndex;
</mgiLoop>
```

Required Parameters:

- **None**

Optional Parameters:

- **indexName -** The indexName is the name of the range of numbers or list of values in the mgiLoop tag. The index name is appended to the loop index marker (&mgiLoopIndex;) before the semi-colon to refer to a specific set of loop values when mgiLoop tags are nested.
- **first -** The first is the first integer in the range of loop values.
- **last -** The last is the last integer in the range of loop values.
- **step -** The step is an integer that represents the loop interval from the first number to

the last number in the range. For example, a step of "2" will loop every other number in the range beginning with the first number. The default step is "1" which loops every number in the range.

- **item -** The item is one item to be interated. Multiple item parameters may be included in one mgiLoop tag. Item parameters are looped in the order of the parameters in the mgiLoop tag.
- **itemList -** The itemList is the list of values to loop. Each value in the item list is separated by the character specified by the listDelimiter parameter.
- **listDelimiter -** The listDelimiter is the character that separates each value in the item list. To use escape characters such as carriage returns, line feeds, and tabs as delimiters use the appropriate Escaped String format. The default list delimiter is a comma.

The Loop Index Marker:

- **&mgiLoopIndex; -** The loop index marker is the location of the number or list value during each loop. The index name (indexName) is appended to the loop index marker (before the semi-colon) to refer to a specific set of loop values when mgiLoop tags are nested. For example, to refer to the index name "Colors", the loop index marker would be the following:

  &mgiLoopIndexColors;

Example Usage and Output

Counting Loop

```
<mgiLoop first="1" last="10" step="2">
&mgiLoopIndex; <BR>
</mgiLoop>
```

In this example, every other number (beginning with the first number) between 1 and 10 is displayed on a separate line as follows. If values are not separated by an HTML tag such as a header tag, paragraph tag or break tag then they are displayed continuously on one line.

```
1
3
5
7
9
```

Item Loop

```
<mgiLoop item="Dog" item="Cat" item="Rabbit"
indexName="Animals">
```

```
<P>Is your favorite animal a &mgiLoopIndexAnimals;?</P>
</mgiLoop>
```

In this example, each of three items is placed in the sentence "Is your favorite animal a...?" If values are not separated by an HTML tag such as a header tag, paragraph tag or break tag then they are displayed continuously on one line.

> Is your favorite animal a Dog?
>
> Is your favorite animal a Cat?
>
> Is your favorite animal a Rabbit?

List Loop

```
<mgiLoop itemList="Red,Orange,Yellow,Green,Blue"
listDelimiter="," indexName="Colors">
<P>&mgiLoopIndexColors; is a color in the rainbow.</P>
</mgiLoop>
```

In this example, each item in the set of colors is placed in the sentence "*Value* is a color in the rainbow." and displayed as follows. If values are not separated by an HTML tag such as a header tag, paragraph tag or break tag then they are displayed continuously on one line.

> Red is a color in the rainbow.
>
> Orange is a color in the rainbow.
>
> Yellow is a color in the rainbow.
>
> Green is a color in the rainbow.
>
> Blue is a color in the rainbow.

Nested mgiLoop Tags

```
<mgiLoop first="1" last="3" indexName="Numbers">
<P>&mgiLoopIndexNumbers;.
<UL>
  <mgiLoop itemList="A,B,C" listDelimiter=","
  indexName="Letters">
  <LI>&mgiLoopIndexLetters;
  </mgiLoop>
</UL>
</mgiLoop>
```

When you nest mgiLoop tags, you must use the "indexName" parameter and include the indexName parameter value in the loop index marker (otherwise the index loop marker may not display the values from the appropriate itemlist). In this example, an outline is created nesting a loop of the letters A, B and C in a bullet list inside a loop that counts from 1 to 3. The outline would appear as:

1

- A
- B
- C

2

- A
- B
- C

3

- A
- B
- C

Suggested Usage

- Repeating actions
- Form Processing

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiMath Tag

Tag Behavior

Use the mgiMath tag to perform numerical calculations.

Tag Syntax

The mgiMath tag has a beginning tag with no required parameters and three optional parameters, a body and an ending tag. The tag form is:

```
<mgiMath resultPrecision="Integer" workPrecision="Integer"
displayPrecision="Integer">
Calculation
</mgiMath>
```

Required Parameters:

- **None**

Optional Parameters:

- **resultPrecision -** The resultPrecision is the number of decimal places that the result is rounded to. The default is "8".
- **workPrecision -** The workPrecision is the number of decimal places maintained througout the calculation in order to avoid rounding errors at individual steps during the process. The default is "8".
- **displayPrecision -** The displayPrecision parameter specifies the precision that should be used to display the result. If the specified precision has less decimal places than the actual number, the number is rounded. If the specified precision has more decimal places than the actual number, the number is padded with trailing zeroes. The default is to display the number with no trailing zeroes after the decimal point.

Technical Notes

- **Mathematical operations -** The body of the mgiMath tag can contain mathematical expressions that adhere to the algebraic order of operations. The body can consist of the following characters as well as whitespace:

  ```
  0 1 2 3 4 5 6 7 8 9 . + - * / \ % ^ ( )
  ```

- **Operators**:
    - + Addition
    - - Subtraction or Negate
    - * Multiplication
    - / Division
    - \ Division (Remainder truncated)
    - % Modulo (Remainder of division)
    - ^ Exponentiation
    - ( ) Grouping
- **Order of Operations**:
    1. ( ) Grouping
    2. + - Positive and negative numbers
    3. ^ Exponents
    4. * / % Multiplication, division, and modulo (processed left to right)
    5. + - Addition and subtraction (processed left to right)

Example Usage and Output

```
<mgiMath resultPrecision="4" workPrecision="4">
1.5 + 6.7 * (4 - 12)
</mgiMath>
```

The result of this mathematical calculation is: 52.1

Suggested Usage

- Calculations

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiModifyDatabase Tag

Tag Behavior

Use the mgiModifyDatabase tag to modify a database without the web-based interface (see also mgiEditDatabase)

Tag Syntax

The mgiModifyDatabase tag has eleven modes. Each mode has different required and optional parameters. The modes of mgiModifyDatabase are:

- **listDatabases** - Displays the names of databases in the region.
- **createDatabase** - Creates a new database.
- **deleteDatabase** - Deletes an existing database.
- **listFields** - Displays the names of fields in a database.
- **addField** - Adds a field to a database.
- **deleteField** - Deletes a field from a database.
- **addRecord** - Adds a record to a database.
- **deleteRecord** - Deletes a record from a database.
- **updateRecord** - Modifies a database record.
- **export** - Exports records from a database in tab-delimited format.
- **import** - Imports tab-delimited data into a database.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

ListDatabases Mode

The listDatabases mode of the mgiModifyDatabase tag has one required parameters and four optional parameters. The tag form is:

```
<mgiModifyDatabase mode="listDatabases"
resultVariableName="Variable Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required parameters:

- **mode** - The mode is the function that the mgiModifyDatabase tag performs. In "**listDatabases**" mode, the mgiModifyDatabase tag displays a comma delimited list of database names in the region.

Optional Parameters:

- **resultVariableName** - The resultVariableName is the name prepended to result variables that contain information about the operation. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiModifyDatabase tag.
    - **resultvar_ResultCount** - A page variable containing the total number of databases in the list.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified and/or deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

CreateDatabase Mode

The createDatabase mode of the mgiModifyDatabase tag has two required parameters and four optional parameters. The tag form is:

```
<mgiModifyDatabase mode="createDatabase" databaseName="Name"
resultVariableName="Variable Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required parameters:

- **mode** - The mode is the function that the mgiModifyDatabase tag performs. In "**createDatabase**" mode, the mgiModifyDatabase tag creates a new database.
- **databaseName** - The databaseName is the name of the database to create.

Optional Parameters:

- **resultVariableName** - The resultVariableName is the name prepended to result variables that contain information about the operation. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiModifyDatabase tag.
  - **resultvar_DatabaseName** - A page variable that contains the name of the database in case it was modified by the server.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified and/or deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

DeleteDatabase Mode

The deleteDatabase mode of the mgiModifyDatabase tag has two required parameters and three optional parameters. The tag form is:

```
<mgiModifyDatabase mode="deleteDatabase" databaseName="Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required parameters:

- **mode** - The mode is the function that the mgiModifyDatabase tag performs. In "**deleteDatabase**" mode, the mgiModifyDatabase tag deletes the specified database
- **databaseName** - The databaseName is the name of the database to delete.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified and/or deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

ListFields Mode

The listFields mode of the mgiModifyDatabase tag has two required parameters and four optional parameters. The tag form is:

```
<mgiModifyDatabase mode="listFields" databaseName="Name"
resultVariableName="Variable Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required parameters:

- **mode** - The mode is the function that the mgiModifyDatabase tag performs. In "**listFields**" mode, the mgiModifyDatabase tag displays the names of fields in the specified database. Fields are listed in the format "name.type.length.flags,name.type.length.flags,name.type.length.flags". For example, `field1.Text.25.X,field2.UInteger.0.XQ,field3.LongText.0.X`
- **databaseName** - The databaseName is the name of the database to search.

Optional Parameters:

- **resultVariableName** - The resultVariableName is the name prepended to result variables that contain information about the operation. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiModifyDatabase tag.
    - **resultvar_ResultCount** - A page variable that contains the number of fields found.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the

server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified and/or deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

AddField Mode

The addField mode of the mgiModifyDatabase tag has four required parameters and six optional parameters. The tag form is:

```
<mgiModifyDatabase mode="addField" databaseName="Name"
fieldName="Field Name" fieldType="Type"
fieldLength="Integer" fieldFlags="X/Q"
resultVariableName="Variable Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required parameters:

- **mode** - The mode is the function that the mgiModifyDatabase tag performs. In "**addField**" mode, the mgiModifyDatabase tag creates a new database field in the specified database.
- **databaseName** - The databaseName is the name of the database the field is added to.
- **fieldName** - The fieldName is the name of the field to add.
  - The characters allowed in database field names for the built-in MGI database are alpha-numeric characters (i.e., letters and numbers), underscores (_), hyphens (-), and spaces. When adding a field, any non-allowed characters in the field name are automatically replaced with an underscore.
  - The length of field names in the internal MGI database is limited according to the type of field and operating system. All field names that are not indexed are limited to 63 characters. The limit for non-indexed field names applies to all operating systems. Indexed field names on Macintosh operating systems 9.04 and prior are limited to 25 characters. Indexed field names on all supported Windows operating systems are limited to 63 characters.
- **fieldType** - The fieldType is the type of field to add. Valid field types are Boolean,

Integer, UInteger, Number, Text, LongText. See the mgiEditDatabase tag for field type descriptions.

Optional Parameters:

- **fieldLength** - The fieldLength is the length of the field in characters if applicable to the field type.
- **fieldFlags** - The fieldFlags are designations for indexed and unique fields. Valid flags are "**X**" for indexed and "**Q**" for unique.
- **resultVariableName** - The resultVariableName is the name prepended to result variables that contain information about the operation. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiModifyDatabase tag.
    - **resultvar_FieldName** - A page variable that contains the name of the field in case it was modified by the server.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified and/or deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

DeleteField Mode

The deleteField mode of the mgiModifyDatabase tag has three required parameters and three optional parameters. The tag form is:

```
<mgiModifyDatabase mode="deleteField" databaseName="Name"
fieldName="Field Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required parameters:

- **mode** - The mode is the function that the mgiModifyDatabase tag performs. In "**deleteField**" mode, the mgiModifyDatabase tag deletes a field from the specified database
- **databaseName** - The databaseName is the name of the database the field is deleted from.
- **fieldName** - The fieldName is the name of the field to delete.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified and/or deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

AddRecord Mode

The addRecord mode of the mgiModifyDatabase tag has four of seven required parameters and three optional parameters. The tag form is:

```
<mgiModifyDatabase mode="addRecord" databaseName="Name"
fieldName="Field Name" fieldValue="Value"
fieldName="Field Name" pageVariableName="Variable Name"
fieldName="Field Name" pathArgumentName="Path Argument Name"
fieldName="Field Name" postArgumentName="Post Argument Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required parameters:

- **mode** - The mode is the function that the mgiModifyDatabase tag performs. In "**addRecord**" mode, the mgiModifyDatabase tag adds a record to the specified database.
- **databaseName** - The databaseName is the name of the database the record is added to.
- **fieldName** - The fieldName is the name of the database field to populate. Multiple

fieldName parameters may be included in one mgiModifyDatabase tag, but each fieldName parameter must be **immediately followed** by a fieldValue, pageVariableName, pathArgumentName or postArgumentName parameter that contains the value to populate the field.

- **fieldValue** - The fieldValue is the value used to populate the field specified by the preceding fieldName parameter in the mgiModifyDatabase tag.
- **pageVariableName** - The pageVariableName is the name of the page variable that contains the value used to populate the field specified by the preceding fieldName parameter in the mgiModifyDatabase tag.
- **pathArgumentName** - The pathArgumentName is the name of the path argument that contains the valueused to populate the field specified by the preceding fieldName parameter in the mgiModifyDatabase tag.
- **postArgumentName** - The postArgumentName is the name of the post argument that contains the value used to populate the field specified by the preceding fieldName parameter in the mgiModifyDatabase tag.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified and/or deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

DeleteRecord Mode

The deleteRecord mode of the mgiModifyDatabase tag has two required parameters and eleven optional parameters. The tag form is:

```
<mgiModifyDatabase mode="deleteRecord" databaseName="Name"
keyfieldName="Field Name" fieldValue="Value"
keyfieldName="Field Name" pageVariableName="Variable Name"
keyfieldName="Field Name" pathArgumentName="Path Argument Name"
keyfieldName="Field Name" postArgumentName="Post Argument Name"
```

```
resultVariableName="Variable Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password"
uniqueIDFieldName="Field">
```

or

```
<mgiModifyDatabase mode="deleteRecord" databaseName="Name"
searchString="String" resultVariableName="Variable Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password" uniqueIDFieldName="Field">
```

Required parameters:

- **mode -** The mode is the function that the mgiModifyDatabase tag performs. In "**deleteRecord**" mode, the mgiModifyDatabase tag deletes records from the specified database.
- **databaseName -** The databaseName is the name of the database to search.

Optional Parameters:

- **keyFieldName -** The keyFieldName is the name of the database field to search for the record(s) to delete. Multiple keyFieldName parameters may be included in one mgiModifyDatabase tag, but each keyFieldName parameter must be **immediately followed** by a fieldValue, pageVariableName, pathArgumentName or postArgumentName parameter that contains the search criteria for that field. When multiple keyFieldName and value pairs are present, an AND search is performed to locate the record(s) to delete. To perform an OR search, you must use the searchString parameter. All records that match the key field search criteria are deleted.
- **fieldValue -** The fieldValue is the search criteria for the preceding keyFieldName parameter in the mgiModifyDatabase tag.
- **pageVariableName -** The pageVariableName is the name of the page variable that contains the search criteria for the preceding keyFieldName parameter in the mgiModifyDatabase tag.
- **pathArgumentName -** The pathArgumentName is the name of the path argument that contains the search criteria for the preceding keyFieldName parameter in the mgiModifyDatabase tag.
- **postArgumentName -** The postArgumentName is the name of the post argument that contains the search criteria for the preceding keyFieldName parameter in the mgiModifyDatabase tag.
- **searchString -** The search string is a fully formatted search string containing the field names to search, the search criteria, and the search operations. If the searchString parameter is specified, the keyFieldName, pageVariableName, pathArgumentName, postArgumentName, orderByField, and reverseOrder parameters are ignored. Search strings should contain field name and search criteria in the format

keyFieldName='criteria'. Valid operators are "=" (equals), ">" (greater than), "<" (less than), ">=" (greater than and equal to), and "<=" (less than and equal to). Field name and search criteria pairs should be separated by the search operator "AND", "OR", or "NOT". To order the results from a search string, include the order by field in the format **ORDER BY keyFieldName**. By default, search results are in asending order (ASC). To reverse the order of results in a search string, specify descending order by including DESC after the ORDER BY field. You may not enter quotes ("") in the value of the search string parameter. The following are example search strings:

```
searchString="LastName='J*' OR LastName='L'
AND Registration='Yes' ORDER BY CustID DESC"

searchString="FirstName='Ken' AND NOT LastName='Barker'
```

- **resultVariableName -** The resultVariableName is the name prepended to result variables that contain information about the operation. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiModifyDatabase tag.
    - **resultvar_ResultCount -** A page variable that contains the number of matching records that were deleted.
- **uniqueIDFieldName -** The uniqueIDFieldName parameter is the name of the ODBC database field to use for a unique record value.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified and/or deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

UpdateRecord Mode

The updateRecord mode of the mgiModifyDatabase tag has two required parameters and ten optional parameters. The tag form is:

```
<mgiModifyDatabase mode="updateRecord" databaseName="Name"
keyfieldName="Field Name" fieldValue="Value"
keyfieldName="Field Name" pageVariableName="Variable Name"
keyfieldName="Field Name" pathArgumentName="Path Argument Name"
keyfieldName="Field Name" postArgumentName="Post Argument Name"
fieldName="Field Name" fieldValue="Value"
fieldName="Field Name" pageVariableName="Variable Name"
fieldName="Field Name" pathArgumentName="Path Argument Name"
fieldName="Field Name" postArgumentName="Post Argument Name"
resultVariableName="Variable Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

or

```
<mgiModifyDatabase mode="updateRecord" databaseName="Name"
searchString="String" resultVariableName="Variable Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required parameters:

- **mode -** The mode is the function that the mgiModifyDatabase tag performs. In "**updateRecord**" mode, the mgiModifyDatabase tag modifies existing records in the specified database.
- **databaseName -** The databaseName is the name of the database to search.

Optional Parameters:

- **keyFieldName -** The keyFieldName is the name of the database field to search for the record(s) to modify. Multiple keyFieldName parameters may be included in one mgiModifyDatabase tag, but each keyFieldName parameter must be **immediately followed** by a fieldValue, pageVariableName, pathArgumentName or postArgumentName parameter that contains the search criteria for that field. When multiple keyFieldName and value pairs are present, an AND search is performed to locate the record(s) to modify. To perform an OR search, you must use the searchString parameter. All records that match the key field search criteria are modified.
- **fieldName -** The fieldName is the name of the database field to populate. Multiple fieldName parameters may be included in one mgiModifyDatabase tag, but each fieldName parameter must be **immediately followed** by a fieldValue, pageVariableName, pathArgumentName or postArgumentName parameter that contains the value to populate the field.
- **fieldValue -** The fieldValue is the search criteria for the preceding keyFieldName or update criteria for the preceding fieldName parameter in the mgiModifyDatabase tag.
- **pageVariableName -** The pageVariableName is the name of the page variable that

contains the search criteria for the preceding keyFieldName or update criteria for the preceding fieldName parameter in the mgiModifyDatabase tag.

- **pathArgumentName -** The pathArgumentName is the name of the path argument that contains the search criteria for the preceding keyFieldName or update criteria for the preceding fieldName parameter in the mgiModifyDatabase tag.
- **postArgumentName -** The postArgumentName is the name of the post argument that contains the search criteria for the preceding keyFieldName or update criteria for the preceding fieldName parameter in the mgiModifyDatabase tag.
- **searchString -** The search string is a fully formatted search string containing the field names to search, the search criteria, and the search operations. If the searchString parameter is specified, the keyFieldName, pageVariableName, pathArgumentName, postArgumentName, orderByField, and reverseOrder parameters are ignored. Search strings should contain field name and search criteria in the format **keyFieldName='criteria'**. Valid operators are "=" (equals), ">" (greater than), "<" (less than), ">=" (greater than and equal to), and "<=" (less than and equal to). Field name and search criteria pairs should be separated by the search operator "AND", "OR", or "NOT". To order the results from a search string, include the order by field in the format **ORDER BY keyFieldName**. By default, search results are in asending order (ASC). To reverse the order of results in a search string, specify descending order by including DESC after the ORDER BY field. You may not enter quotes ("") in the value of the search string parameter. The following are example search strings:

```
searchString="LastName='J*' OR LastName='L'
AND Registration='Yes' ORDER BY CustID DESC"

searchString="FirstName='Ken' AND NOT LastName='Barker'
```

- **resultVariableName -** The resultVariableName is the name prepended to result variables that contain information about the operation. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiModifyDatabase tag.
    - **resultvar_ResultCount -** A page variable that contains the number of matching records that were modified.
- **odbcDatasource** (NT only) **-** The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified and/or deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) **-** The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the

odbcDatasource parameter.

- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

Export Mode

The export mode of the mgiModifyDatabase tag has three required parameters and ten optional parameters. The tag form is:

```
<mgiModifyDatabase mode="export" databaseName="Name"
fileLocation="File Path"
keyfieldName="Field Name" fieldValue="Value"
keyfieldName="Field Name" pageVariableName="Variable Name"
keyfieldName="Field Name" pathArgumentName="Path Argument Name"
keyfieldName="Field Name" postArgumentName="Post Argument Name"
resultVariableName="Variable Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

or

```
<mgiModifyDatabase mode="export" databaseName="Name"
searchString="String" fileLocation="File Path"
resultVariableName="Variable Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required parameters:

- **mode** - The mode is the function that the mgiModifyDatabase tag performs. In "**export**" mode, the mgiModifyDatabase tag modifies exports records found in a search of the specified database.
- **databaseName** - The databaseName is the name of the database to search.
- **fileLocation** - The fileLocation is the relative path to the file where exported files are written. If the file exists, it's content will be overwritten. If the file does not exist, it will be created.

Optional Parameters:

- **keyFieldName** - The keyFieldName is the name of the database field to search for the record(s) to export. Multiple keyFieldName parameters may be included in one mgiModifyDatabase tag, but each keyFieldName parameter must be **immediately followed** by a fieldValue, pageVariableName, pathArgumentName or postArgumentName parameter that contains the search criteria for that field. When multiple keyFieldName and value pairs are present, an AND search is performed to locate the record(s) to export. To perform an OR search, you must use the searchString parameter. All records that match the key field search criteria are modified.

- **fieldValue -** The fieldValue is the search criteria for the preceding keyFieldName parameter in the mgiModifyDatabase tag.
- **pageVariableName -** The pageVariableName is the name of the page variable that contains the search criteria for the preceding keyFieldName parameter in the mgiModifyDatabase tag.
- **pathArgumentName -** The pathArgumentName is the name of the path argument that contains the search criteria for the preceding keyFieldName parameter in the mgiModifyDatabase tag.
- **postArgumentName -** The postArgumentName is the name of the post argument that contains the search criteria for the preceding keyFieldName parameter in the mgiModifyDatabase tag.
- **searchString -** The search string is a fully formatted search string containing the field names to search, the search criteria, and the search operations. If the searchString parameter is specified, the keyFieldName, pageVariableName, pathArgumentName, postArgumentName, orderByField, and reverseOrder parameters are ignored. Search strings should contain field name and search criteria in the format **keyFieldName='criteria'**. Valid operators are "=" (equals), ">" (greater than), "<" (less than), ">=" (greater than and equal to), and "<=" (less than and equal to). Field name and search criteria pairs should be separated by the search operator "AND", "OR", or "NOT". To order the results from a search string, include the order by field in the format **ORDER BY keyFieldName**. By default, search results are in asending order (ASC). To reverse the order of results in a search string, specify descending order by including DESC after the ORDER BY field. You may not enter quotes ("") in the value of the search string parameter. The following are example search strings:

```
searchString="LastName='J*' OR LastName='L'
AND Registration='Yes' ORDER BY CustID DESC"

searchString="FirstName='Ken' AND NOT LastName='Barker'
```

- **resultVariableName -** The resultVariableName is the name prepended to result variables that contain information about the operation. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiModifyDatabase tag.
  - **resultvar_ResultCount -** A page variable that contains the number of matching records that were exported.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified

and/or deleted from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

Import Mode

The import mode of the mgiModifyDatabase tag has three required parameters and four optional parameters. The tag form is:

```
<mgiModifyDatabase mode="import" databaseName="Name"
fileLocation="File Path" resultVariableName="Variable Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required parameters:

- **mode** - The mode is the function that the mgiModifyDatabase tag performs. In "**import**" mode, the mgiModifyDatabase tag modifies imports record to the specified database.
- **databaseName** - The databaseName is the name of the database to use.
- **fileLocation** - The fileLocation is the relative path to the file to import.

Optional Parameters:

- **resultVariableName** - The resultVariableName is the name prepended to result variables that contain information about the operation. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiModifyDatabase tag.
    - **resultvar_ResultCount** - A page variable that contains the number of matching records that were imported.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be stored, modified and/or deleted from the specified ODBC database rather than the internal MGI database.

Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Return to the mgiModifyDatabase Mode Menu | Examples and Output | Suggested Usage

Example Usage and Output

```
<mgiModifyDatabase mode="listDatabases">
```

In this example, all databases in a region (including internal MGI databases) are listed in a comma-delimited format such as:

```
        Books,_MGIDB_Authentication_,_MGIDB_Banner_Ads_
```

```
<mgiModifyDatabase mode="addRecord" databaseName="Contacts"
fieldName="Name" postArgumentName="fullName"
fieldName="Address" postArgumentName="streetAddress"
fieldName="City" postArgumentName="city"
fieldName="State" postArgumentName="state"
fieldName="Zip" postArgumentName="zipCode"
fieldName="Phone" postArgumentName="phoneNumber"
fieldName="Email" postArgumentName="emailAddress">
```

In this example, a contact is added to the "Contacts" database from a form submission.

Suggested Usage

- Form-Based Database Modifications

The mgiModifyFile Tag

Tag Behavior

Use the mgiModifyFile tag to overwrite or append to an existing file.

Tag Syntax

The mgiModifyFile tag has two required parameters and one optional parameter. The tag form is:

```
<mgiModifyFile fileLocation="File Path" mode="Mode"
appendCRLF="Yes/No">
MGI Tags, HTML or Text
</mgiModifyFile>
```

Required Parameters:

- **fileLocation -** The fileLocation is the relative path to the file to modify. If the file does not exist, it will be created.
- **mode -** The mode is the function that the mgiModifyFile tag performs. In "**write**" mode, the mgiModifyFile tag overwrites the contents of the file listed in the fileLocation parameter with the information in the body of the mgiModifyFile tag. In "**append**" mode, the mgiModifyFile tag appends the information in the body of the mgiModifyFile tag to the end of the file listed in the fileLocation parameter.

Optional Parameters:

- **appendCRLF -** The appendCRLF parameter determines whether a carriage return and line feed are appended after the information in the body of the mgiModifyFile tag is written or appended. If the appendCRLF parameter value is "**Yes**", then a carriage return (Mac) or a carriage return and line feed (NT) are appended. If the appendCRLF parameter value is "**No**", then no characters (visible or hidden) are appended. The default value is "Yes"

Example Usage and Output

```
<mgiModifyFile mode="append"
fileLocation="contestants.txt" appendCRLF="Yes">
<mgiPostArgument name="Name"> <mgiPostArgument name="Email">
```

```
<mgiPostArgument name="Phone">
</mgiModifyFile>
```

In this example, visitors register for a contest and their information is appended to a file named contestants.txt

---

Suggested Usage

- Contests
- Web-Based Dynamic Content Updates

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiModifyFileSystem Tag

Tag Behavior

Use the mgiModifyFileSystem tag to create or delete files and folders.

Tag Syntax

The mgiModifyFileSystem tag has two required parameters and one optional parameter. The tag form is:

```
<mgiModifyFileSystem mode="Mode" itemLocation="Path"
targetLocation="Path">
```

Required Parameters:

- **mode -** The mode is the function that the mgiModifyFileSystem tag performs. In "**createFile**" mode, the file listed in the itemLocation parameter is created. In "**createFolder**" mode, the folder listed in the itemLocation parameter is created. In "**delete**" mode, the file or folder listed in the itemLocation parameter is deleted. In "**move**" mode, the file or folder listed in the itemLocation parameter is moved to the location listed in the targetLocation parameter. In "**rename**" mode, the file or folder listed in the itemLocation parameter is renamed to the name listed in the targetLocation parameter. In the "**copy**" mode, the file or folder listed in the itemLocation parameter is copied to the location listed in the targetLocation parameter.
- **itemLocation -** The itemLocation is the relative path to the file or folder.

Optional Parameters:

- **targetLocation -** The targetLocation is the relative path to the moved or renamed file or folder. The targetLocation parameter is required for the "move", "rename" and "copy" modes.

Example Usage and Output

```
<mgiModifyFileSystem itemLocation="content1.mgi"
mode="createFile">
```

In this example, a page named "content1.mgi" is created at the same level of the file system as the page containing the mgiModifyFileSystem tag.

```
<mgiModifyFileSystem itemLocation="Articles"
mode="createFolder">
```

In this example, a folder named "Articles" is created at the same level of the file system as the page containing the mgiModifyFileSystem tag.

```
<mgiModifyFileSystem itemLocation="main.htm"
targetLocation="Archive/02212001.htm" mode="move">
```

In this example, a file named "main.htm" is moved to the "Archive" folder and renamed to "02212001.htm".

Suggested Usage

- Creating New Files and Folders
- Managing files via a Web-Based Interface
- Moving and Renaming Files

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiPathArgument Tag

Tag Behavior

Use the mgiPathArgument tag to display the value of a path argument from a URL. Path arguments are created when you submit a form using the GET method.

Tag Syntax

The mgiPathArgument tag has no required parameters and four optional parameters. The tag form is:

```
<mgiPathArgument name="Name" defaultValue="Text"
mode="Mode" delimiter="Character">
```

Required Parameters:

- **None.**

Optional Parameters:

- **name** - The name is the name of the path argument to display.
- **defaultValue** - The defaultValue is the text that displays if the path argument is not found or if the path argument is blank. If a default value is not specified, an empty string (blank) is displayed.
- **mode** - The mode is the function that the mgiPathArgument performs. In "**returnAllNames**" mode, the names of all path arguments are displayed. In "**returnAllValues**" mode, the values of all path arguments are displayed. In "**returnAll**" mode, the names and values of all path arguments are displayed in the format below. If you include the mode parameter, the name parameter is not required.

  Name1: Value1

  Name2: Value2

  Name3: Value3

  NameX: ValueX
- **delimiter** - The delimiter is the character (e.g., "_", tabs, etc.) that is used to separate the names and values in returnAllNames and returnAllValues modes. The delimiter is also used to separate multiple values for the same path argument name. The default delimiter is a comma. In order to use special reserved characters such as tabs and backslashes as delimiters, you must use the appropriate Escaped String format.

Example Usage and Output

```
<mgiPathArgument name="AffilateID">
```

In this example, an affiliate of the company links customers to the site using an Affiliate ID path argument to identify the customer's origin. The affiliate's ID is displayed for use in a shopping basket order, for example.

```
<mgiPathArgument mode="returnAllValues" delimiter="\t">
```

In this example, all path argument values are returned in a tab-delimited string to import into a database.

Suggested Usage

- Form Processing
- Affiliate Systems
- Shopping Baskets

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiPGP Tag

Tag Behavior

Use the mgiPGP tag to encrypt the information in the body of the tag with a passphrase. The PGP function requires a utility on the user's local computer to decrypt the text. Free PGP utilities for personal use can be found at http://web.mit.edu/network/pgp.html. Commercial PGP utilities can be found at http://www.pgp.com.

Tag Syntax

The mgiPGP tag has two forms. Each form has a beginning tag with one required parameter and two optional parameters, a body, and an ending tag. The tag forms are:

```
<mgiPGP passPhrase="Phrase" mode="Mode"
suppressErrorMessage="Yes/No">
Text to encrypt
</mgiPGP>
```

or

```
<mgiPGP fileLocation="File Path" mode="Mode"
suppressErrorMessage="Yes/No">
Text to encrypt
</mgiPGP>
```

Required Parameters:

- **passPhrase** - The passphrase is the phrase used to encrypt the information in the body of the mgiPGP tag. If you include the passPhrase parameter, the fileLocation parameter is not required.
- **fileLocation** - The fileLocation is the relative path to the file that contains the phrase used to encrypt the information in the body of the mgiPGP tag. If you include the fileLocation parameter, the passPhrase parameter is not required.

Optional Parameters:

- **mode** - The mode is the function that the mgiPGP tag performs. In "**encryptWithPhrase**" mode, the mgiPGP tag encrypts the information in the body of the tag with the specified phrase. In "**decryptWithPhrase**" mode, the mgiPGP tag decrypts the information in the body of the tag with the specified phrase. The default is "encryptWithPhrase".

- **suppressErrorMessages -** The suppressErrorMessages parameter determines if error messages associated with the mgiPGP tag are displayed or not displayed. If the suppressErrorMessages parameter value is "**Yes**", then error messages associated with the mgiPGP tag are not displayed. If the suppressErrorMessages parameter value is "**No**", then error messages associated with the mgiPGP tag are displayed. The default value is "No".

Example Usage and Output

```
<mgiSendMail to="accounting@domain.com"
from="webmaster@domain.com"
subject="Payment" mailserver="mail@domain.com">

<mgiPGP passPhrase="Security Is Necessary 5589456">

<mgiPostArgument name="Account">
<mgiPostArgument name="CreditCardType">
<mgiPostArgument name="CreditCardNumber">
<mgiPostArgument name="ExpireMonth">
<mgiPostArgument name="ExpireYear">

</mgiPGP>
</mgiSendMail>
```

In this example, the mgiPGP tag is used to encrypt account payments that include credit card numbers. When an account payment is made, an encrypted email that contains the account and credit card information is sent to the accounting department. The accounting department receives the email and decrypts it using a PGP utility.

Suggested Usage

- Secure Order Processing
- Secure Email Transmissions

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiPoll Tag

Tag Behavior

Use the mgiPoll tag to collect and display the results of online surveys.

Tag Syntax

The mgiPoll tag has three modes. Each mode has different required and optional parameters. The modes of mgiPoll are:

- **collect** - Collects poll data.
- **admin** - Creates a web-based interface to manage polls.
- **tally** - Displays poll results.

Collect Mode

The collect mode of mgiPoll has one required parameter and five optional parameters. The tag form is:

```
<mgiPoll mode="collect" name="Name" username="Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **name -** The name is the name of the poll that corresponds with the data being collected.

Optional Parameters:

- **mode -** The mode is the function that the mgiPoll tag performs. In "**collect**" mode, the mgiPoll tag collects poll results and stores them in the poll database. "Collect" is the default mode of the mgiPoll tag.
- **username -** The username is the username of the administrator that created the poll. Polls created by different administrators may have the same name and this username parameter differentiates which poll data to collect. The default username is "Admin".
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, poll information will be stored in the specified

ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Admin Mode

The admin mode of mgiPoll has one required parameter and four optional parameters. The tag form is:

```
<mgiPoll mode="admin" username="Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiPoll tag performs. In "**admin**" mode, the mgiPoll tag creates a web-based interface that allows you to add and manage poll data.

Optional Parameters:

- **username** - The username identifies the poll administrator. Administrators will only have access to the polls created with their username. Polls created with different usernames (i.e, by different administrators) may have the same name. The default username is "Admin".
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, poll information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Tally Mode

The tally mode of mgiPoll has two required parameter and eight optional parameters. The tag form is:

```
<mgiPoll mode="tally" name="Name" username="Name"
color="Color" postArgumentName="Post Argument"
responseName="Post Argument Value"
responseNameCellWidth="Whole Number"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiPoll tag performs. In "**tally**" mode, the mgiPoll tag calculates the current poll data and displays results in a bar graph.
- **name** - The name is the name of the poll to tally.

Optional Parameters:

- **username** - The username is the username of the administrator that created the poll. Polls created by different administrators may have the same name and this username parameter differentiates which poll data to tally. The default username is "Admin".
- **postArgumentName** - The postArgumentName is the name of the post argument to tally and display. The default behavior is to tally all post arguments in the poll.
- **responseName** - The responseName is the value of the post argument to tally and display. The default behavior is to tally and display all values of a post argument. The postArgumentName parameter is required if you include the responseName parameter.
- **responseNameCellWidth** - The width of the cell in which the response name is displayed in the tally. The default value is 125.
- **color** - The color is the color name or hex code (including pound sign - #) for the bars in the poll graph. If the color parameter is not included, the first color in the poll admin color cycle will be used.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, poll information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.

- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Technical Notes

- **Post Argument Name Limit -** Post Argument names for poll questions may not exceed 35 characters. The post argument name is not shown to visitors and can be abbreviated to meet the 35 character limit if necessary (e.g., "Q1", "Q2", etc.). The post argument value is not limited and will be displayed to the visitor as a label for poll results.

Example Usage and Output

Collect Mode

```
<mgiPoll name="President">
```

In this example, the data from the "President" poll is collected from each poll submission and stored in the poll database.

Admin Mode

```
<mgiPoll mode="admin">
```

When you access a page with the mgiPoll tag in admin mode, the functions for creating polls and setting poll preferences are displayed.

Tally Mode

```
<mgiPoll mode="tally" name="VicePres">
```

In this example, the results from the "VicePres" poll is tallied and displayed in a bar graph.

```
<mgiPoll mode="tally" name="VicePres"
postArgumentName="Republican">
```

In this example, the results from the "Republican" question of the "VicePres" poll is tallied and displayed in a bar graph.

```
<mgiPoll mode="tally" name="VicePres"
postArgumentName="Republican" responseName="Dick Cheney">
```

In this example, the results for the "Dick Cheney" value of the "Republican" question in the

"VicePres" poll is tallied and displayed in a bar graph.

Suggested Usage

- Polls
- Surveys

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiPOP Tag

Tag Behavior

Use the mgiPOP tag to download emails from a POP email account and enter each email as a record in an MGI database. Depending on the POP account settings, emails may be left on the server or deleted from the server as they are downloaded. **Attachments are stripped from the email and are not saved**.

Tag Syntax

The mgiPOP tag has four required parameters and one optional parameter. The tag form is:

```
<mgiPOP username="Username" password="Password"
mailServer="POP Server" databaseName="Database"
deleteMessagesOnServer="Yes/No">
```

Required Parameters:

- **username** - The username is the login for the POP account.
- **password** - The password is the password for the POP account.
- **mailServer** - The mailServer is the incoming POP server that is used to retrieve the email.
- **databaseName** - The name of the MGI database where email records are added. If the database does not exist, it will be created. Emails are parsed into the following fields:
  - Header (long text)
  - Message (long text)
  - Read (boolean)
  - Unique ID (text, 50, indexed, unique)
  - To (long text)
  - From (long text)
  - From Name (long text)
  - From Email (long text)
  - Date (long text)
  - Subject (long text)
  - Size (long text)
  - Return-Path (long text)
  - Received (long text)

- X-Sender (long text)
- Message-ID (long text)
- Mime-Version (long text)
- Content-Type (long text)
- Status (long text)
- AttachmentFileNames (long text) (*Not used - for future enhancement only*).

Optional Parameters:

- **deleteMessagesOnServer** - The deleteMessagesOnServer parameter determines whether messages downloaded by the mgiPOP tag are deleted from the server. If the deleteMessagesOnServer parameter value is "**Yes**", then messages are deleted from the server. If the deleteMessagesOnServer parameter value is "**No**", then message are not deleted from the server. Even if messages are not deleted from the server, duplicate messages are not downloaded. Check with your email administrator for specifics about your POP account. Some email servers do not allow email to be left on the server regardless of the email client settings, therefore your original email messages could be deleted even if the value of this parameter is "No".

Example Usage and Output

```
<mgiPOP username="archive5" password="HF4577dg"
mailServer="pop.domain.com" databaseName="TalkList">
```

In this example, emails from a talk list are retrieved from the "archive5" pop account and saved as records in the "TalkList" database.

Suggested Usage

- Searchable Email Archives

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiPostArgument Tag

Tag Behavior

Use the mgiPostArgument tag to display the value of a post argument. Post arguments are created when you submit a form using the POST method.

Tag Syntax

The mgiPostArgument tag has no required parameters and four optional parameters. The tag form is:

```
<mgiPostArgument name="Name" defaultValue="Text"
mode="Mode" delimiter="Character">
```

Required Parameters:

- **None.**

Optional Parameters:

- **name** - The name is the name of the post argument to display.
- **defaultValue** - The defaultValue is the text that displays if the post argument is not found or if the post argument is blank. If a default value is not specified, an empty string (blank) is displayed.
- **mode** - The mode is the function that the mgiPostArgument performs. In "**returnAllNames**" mode, the names of all post arguments are displayed. In "**returnAllValues**" mode, the values of all post arguments are displayed. In "**returnAll**" mode, the names and values of all post arguments are displayed in the format below. If you include the mode parameter, the name parameter is not required.

  Name1: Value1

  Name2: Value2

  Name3: Value3

  NameX: ValueX
- **delimiter** - The delimiter is the character (e.g., "_", tabs, etc.) that is used to separate the names and values in returnAllNames and returnAllValues modes. The delimiter is also used to separate multiple values for the same post argument name. The default delimiter is a comma. In order to use special reserved characters such as tabs and backslashes as delimiters, you must use the appropriate Escaped String format.

Example Usage and Output

```
<mgiPostArgument name="Email">
```

In this example, a field named "Email" is included on a form. When a web site visitor completes the form, they enter "info@domain.edu" in the "Email" field and submit the form. The following is displayed in the location of the mgiPostArgument tag:

info@domain.edu

Suggested Usage

- Form Processing

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiQuiz Tag

Tag Behavior

Use the mgiQuiz tag to create, grade and manage online quizzes.

Tag Syntax

The mgiQuiz tag has two modes. Each mode has different required and optional parameters. The modes of mgiQuiz are:

- **grade -** Grades quiz answers.
- **admin -** Creates a web-based interface to create and manage quizzes.

Grade Mode

The grade mode of mgiQuiz has one required parameter and six optional parameters. The tag form is:

```
<mgiQuiz mode="grade" name="Quiz" username="Name"
type="Quiz Type" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **name -** The name is the name of the quiz to grade. Quizzes are named when they are created via the quiz admin interface. All quiz names are **case-sensitive**.

Optional Parameters:

- **mode -** The mode is the function that the mgiQuiz tag performs. In "**grade**" mode, the mgiQuiz tag grades quiz answers, records quiz grades in the database (for "Composite" quizzes only) and displays the quiz grade. "Grade" is the default mode of the mgiQuiz tag.
- **username -** The username is the username of the administrator that created the quiz. Quizzes created by different administrators may have the same name and this username parameter differentiates which quiz to grade. The default username is "Admin". All usernames are **case-sensitive**.
- **type -** The type is the type of quiz you are grading. If the type parameter value is "**composite**", then the quiz you are grading was created from quiz questions in the quiz database. If the type parameter value is "**custom**", then the quiz questions were

submitted from a custom quiz form and do not necessarily appear in the quiz database. The default value is "composite".

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, quiz information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Admin Mode

The admin mode of mgiQuiz has one required parameter and five optional parameters. The tag form is:

```
<mgiQuiz mode="admin" username="Name"
displayLimit="Integer" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **mode** - The mode is the function that the mgiQuiz tag performs. In "**admin**" mode, the mgiQuiz tag creates a web-based interface that allows you to add new quiz questions, "print" quiz forms, and manage quiz grades from composite quizzes.

Optional Parameters:

- **username** - The username identifies the quiz administrator. Administrators will only have access to the quizzes created with their username. Quizzes created with different usernames (i.e, by different administrators) may have the same name. The default username is "Admin". All usernames are **case-sensitive**.
- **displayLimit** - The displayLimite is the number of quiz questions to display per page in the quiz admin. The default value is "10". The maximum value is "50".
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, quiz information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you

include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Example Usage and Output

Grade Mode

```
<mgiQuiz name="Algebra1A" username="MathDept">
```

In this example, the "Algeabra1A" quiz questions will be graded and the grades and associated information will be stored in the quiz database.

Admin Mode

```
<mgiQuiz mode="admin" username="ScienceDept"
displayLimit="25">
```

When you access a page with the mgiQuiz tag in admin mode, the quiz admin interface displays allowing you to manage quiz questions and grades. In this example, only those quizzes created by the "Science" administrator will appear and questions will be displayed 25 per page.

Suggested Usage

- Online Quizzes

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiRandomNumber Tag

Tag Behavior

Use the mgiRandomNumber tag to display a random number within a specified range.

Tag Syntax

The mgiRandomNumber tag has no required parameters and two optional parameters. The tag form is:

```
<mgiRandomNumber lowerBound="Integer" upperBound="Integer">
```

Required Parameters:

- **None.**

Optional Parameters:

- **lowerBound -** The lowerBound is the the smallest whole number in the range from which random numbers are generated.
- **upperBound -** The upperBound is the the largest whole number in the range from which random numbers are generated.

Example Usage and Output

```
<mgiRandomNumber>
```

In this example, a random number between 1 and 4,294,967,295 will be displayed.

```
<mgiRandomNumber lowerbound="1" upperbound="1000">
```

In this example, a random number between 1 and 1000 will be displayed.

```
<mgiRandomNumber lowerbound="10000" upperbound="99999">
```

In this example, a random number between 10000 and 99999 will be displayed.

Suggested Usage

- Password Protection

- Unique Database Data
- Browser Caching Problems

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiRedirect Tag

Tag Behavior

Use the mgiRedirect tag to redirect an incoming HTTP request to a specified page. Note*:* MGI tags are parsed in order from the top of a page to the bottom of a page, so when an mgiRedirect tag is encountered a visitor is immediately redirected to the specified page and any MGI tags located below (after) the mgiRedirect tag will not be parsed (although MGI tags encountered before the mgiRedirect tag will be parsed completely).

Tag Syntax

The mgiRedirect tag has one required parameter. The tag form is:

```
<mgiRedirect url="URL">
```

Required Parameters:

- **url** - The url is the URL address where the browser is redirected.

Optional Parameters:

- **None.**

Example Usage and Output

```
<mgiRedirect url={mgiPostArgument name="Navigation"}>
```

In this example, the mgiRedirect tag is used in conjunction with a pop-up list and form to redirect visitors to a specific page of a web site. When a visitor selects and submits an option from the pop-up list, the mgiRedirect tag redirects to the URL that was selected.

Suggested Usage

- Pop-Up Menu Navigation Links.
- Outdated Links
- Conditional Comparisons

The mgiRequest Tag

Tag Behavior

Use the mgiRequest tag to display HTTP header information in the location of the tag.

Tag Syntax

The mgiRequest tag has one required parameter and no optional parameters. The tag form is:

```
<mgiRequest data="Type">
```

Required Parameters:

- **data** - The data is the type of HTTP header information to display. Valid data types are:
    - **Username** - The username if authentication was required.
    - **Password** - The password if authentication was required.
    - **ClientAddress** - The domain name of the client (or the client's IP address if DNS look-ups are disabled).
    - **ServerAddress** - The domain name of the server (or the server's IP address if DNS look-ups are disabled).
    - **ServerPort** - The TCP/IP port the server is listening on.
    - **ScriptName** - The file portion of the URL.
    - **ContentType** - The MIME type of post arguments, if present.
    - **Referrer** - The URL of the page visited prior to accessing the page which contains mgiRequest.
    - **UserAgent** - The browser software name and version.
    - **Method** - The HTTP method being requested (such as GET, GET_CONDITIONAL, POST, etc.).
    - **ClientIP** - The IP address of the client.
    - **VirtualHost** - The name of the web site that is being accessed.
    - **PageID** - The number of the page served since the last server restart for each region.
    - **QueryString** - The string of path arguments in the URL.

Optional Parameters:

- **None.**

Example Usage and Output

```
Welcome Back <mgiRequest data="username">!
```

In this example, the mgiRequest tag is used to display the username entered by a visitor into an authentication dialogue box. If the username was Pete, then the following would be displayed:

Welcome Back Pete!

Suggested Usage

- User Authentication

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiRollOver Tag

Tag Behavior

Use the mgiRollOver tag to display Javascript roll over (mouse over) images.

Tag Syntax

The mgiRollOver tag has two modes. Each mode has different required and optional parameters. The modes of mgiRollOver are:

- **head** - Creates the Javascript required for the roll over function.
- **image** - Specifies which roll over image to display.

Head Mode

The head mode of the mgiRollOver tag has three required parameters and two optional parameters. The tag form is:

```
<mgiRollOver mode="head" rollOverNames="Name List"
rollOverImages="File Names" baseURL="File Location"
delimiter="Character">
```

Required Parameters:

- **mode** - The mode is the function of the mgiRollOver tag. In "**head**" mode, the mgiRollOver tag creates the Javascript required for the roll over function.
- **rollOverNames** - The rollOverNames parameter is the delimited list of names that refer to each set of roll over images. The order of the names in the rollOverNames parameter must match the order of images listed in the rollOverImages parameter. The number of names listed in the rollOverNames parameter must also match the number of images listed in the rollOverImages parameter. The rollOverNames should be delimited by a comma (default) or by the character listed in the delimiter parameter.
- **rollOverImages** - The rollOverImages parameter is the delimited list of image file names. The list should contain only file names and should not contain full file paths (use the optional baseURL parameter to specifiy a file path). These images are the roll over images (i.e., the images that display after you mouse over each image listed in the "image" mode of the mgiRollOver tag). All roll over images should reside in the same folder. The order of the image file names in the rollOverImages parameter must match the order of the names in the rollOverNames parameter. The number of image files listed in the rollOverImages parameter must also match the number of names listed in

the rollOverNames parameter. The rollOverImages should be delimited by a comma (default) or by the character listed in the delimiter parameter.

Optional Parameters:

- **baseURL -** The baseURL parameter is the absolute URL to the folder that contains the image files listed in the rollOverImages parameter. By default, images are referenced in the same folder as the page containing the mgiRollOver tag.
- **delimiter -** The delimiter parameter is the character that separates each name and image in the rollOverNames and rollOverImages parameters. The default delimiter is a comma.

Image Mode

The image mode of the mgiRollOver tag has four required parameters and six optional parameters. The tag form is:

```
<mgiRollOver mode="image" name="RollOver Name"
image="Image Path" height="Image Height"
width="Image Width" border="Image Border"
URL="Image Link" target="Link Target" alt="Name"
message="Display Message">
```

Required Parameters:

- **name -** The name parameter is the name of the roll over set to display. The name must match a name listed in the rollOverNames parameter in the "head" mode of mgiRollOver.
- **image -** The image parameter is the absolute or relative path to the base image file to display. The base image file displayes until the roll over function is activated by the cursor moving over the base image.
- **height -** The height parameter is the height of the base image.
- **width -** The width parameter is the width of the base image.

Optional Parameters:

- **mode -** The mode is the function of the mgiRollOver tag. In "**image**" mode, the mgiRollOver tag specifies the image and roll over image to display. "Image" is the default mode of the mgiRollOver tag.
- **border -** The border parameter is the size of the base image's border. The default border is "0" (zero).
- **URL -** The URL parameter is the URL of the page the roll over links to when clicked. By default, the image is not linked.
- **target -** The target parameter is the target of the link listed in the URL parameter.

- **alt -** The alt parameter is the alternative label for the base image (i.e., the label that displays if an image does not load or the visitor has images turned off on the browser).
- **message -** The message parameter is the text that appears in the status bar of the browser when the roll over is activated.

Example Usage and Output

The head and image modes of mgiRollOver must both be used in order for the roll over function to work properly. Enter one mgiRollOver tag in head mode between the HTML <HEAD> tags. The mgiRollOver tag in head mode should not be placed between <SCRIPT> tags. Next, enter one mgiRollOver tag in image mode for each roll over you wish to display. The following is a simple example using both modes of mgiRollOver.

```
<html>

<head>

<title>Rollovers by MGI</title>

<mgiRollOver mode="head" delimiter=","
rollOverNames="about,service,contact"
rollOverImages="abOver.gif,servOver.gif,conOver.gif"
baseURL="http://www.domain.com/images/">

</head>

<body>

<mgiRollOver mode="image" name="about"
alt="About Us" message="Information About Our Company"
image="about.gif" height="32" width="64"
border="0" url="about.mgi"><br>

<mgiRollOver mode="image" name="service"
alt="Services" message="Service Details and Pricing"
image="service.gif" height="32" width="64"
border="0" url="service.mgi"><br>

<mgiRollOver mode="image" name="contact"
alt="Contact Us" message="Phone, Mail and Email Info"
image="contact.gif" height="32" width="64"
border="0"><br>
```

```
</body>

</html>
```

Suggested Usage

- Roll over images

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiRotate Tag

Tag Behavior

Use the mgiRotate tag to randomly rotate the display of a database value, file name, or list item each time a page is served.

Tag Syntax

The mgiRotate tag has three sources. Each source has different required and optional parameters. The sources of mgiRotate are:

- **databaseField** - Database field values are randomly rotated.
- **folder** - Item names in a folder are randomly rotated. *(Note: Hidden files will not be rotated.)*
- **itemList** - Items in a list are randomly rotated.

DatabaseField Source

The databaseField source of the mgiRotate tag has three required parameters and three optional parameters. The tag form is:

```
<mgiRotate source="databaseField" databaseName="Database"
fieldName="Field" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">
```

Required Parameters:

- **source** - The source is the location of the values or items to randomly rotate. The "**databaseField**" source rotates values from a database field.
- **databaseName** - The databaseName is the name of the database that contains the values to rotate.
- **fieldName** - The fieldName is the name of the database field that contains the values to rotate.

Optional Parameters:

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, counter information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Folder Source

The folder source of the mgiRotate tag has two required parameter and no optional parameters. The tag form is:

```
<mgiRotate source="folder" folderLocation="Path">
```

Required Parameters:

- **source -** The source is the location of the values or items to randomly rotate. The "**folder**" source rotates filenames from the specified folder.
- **folderLocation -** The folderLocation is the relative path to the folder that contains the filenames to rotate.

Optional Parameters:

- **None.**

ItemList Source

The itemList source of the mgiRotate tag needs two of three required parameters and two optional parameters. The tag form is:

```
<mgiRotate source="itemList" item="Item"
itemList="Item List" delimiter="Character">
```

Required Parameters:

- **source** - The source is the location of the values or items to randomly rotate. The "**itemList**" source rotates items from a character delimited list of items.
- **item -** The item is one value to rotate. Multiple item parameters may be included in one mgiRotate tag and may be included with itemList parameters. At least one item parameter or one itemList parameter is required.
- **itemList -** The itemList is a string of items separated by a character delimiter. Multiple itemList parameters may be included in one mgiRotate tag and may be included with item parameters. At least one item parameter or one itemList parameter is required.

Optional Parameters:

- **delimiter** - The delimiter is the character that separates each item in the item list. The default value is a comma. If special reserved characters such as tabs and backslashes are delimiters, you must use the appropriate Escaped String format.

Example Usage and Output

DatabaseField Source

```
One of the exciting positions now available is
<mgiRotate source="databaseField" databaseName="Jobs"
fieldName="Position">
```

In this example, current positions in a job database are randomly rotated in the sentence "One of the exciting positions now available is..." The positions are in a database named "Jobs", and in a field named "Position".

Folder Source

```
<mgiSet name="imageLocation">
   Images/<mgiRotate source="folder" folderLocation="Images">
</mgiSet>
<img src={mgiGet name="imageLocation"}>
```

In this example, image filenames located in the folder Images are randomly rotated to cause an image to change at random on a web page.

ItemList Source

```
<mgiRotate source="itemList" item="100"
itemList={mgiPathArgument mode="returnAllValues"}>
```

In this example, numerical values from path arguments are randomly rotated with the static value 100.

Suggested Usage

- Dynamic Content

The mgiSearchDatabase Tag

Tag Behavior

Use the mgiSearchDatabase tag is used to search a database and display the results.

Tag Syntax

The mgiSearchDatabase tag has a beginning tag with one required parameter and thirteen optional parameters, a body, and an ending tag . The tag form is:

```
<mgiSearchDatabase databaseName="Database"
keyFieldName="Field Name" fieldValue="Value"
keyFieldName="Field Name" pageVariableName="Variable Name"
keyFieldName="Field Name" pathArgumentName="Path Argument Name"
keyFieldName="Field Name" postArgumentName="Post Argument Name"
orderByField="Field Name" reverseOrder="Yes/No"
resultsPerPage="Integer" page="Integer"
resultVariableName="Variable Name" odbcDatasource="Source Name"
odbcUsername="Name" odbcPassword="Password">

Template for Results Display

</mgiSearchDatabase>
```

or

```
<mgiSearchDatabase databaseName="Database"
searchString="String" resultsPerPage="Integer"
page="Integer" resultVariableName="Variable Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">

Template for Results Display

</mgiSearchDatabase>
```

Required Parameters:

- **databaseName -** The database is the name of the database to search.

Optional Parameters:

- **keyFieldName -** The keyFieldName is the name of the database field to search. Multiple keyFieldName parameters may be included in one mgiSearchDatabase tag, but each keyFieldName parameter must be **immediately followed** by a fieldValue, pageVariableName, pathArgumentName, or postArgumentName parameter that contains the search criteria for that field. When multiple keyFieldName and value pairs are present, an AND search is performed. To perform an OR search, you must use the searchString parameter.
- **fieldValue -** The fieldValue is the search criteria for the preceding keyFieldName parameter in the mgiSearchDatabase tag.
- **pageVariableName -** The pageVariableName is the name of the page variable that contains the search criteria for the preceding keyFieldName parameter in the mgiSearchDatabase tag.
- **pathArgumentName -** The pathArgumentName is the name of the path argument that contains the search criteria for the preceding keyFieldName parameter in the mgiSearchDatabase tag.
- **postArgumentName -** The postArgumentName is the name of the post argument that contains the search criteria for the preceding keyFieldName parameter in the mgiSearchDatabase tag.
- **searchString -** The search string is a fully formatted search string containing the field names to search, the search criteria, and the search operations. If the searchString parameter is specified, the keyFieldName, pageVariableName, pathArgumentName, postArgumentName, orderByField, and reverseOrder parameters are ignored. Search strings should contain field name and search criteria in the format **keyFieldName='criteria'**. Valid operators are "=" (equals), ">" (greater than), "<" (less than), ">=" (greater than and equal to), and "<=" (less than and equal to). Field name and search criteria pairs should be separated by the search operator "AND", "OR", or "NOT". To order the results from a search string, include the order by field in the format **ORDER BY keyFieldName**. By default, search results are in asending order (ASC). To reverse the order of results in a search string, specify descending order by including DESC after the ORDER BY field. You may not enter quotes ("") in the value of the search string parameter. The following are example search strings:

  ```
  searchString="LastName='J*' OR LastName='L'
  AND Registration='Yes' ORDER BY CustID DESC"

  searchString="FirstName='Ken' AND NOT LastName='Barker'
  ```
- **orderByField -** The orderByField is the name of the database that is used to order search results. Only indexed fields can be used to order search results.
- **reverseOrder -** The reverseOrder parameter specifies if the results are listed in ascending or descending order. If the reverseOrder parameter value is "**Yes**", then results are listed in descending order. If the reverseOrder parameter value is "**No**", then results are listed in ascending order. The default value is "No".
- **resultsPerPage -** The resultsPerPage is the number of search results to display. The

default value is 25. The maximum value of this parameter is controlled by the region settings for the Database Module. The default maximum is 100 and must be specifically overridden in the region settings to be able to return more than 100 records.

- **page** - The page is the set of search results to display. For example, if the resultsPerPage is set to "25", then page 1 displays results 1 to 25 and page 2 displays results 26 to 50. The default value is 1.
- **resultVariableName** - The resultVariableName is the name prepended to result variables that contain information about the search. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName value is "**resultvar**". Result variables are created and available anywhere on the page after the mgiSearchDatabase tag.
    - **resultvar_ResultCount** - A page variable with the total number of results.
    - **resultvar_FirstIndex** - A page variable with the index number of the first result shown on the page.
    - **resultvar_LastIndex** - A page variable with the index number of the last result shown on the page.
    - **resultvar_PrevPage** - A page variable with the number of the page preceding the current page. The value of this variable is zero (0) if the current page is 1.
    - **resultvar_Page** - A page variable with the number of the page currently displaying
    - **resultvar_NextPage** - A page variable with the number of the page following the current page. The value of this variable is zero (0) if the current page is the last page.
    - **resultvar_LastPage** - A page variable with the number of the last page.
- **displayPrecision** - The displayPrecision parameter specifies the precision that should be used to display a decimal number. The format is "fieldName, precision". Multiple displayPrecision parameters may be specified, one for each decimal number field that is to be formatted with a certain number of decimal places. If the specified precision has less decimal places than the actual number, the number is rounded. If the specified precision has more decimal places than the actual number, the number is padded with trailing zeroes. For example, use displayPrecision="Price, 2" to display the Price field formatted with two decimal places.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, database information will be searched in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the

ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.

- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Results Display:

The body of the mgiSearchDatabase tag contains a template display that is repeated for each search result. To retrieve a field value for a result, use the placeholder format of **&mgiDBFieldEnterFieldNameHere;** in the body of the mgiSearchDatabase tag. Each placeholder will be replaced with the specific information in that field for each result. For example, to include values from the FirstName field in the database for each result, you would enter the following in your template:

```
&mgiDBFieldFirstName;
```

To display the unique database ID for a result, use the field name "MGIIDFIELD".

Example Usage and Output

```
<mgiSearchDatabase databaseName="Homes"
keyFieldName="Style" postArgumentName="StyleSearch"
resultsPerPage="25" page="1"
resultVariableName="HomeSearch">

<P>House Address: &mgiDBFieldAddress;
<P>House Style: &mgiDBFieldStyle;
<P>Price: &mgiDBFieldPrice;
<BR>

</mgiSearchDatabase>

Results found: <mgiGet name="HomeSearch_ResultCount">
```

In this example, a user selects search criteria for a home based on home styles. The Homes database is searched and the following would display if there were three matching results:

```
House Address: 128 B Lane Ave
House Style: Ranch
Price: 120,000

House Address: 8827 Main St
House Style: Ranch
Price: 80,000
```

```
House Address: 3289 Octavia St
House Style: Ranch
Price: 200,000

Results found: 3
```

In the following example, the Homes database is searched using a string search with ordering.

```
<mgiSet name="SearchString">
Style='<mgiPostArgument name="StyleSearch">'
ORDER BY Price DESC
</mgiSet>

<mgiSearchDatabase databaseName="Homes"
searchString={mgiGet name="SearchString"}
resultsPerPage="25" page="1"
resultVariableName="HomeSearch">

<P>House Address: &mgiDBFieldAddress;
<P>House Style: &mgiDBFieldStyle;
<P>Price: &mgiDBFieldPrice;
<BR>

</mgiSearchDatabase>
```

The results of the string search would be:

```
House Address: 3289 Octavia St
House Style: Ranch
Price: 200,000

House Address: 128 B Lane Ave
House Style: Ranch
Price: 120,000

House Address: 8827 Main St
House Style: Ranch
Price: 80,000
```

Suggested Usage

- Database Searches

The mgiSendMail Tag

Tag Behavior

Use the mgiSendMail tag to send a formatted email including attachments from the server.

Tag Syntax

The mgiSendMail tag has two modes. Each mode has different required and optional parameters. The modes of mgiSendMail are:

- **sendMail** - Sends a formatted email.
- **admin** - Creates a web-based interface to manage failed mail.

SendMail Mode

The sendMail mode of the mgiSendMail tag has a beginning tag with three required parameters and eleven optional parameters, a body, and an end tag. The tag form is:

```
<mgiSendMail mode="sendMail" to="Email Address"
from="Email Address" cc="Email Address" bcc="Email Address"
replyTo="Email Address" gmtOffset="Hours"
subject="Title" mailServer="SMTP Server"
attachmentData="Data" attachmentFileLocation="File Path"
sendAsHTML="Yes/No" headerName="Name" headerValue="Value">
MGI Tags, Formatted Text and HTML (when sendAsHTML is set to "Yes")
</mgiSendMail>
```

Required Parameters:

- **to** - The to parameter is the email recipient's email address or a comma delimited list of email addresses for multiple recipients. The to value must be a valid email format. Multiple "to" parameters may also be included in one mgiSendMail tag.
- **from** - The from parameter is the sender's email address. The from value must be a valid email format.
- **mailServer** - The mailServer is the outgoing SMTP server that is used to send the email. A global outgoing mail server can be enabled by default or always in the region preferences for your domain. If you do not have access to the region preferences for your domain, contact the server administrator. If a global mail server is enabled as "always", then the mail server preference always overrides the mailServer parameter in your region.

Optional Parameters:

- **mode** - The mode is the function of the mgiSendMail tag. In "**sendMail**" mode, the mgiSendMail tag sends a formatted email. "SendMail" is the default mode of the mgiSendMail tag.
- **cc** - The cc parameter is the email address or comma delimited list of email addresses for multiple carbon copy recipients. Multiple "cc" parameters may also be included in one mgiSendMail tag.
- **bcc** - The bcc parameter is the email address or comma delimited list of email addresses for multiple blind carbon copy recipents. Blind carbon copy addresses do not appear in the email header. Multiple "bcc" parameters may also be included in one mgiSendMail tag.
- **replyTo** - The replyTo parameter is the email addresses where recipient email replies are sent. If a replyTo address is supplied, the address must be in a valid email format.
- **subject** - The subject is the title of the email. The default subject is blank.
- **gmtOffset** - The gmtOffset is the whole or half number of hours the time of the email is offset from Greenwich Mean Time preceeded by a positive (+) or negative (-) sign (e.g., "+5" or "-3.5"). Only half hours may be entered. Other tenths of an hour may not be entered. When the GMTOffset parameter is not present, the local time of the server is sent. A global GMTOffset can be enabled in the region preferences for your domain. If you do not have access to the region preferences for your domain, contact the server administrator. Including a gmtOffset parameter overrides the gmtOffset in the region preferences.
- **attachmentData** - The attachmentData is a byte stream of data that is sent as an attachment to the email. Multiple "attachmentData" parameters can be included in one mgiSendMail tag.
- **attachmentFileLocation** - The attachmentFileLocation is the relative path to the file that is attached to the email. Multiple "attachmentFileLocation" parameters can be included in one mgiSendMail tag.
- **sendAsHTML** - The sendAsHTML parameter determines whether the body of the mgiSendMail tag is sent as HTML or plain text. If the value of the sendAsHTML parameter is "**Yes**", then the body of the mgiSendMail tag is sent as HTML and can be read as HTML by HTML-enabled email clients. If the value of the sendAsHTML parameter is "**No**", then the body of the mgiSendMail tag is sent as plain text (even if the body includes HTML tags). The default value is "No". Note: the sendAsHTML parameter is ignored if the headerName and headerContent parameters are included.
- **headerName** - The headerName is the name of the email header attribute (e.g., "Content-Type"). Multiple headerName and headerValue pairs are allowed.
- **headerValue**- The headerValue is the name of the value of the header attribute (e.g., "text/plain; charset=US-ASCII"). Multiple headerName and headerValue pairs are allowed.

Admin Mode

The admin mode of the mgiSendMail tag has a beginning tag with one required parameter and one optional parameter, a body, and an end tag. The tag form is:

```
<mgiSendMail mode="admin" displayLimit="Integer">
</mgiSendMail>
```

Required Parameters:

- **mode** - The mode is the function of the mgiSendMail tag. In "**admin**" mode, the mgiSendMail tag creates a web-based interface that allows you to manage failed emails from the sendMail mode of mgiSendMail.

Optional Parameters:

- **displayLimit** - The displayLimit is the maximum number of emails displayed per screen in the administration interface. The default value is 10. The maximum value of the displayLimit parameter is 50.

Example Usage and Output

SendMail Mode

```
<mgiSendMail to="info@domain.com" from="webmaster@domain.com"
subject="Information Request" mailServer="mail.isp.com"
replyTo="sales@domain.com">
Information Request
*******************

    Name: <mgiPostArgument name="Name">
   Phone: <mgiPostArgument name="Phone">
   Email: <mgiPostArgument name="Email">
</mgiSendMail>
```

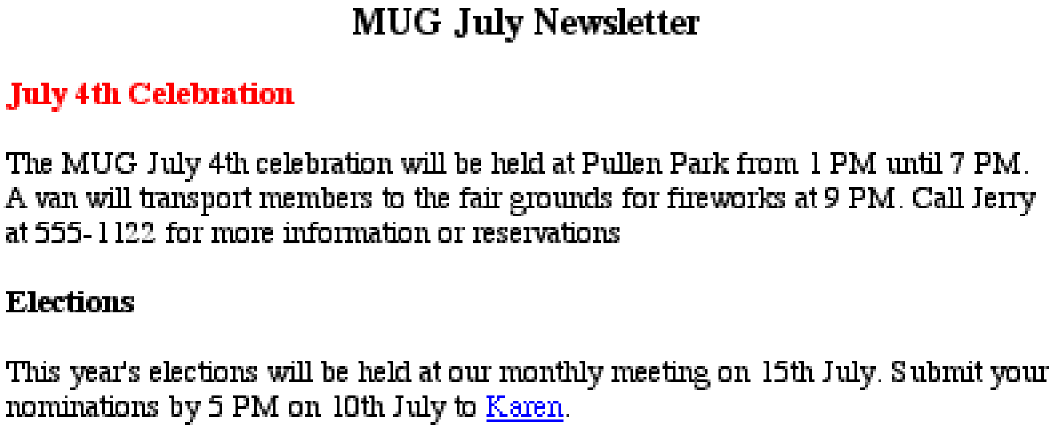
The mgiSendMail tag in this example is used to send a formatted email from a form submission. When the email is received at "info@domain.com", it will be in the following format:

```
Information Request
*******************

    Name: Jane Lowe
   Phone: (472) 234-7331
   Email: jlowe@aol.com
```

SendMail Mode - Send As HTML

```
<mgiSendMail to="post@list.domain.com" from="info@domain.com"
subject="MUG July Newsletter" mailServer="mail.domain.com"
sendAsHTML="Yes">

<html>

<head>
<title>MUG July Newsletter</title>
</head>

<body>
<H3 align="center">MUG July Newsletter</H3>
<p><font color="red"><b>July 4th Celebration</b></font>
<p>The MUG July 4th celebration will be held at Pullen Park
from 1 PM until 7 PM.  A van will transport members to the
fair grounds for fireworks at 9 PM. Call Jerry at
555-1122 for more information or reservations
<p><b>Elections</b>
<p>This year's elections will be held at our monthly
meeting on 15th July. Submit your nominations by 5 PM on
10th July to <a href="mailto:karen@domain.com">Karen</a>.
</body>

</html>

</mgiSendMail>
```

This mgiSendMail tag in this example sends an email formatted with HTML tags. Any email client that has an HTML reader will display the email as HTML. If the email client does not have an HTML reader, the email will be displayed as plain text (including the HTML tags). The following is an example of an HTML display in an email client.



MUG July Newsletter

July 4th Celebration

The MUG July 4th celebration will be held at Pullen Park from 1 PM until 7 PM. A van will transport members to the fair grounds for fireworks at 9 PM. Call Jerry at 555-1122 for more information or reservations

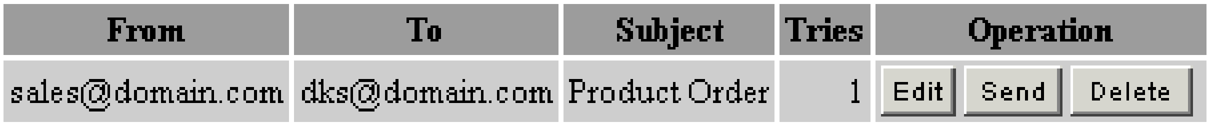
Elections

This year's elections will be held at our monthly meeting on 15th July. Submit your nominations by 5 PM on 10th July to Karen.

Admin Mode

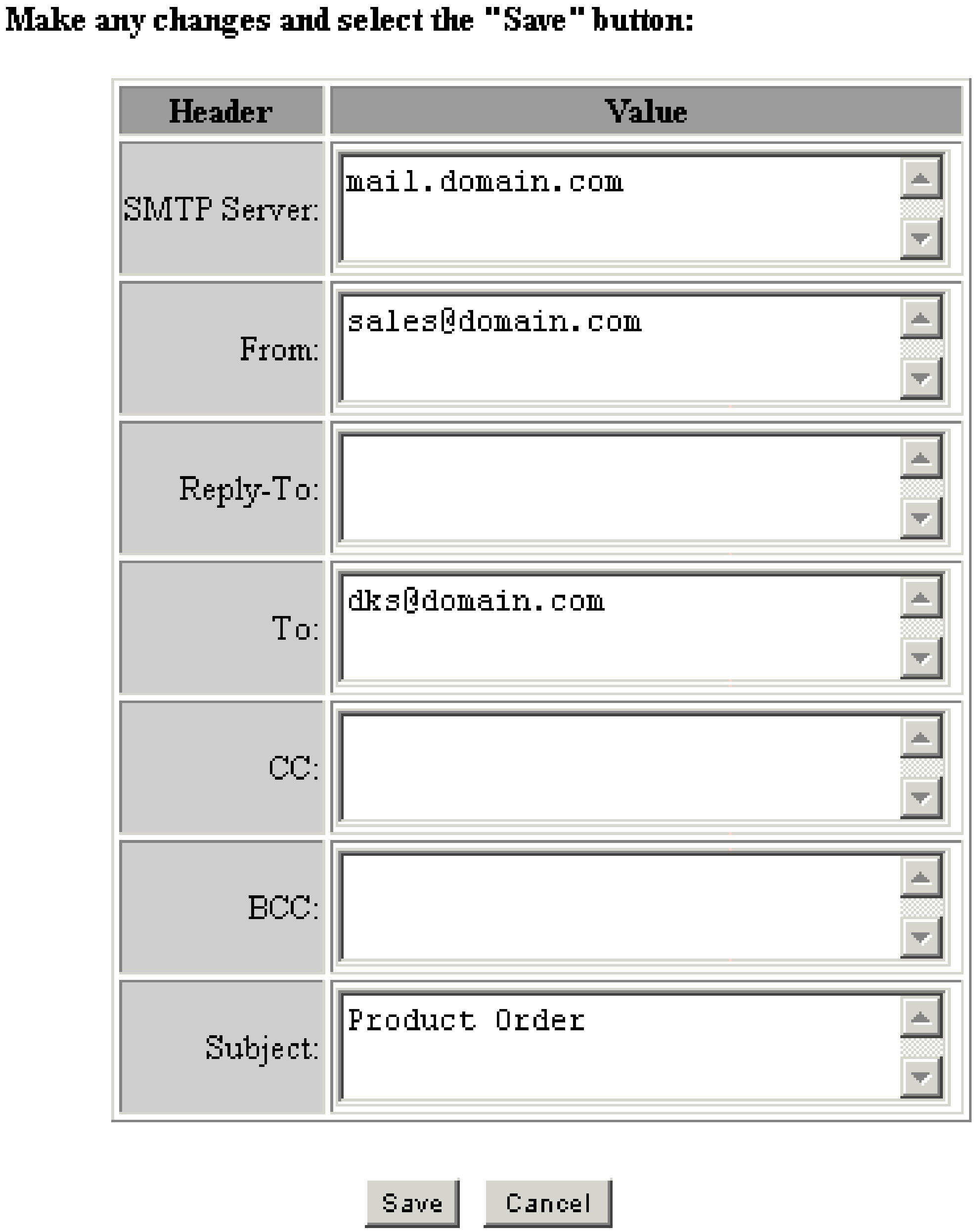
```
<mgiSendMail mode="admin" displayLimit="25">
</mgiSendMail>
```

When you access a page with the mgiSendMail tag in admin mode, a database of failed emails displays.



From	To	Subject	Tries	Operation
sales@domain.com	dks@domain.com	Product Order	1	Edit Send Delete

To resend the failed email without modifications, click the "Send" button. To delete the email, click the "Delete" button. To modify the email before it is re-sent, click the "Edit" button. The edit screen displays the following with the full text of the email message shown under "Message Text:"



Make any changes and select the "Save" button:

Header	Value
SMTP Server:	mail.domain.com
From:	sales@domain.com
Reply-To:	
To:	dks@domain.com
CC:	
BCC:	
Subject:	Product Order

Message text:

Modify the SMTP Server, From, Reply-To, To, CC, BCC, or Subject Fields and click the "Save" button. Clicking the Save or Delete button returns you to the original admin interface. To re-send the modified email, click the "Send" button.

Suggested Usage

- Form Processing
- Dynamic Emails
- Online Listserv Registration

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiSendOrder Tag

Tag Behavior

Use the mgiSendOrder tag to send an email or create a text file containing the visitor's shopping basket order.

Tag Syntax

The mgiSendOrder tag has a beginning tag with five required parameters and thirteen optional parameters, a body, and an ending tag. The tag form is:

```
<mgiSendOrder handle="Configuration Name"
shoppingBasketURL="URL" priceRule="Price Rule Name"
shippingRule="Shipping Rule Name" taxRule="Tax Rule Name"
customerLocation="Location List" to="Email Address"
from="Email Address" cc="Email Address" bcc="Email Address"
replyTo="Email Address" Subject="Title" mailServer="SMTP Server"
attachmentData="Data" attachmentFileLocation="File Path"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">

Customer Info

<!-- Begin Template -->

Repeating Product Template

<!-- End Template -->

Order Totals

</mgiSendOrder>
```

Required Parameters:

- **handle -** The handle is the name of the shopping basket configuration to use. Shopping basket configurations are defined using the admin mode of the mgiShoppingBasket tag.
- **shoppingBasketURL -** The shoppingBasketURL is the absolute address to the folder

region containing the visitor's shopping basket (i.e., the region containing the mgiShoppingBasket tag that created the visitor's shopping basket).

- **to** - The to parameter is the email recipient's email address or a comma delimited list of email addresses for multiple recipients. Multiple "to" parameters may also be included in one mgiSendOrder tag.
- **from** - The from parameter is the sender's email address.
- **mailServer** - The mailServer is the outgoing SMTP server that is used to send the email. A global outgoing mail server can be enabled by default or always in the region preferences for your domain. If you do not have access to the region preferences for your domain, contact the server administrator. If a global mail server is enabled as "always", then the mail server preference always overrides the mailServer parameter in your region.

Optional Parameters:

- **priceRule** - The priceRule is the name of the price algorithm to apply to products in the final shopping basket. Price rules included in the mgiSendOrder tag must have a scope of "basket" in order to be applied to all products that are purchased. Price rules are defined in the admin mode of the mgiShoppingBasket tag.
- **shippingRule** - The shippingRule rule is the name of the shipping algorithm to apply to products in the final shopping basket. Shipping rules included in the mgiSendOrder tag must have a scope of "basket" in order to be applied to all products that are purchased. Shipping rules are defined in the admin mode of the mgiShoppingBasket tag.
- **taxRule** - The taxRule is the name of the tax algorithm to apply to products in the final shopping basket. Tax rules included in the mgiSendOrder tag must have a scope of "basket" in order to be applied to all products that are purchased. Tax rules are defined in the admin mode of the mgiShoppingBasket tag.
- **customerLocation** - The customerLocation is a comma delimited list of the customer's location that is used to determine the customer's tax according to the tax rule. If there is a chain of tax rules, the first location in the list should correspond to the first tax rule, the second location in the list should correspond to the second tax rule, etc.
- **cc** - The cc parameter is the email address or comma delimited list of email addresses for multiple carbon copy recipients. Multiple "cc" parameters may also be included in one mgiSendOrder tag.
- **bcc** - The bcc parameter is the email address or comma delimited list of email addresses for multiple blind carbon copy recipents. Blind carbon copy addresses do not appear in the email header. Multiple "bcc" parameters may also be included in one mgiSendOrder tag.
- **subject** - The subject is the title of the email. The default subject is blank.
- **replyTo** - The replyTo parameter is the email addresses where recipient email replies are sent.

- **attachmentData -** The attachmentData is a byte stream of data that is sent as an attachment to the email. Multiple "attachmentData" parameters can be included in one mgiSendOrder tag.
- **attachmentFileLocation -** The attachmentFileLocation is the relative path to the file that is attached to the email. Multiple "attachmentFileLocation" parameters can be included in one mgiSendOrder tag.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, shopping basket information will be retrieved from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Technical Notes

- **Price, Shipping and Tax Rules -** Price, shipping and tax rules can be configured in the admin mode of mgiShoppingBasket.
  - Shopping basket rules are shared across all shopping basket handles within a region.
  - Only "basket" type rules may be applied in the mgiShoppingBasket tag. Apply "item" type rules in the mgiBuyMe tag.
- **Custom Shopping Basket Display -** The body of the mgiSendOrder tag contains a template display that is repeated for each product in the order. If the body of the mgiSendOrder tag is left empty, the default template will be used. To create a custom order format, use HTML, MGI, text , and the following placeholders between the "Begin Template" marker and the "End Template" marker in the body of the mgiSendOrder tag. Information that does not repeat for each product should be placed outside the "Begin Template" marker and the "End Template" marker. The placeholders will be replaced with each product's specific information.
  - **&mgiDBFieldQuantity;** is the quantity of the product.
  - **&mgiDBFieldQuantityMultiplier;** is the quantity multiplier of the product.
  - **&mgiDBFieldProductID;** is the product ID.

  - **&mgiDBFieldName;** is the product name (short description).
  - **&mgiDBFieldDescription;** is the product description
  - **&mgiDBFieldPrice;** is the product price.
  - **&mgiDBFieldShipping;** is the product shipping cost.
  - **&mgiDBFieldQualifier1;** is the first product qualifier.
  - **&mgiDBFieldQualifier2;** is the second product qualifier.
  - **&mgiDBFieldQualifier3;** is the third product qualifier.
  - **&mgiDBFieldQualifier4;** is the fourth product qualifier.
  - **&mgiDBFieldQualifier5;** is the fifth product qualifier.
  - **&mgiDBFieldWeight;** is the product weight.
  - **&mgiSBItemPriceTotal;** is the price of a product multiplied by the quantity purchased and includes any price rules that have been applied.
  - **&mgiSBItemShippingTotal;** is the shipping price of a product after shipping rules have been applied.
  - **&mgiSBItemTotal;** is the total price of a product (price subtotal plus shipping subtotal) after price and shipping rules have been applied.
  - **&mgiSBSubtotal;** is the price of the order after the price rule(s) have been applied.
  - **&mgiSBTax;** is the tax on the order after the tax rule(s) have been applied.
  - **&mgiSBShippingTotal;** is the shipping price of the order after the shipping rule(s) have been applied.
  - **&mgiSBTotal;** is the total price of the order after all rules have been applied.
- **Default Payment, Billing and Shipping Post Arguments** - If you use the mgiCollectUserInfo tag to collect payment, billing, and shipping information, then you can display that information in the custom order email by using the following post argument names with the mgiPostArgument tag.
  - **PaymentMethod** - Shopping basket payment method. Values include "Credit Card", "Purchase Order", and "Check".
  - **CardType** - Credit card type. Values include "Visa", "Mastercard", "Disover/Novus", "American Express", "Carte Blanche", "Diner's Club", "Bankcard", and/or "JCB" depending on shopping basket configuration.
  - **CCNumber** - Credit card number.
  - **ExpMo** - Credit card expiration month.
  - **ExpYe** - Credit card expiration year.
  - **CCName** - Name that appears on credit card.
  - **PONumber** - Purchase order number.
  - **POCompany** - Purchase order company.

- **BName -** Billing name
- **BCompany** - Billing company
- **BAddress1** - Billing address
- **BCity** - Billing city
- **BState -** Billing state
- **BProvince** - Billing province
- **BZipCode** - Billing zip code
- **BCountry** - Billing country
- **BPhone** - Billing phone
- **BFax** - Billing fax
- **BEmail** - Billing email
- **SCompany** - Shipping company
- **SAddress1** - Shipping address
- **SCity** - Shipping city
- **SState -** Shipping state
- **SProvince** - Shipping province
- **SZipCode** - Shipping zip code
- **SCountry** - Shipping country
- **SPhone -** Shipping phone
- **SFax** - Shipping fax
- **SEmail** - Shipping email

Example Usage and Output

Default Order Email

```
<mgiSendOrder handle="Books"
shoppingBasketURL="http://www.domain.com/"
to="orders@domain.com" from="onlineorder@domain.com"
subject="Online Order" mailServer="mail.domain.com">
</mgiSendOrder>
```

In this example, the mgiSendOrder tag will email an order in the default format to orders@domain.com with the subject "Online Order". The default order email contains the payment, billing, shipping, and order information in this format:

```
Payment Information
-------------------
```

```
 Payment Method:  Credit Card
           Type:  VISA
         Number:  1111222233334444
Expiration Date:  12/2001
           Name:  Samuel Smith

Billing Information
-------------------
           Name:  Samuel Smith
        Company:  RedStar
        Address:  1100 Main Street
           City:  Charlotte
          State:  North Carolina
       Zip Code:  25849
        Country:  United States of America
          Phone:  704-123-4567
            Fax:  704-123-4568
          Email:  ssmith@redstar.com

Product Information
--------------------
       Quantity:  1
     Product ID:  0-688-16199-5
           Name:  How to Cook Meat
    Total Price:  $28.00

       Quantity:  1
     Product ID:  0-060-16010-1
           Name:  The New York Times Cook Book
    Total Price:  $26.00

       Subtotal:  $54.00
            Tax:  $0.00
       Shipping:  $0.00
          Total:  $54.00
```

Custom Order Email

```
<mgiSendOrder handle="Default"
shoppingBasketURL="http://www.domain.com/"
to="orders@domain.com" from="webmaster@domain.com"
mailServer="mail.domain.com"
subject="Test MGI2 SB Order">

Payment
```

```
-------
       Type: <mgiPostArgument name="CardType">
     Number: <mgiPostArgument name="CCNumber">
        Exp: <mgiPostArgument name="EXPMO">
<mgiPostArgument name="EXPYE">

Billing
-------
       Name: <mgiPostArgument name="bName">
    Company: <mgiPostArgument name="bCompany">
    Address: <mgiPostArgument name="bAddress1">
             <mgiPostArgument name="bCity">
<mgiPostArgument name="bState">
<mgiPostArgument name="bZipCode">
             <mgiPostArgument name="bCountry">
      Phone: <mgiPostArgument name="bPhone">
        Fax: <mgiPostArgument name="bFax">
      Email: <mgiPostArgument name="bEmail">

Order
-----
  Order Ref: <mgiPathArgument name="mgiToken">
 Order Date: <mgiDate>
<!-- Begin Template -->

 Product ID: &mgiDBFieldProductID;
        Qty: &mgiDBFieldQuantity;
    Product: &mgiDBFieldName;
 Price Each: $&mgiDBFieldPrice;
 Item Total: $&mgiSBItemPriceTotal;
<!-- End Template -->

        Tax: $&mgiSBTax;
      Total: $&mgiSBTotal;
</mgiSendOrder>
```

If you prefer a different email format, use the mgiSendOrder placeholders and mgiPostArgument tags to customize the email layout. Placeholders between the "Begin Template" and "End Template" comments will be repeated for each item ordered and replaced with the item's specific information. If you use the mgiCollectUserInfo tag to collect payment, billing, and shipping information, then you can display that information in the custom order email by using specific post arguments (listed above under Technical Notes). However, if you created a custom form to collect payment, billing and shipping information, then use your post argument names to display that information in the custom order email. Custom order emails are sent as text, therefore you may use spaces to align information.

The **mgiSendOrder email is not "post-processed"** which means that you cannot perform MGI functions such as math calculations with placeholders in the body of the mgiSendOrder tags. If you need to perform such functions with placeholder information, we suggest you perform the function in the mgiConfirmOrder tag and use a hidden post argument to transfer the result to the mgiSendOrder page. The custom order email in this example has the following format:

```
Payment
-------
       Type: VISA
     Number: 1111222233334444
        Exp: 12 2001

Billing
-------
       Name: Samuel Smith
    Company: RedStar
    Address: 1100 Main Street
             Charlotte North Carolina 25849
             United States of America
      Phone: 704-123-4567
        Fax: 704-123-4568
      Email: ssmith@redstar.com

Order
-----
  Order Ref: 36Q03NHIB14GT5
 Order Date: Tuesday, August 21, 2001

 Product ID: 0-688-16199-5
        Qty: 1
    Product: How to Cook Meat
 Price Each: $28.00
 Item Total: $28.00

 Product ID: 0-060-16010-1
        Qty: 1
    Product: The New York Times Cook Book
 Price Each: $26.00
 Item Total: $26.00

        Tax: $0.00
      Total: $54.00
```

Suggested Usage

- Shopping Basket

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiSet Tag

Tag Behavior

Use the mgiSet tag to create a variable with a specific scope and value (see also mgiGet).Variables are containers for text, numbers, calculations, results, etc. that can be used after they are created (i.e., set) throughout a single page (page variables) or throughout an entire web site (site variables).

Tag Syntax

The mgiSet tag has a beginning tag with one required parameter and five optional parameters, a body, and an ending tag. The tag form is:

```
<mgiSet name="Name" scope="Page/Site" defaultValue="Value"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
Variable Value
</mgiSet>
```

Required Parameters:

- **name** - The name is the name of the variable to create. Variable names must be unique within a scope.

Optional parameters:

- **scope** - The scope is the type of variable to create. "**Page**" variables are are created at the location they are set on a page and are available to display until the end of the page. "**Site**" variables are created and stored in an internal MGI database and are available to display after they are set on any page in a web site. The default scope is "Page".
- **defaultValue** - The defaultValue is the text that is set in the variable if the variable value is blank. If a default value is not specified, an empty string (blank) is set.
- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, site variable information will be retrieved from the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.

- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Example Usage and Output

```
<mgiSet name="CustomerSubject">
Welcome <mgiPostArgument name="FirstName">
<mgiPostArgument name="LastName">
</mgiSet>

<mgiSendMail to={mgiPostArgument name="Email"}
from="registration@domain.com"
subject={mgiGet name="CustomerSubject"}
mailserver="mail.domain.com">
<mgiGet name="CustomerSubject">

You are now registered for our online service.
</mgiSendMail>
```

Variables are so versatile that you could use them in combination with almost any other MGI tag. In this example, a page variable is used to combine information from a form submission. The information is then used in the subject of an email that is sent to a customer.

Suggested Usage

- Storage of Information
- Combining Information from Multiple Sources (Post Arguments, Path Arguments, Variables, etc.)
- Site-wide Includes

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiSetCookie Tag

Tag Behavior

Use the mgiSetCookie tag to set a cookie with a specified value on a visitor's computer for later retrieval (see also mgiGetCookie).

Tag Syntax

The mgiSetCookie tag has a beginning tag with one required parameter and three optional parameters, a body, and an ending tag. The tag form is:

```
<mgiSetCookie name="Name" expiration="Julian Day"
useEncryption="Yes/No" defaultValue="Text">
Cookie Value
</mgiSetCookie>
```

Required Parameters:

- **name -** The name is the name of the cookie that is set on the visitor's computer. Cookie names must be unique.

Optional Parameters:

- **expiration -** The julianDayExpiration is the julian day the cookie expires from the visitor's computer. (see mgiJulianDay)
- **useEncryption -** The useEncryption parameter specifies whether the cookie value is encrypted. If the useEncryption parameter value is "**Yes**", then the cookie value is encrypted. If the useEncryption parameter value is "**No**", then the cookie value is not encrypted. The default value is "No."
- **defaultValue -** The defaultValue is the value that is set if the cookie value is blank. If a default value is not specified, an empty string (blank) is set.

Example Usage and Output

```
<mgiSetCookie name="Customer">
<mgiPostArgument name="FirstName">
<mgiPostArgument name="LastName">
</mgiSetCookie>
```

In this example, a customer's information from a form submission is stored in a cookie for later retrieval.

```
<mgiSetCookie name="Password" useEncryption="Yes"
expiration={mgiJulianDay month="December" day="31" year="2000"}>
<mgiPostArgument name="Password">
</mgiSetCookie>
```

In this example, a user's password is encrypted and stored in a cookie on their machine. The cookie will expire on December 31, 2000.

Suggested Usage

- Repeat Customer Information
- Login Information

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiSetResponseHeader Tag

Tag Behavior

Use the mgiSetResponseHeader tag to create a header with a specified value in the HTTP response of the page.

Tag Syntax

The mgiSetResponseHeader tag has a beginning tag with one required parameter and no optional parameters, a body, and an ending tag. The tag form is:

```
<mgiSetResponseHeader name="Name">
Header Value
</mgiSetResponseHeader>
```

Required Parameters:

- **name -** The name is the name of the header that is created.

Optional Parameters:

- **None**

Example Usage and Output

```
<mgiSetResponseHeader name="Cache-Control">
No-Cache
</mgiSetResponseHeader>
```

In this example, a Cache-Control header is added to the page to eliminate caching from proxy servers, etc.

```
<mgiSetResponseHeader name="Content-Type">
Image/Jpeg
</mgiSetResponseHeader>
```

In this example, the Content-Type header of the page is changed to image/jpeg. This provides a way to override the default Content-Type that MGI returns.

Suggested Usage

- Cache Control
- Custom Content Types

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiShoppingBasket Tag

Tag Behavior

Use the mgiShoppingBasket tag to display the contents of a visitor's shopping basket and define shopping basket configurations.

Tag Syntax

The mgiShoppingBasket tag has a beginning tag with one required parameter and six optional parameters, a body, and an ending tag. The tag form is:

```
<mgiShoppingBasket handle="Configuration Name"
mode="Mode" priceRule="Price Rule Name"
shippingRule="Shipping Rule Name"
odbcDatasource="Source Name" odbcUsername="Name"
odbcPassword="Password">
Custom Shopping Basket Product Template
</mgiShoppingBasket>
```

Required Parameters:

- **handle** - The handle is the name of the shopping basket configuration to use. Shopping basket configurations are defined using the admin mode of the mgiShoppingBasket tag. When using admin mode, the handle parameter is not required.

Optional Parameters:

- **mode** - The mode is the function of the mgiShoppingBasket tag. In "**admin**" mode, the shopping basket creates a web-based interface that allows you to configure multiple shopping baskets with options for tracking, customer information, price rules, shipping rules, tax rules, inventory and order processing. When using admin mode, the handle parameter is not required.
- **priceRule** - The priceRule is the name of the price algorithm to apply to products in the shopping basket. Price rules included in the mgiShoppingBasket tag must have a scope of "basket" in order to be applied to all products in the shopping basket. Price rules are defined in the admin mode of the mgiShoppingBasket tag.
- **shippingRule** - The shippingRule rule is the name of the shipping algorithm to apply to products in the shopping basket. Shipping rules included in the mgiShoppingBasket

tag must have a scope of "basket" in order to be applied to all products that are purchased.

- **odbcDatasource** (NT only) - The odbcDatasource is the name of datasource on the server that provides access information for an external ODBC database. If the odbcDatasource parameter is included, shopping basket information will be stored in the specified ODBC database rather than the internal MGI database. Inquire with the server administrator for additional information about the use of ODBC databases. If you include the odbcDatasource parameter, the odbcUsername and odbcPassword parameters are required.
- **odbcUsername** (NT only) - The odbcUsername is the username required to access the ODBC datasource. The odbcUsername parameter is required if you include the odbcDatasource parameter.
- **odbcPassword** (NT only) - The odbcPassword is the code required to access the ODBC datasource. The odbcPassword parameter is required if you include the odbcDatasource parameter.

Technical Notes

- **Custom Shopping Basket Display** - The body of the mgiShoppingBasket tag contains a template display that is repeated for each product in the shopping basket. If the body of the mgiShoppingBasket tag is left empty, the default template will be used. To create a custom shopping basket display, use HTML, MGI, text , and the following placeholders in the body of the mgiShoppingBasket tag. The placeholders will be replaced with each product's specific information.
    - **&mgiDBFieldQuantity;** is the quantity of the product.
    - **&mgiDBFieldQuantityMultiplier;** is the quantity multiplier of the product.
    - **&mgiDBFieldProductID;** is the product ID.
    - **&mgiDBFieldName;** is the product name (short description).
    - **&mgiDBFieldDescription;** is the product description
    - **&mgiDBFieldPrice;** is the product price.
    - **&mgiDBFieldShipping;** is the product shipping cost.
    - **&mgiDBFieldQualifier1;** is the first product qualifier.
    - **&mgiDBFieldQualifier2;** is the second product qualifier.
    - **&mgiDBFieldQualifier3;** is the third product qualifier.
    - **&mgiDBFieldQualifier4;** is the fourth product qualifier.
    - **&mgiDBFieldQualifier5;** is the fifth product qualifier.
    - **&mgiDBFieldWeight;** is the product weight.
    - **&mgiSBDeleteTrashcan;** is a clickable trashcan icon that removes the product

from the shopping basket.

  - **&mgiSBDeleteRadioButtons;** is a set of Keep/Remove radio buttons that allows the product to be removed from the shopping basket.
  - **&mgiSBModifiableQuantity;** is a text input containing the quantity purchased for the product. This value can be modified.
  - **&mgiSBItemPriceTotal;** is the price of a product multiplied by the quantity purchased and includes any price rules that have been applied.

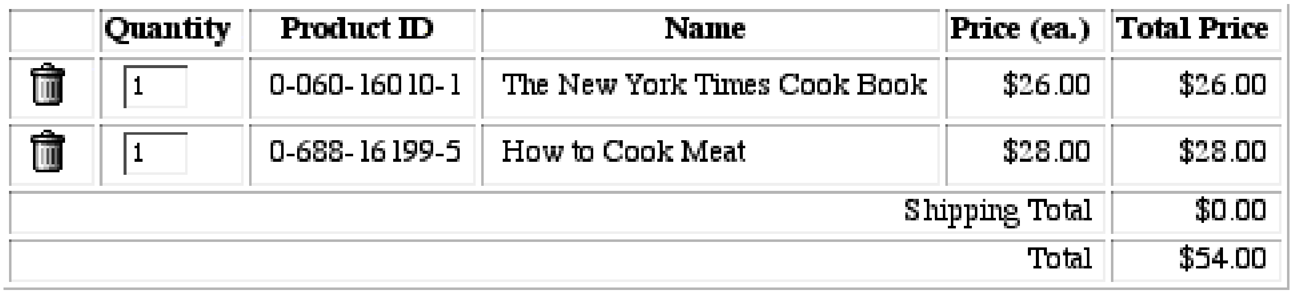
- **Variables -** The mgiShoppingBasket tag sets the following page variables. The value of these page variables can be displayed anywhere on the page after the mgiShoppingBasket tag by using an mgiGet tag.
  - **mgiSBShippingTotal** is the shipping price of the order after all shipping rules have been applied.
  - **mgiSBTotal** is the total price of the order after all price rules have been applied.
- **Price, Shipping and Tax Rules -** Price, shipping and tax rules can be configured in the admin mode of mgiShoppingBasket.
  - Shopping basket rules are shared across all shopping basket handles within a region.
  - Only "basket" type rules may be applied in the mgiShoppingBasket tag. Apply "item" type rules in the mgiBuyMe tag.

Example Usage and Output

Default Shopping Basket

```
<mgiToken>

<form action="shoppingbasket.mgi" method="post">

<mgiShoppingBasket handle="Books">
</mgiShoppingBasket>

<mgiButton name="Modify Products">

</form>

<a href="checkout.mgi">Checkout Now</a>

</mgiToken>
```

In this example, the mgiShoppingBasket tag will display the contents of the shopping basket based on the "Books" configuration using the default shopping basket display. The default shopping basket displays a trashcan icon to delete items, a modifiable quantity field, the product ID, product name, product price, extended product price, shipping total and total (does not include tax which is added in the mgiConfirmOrder tag).



	Quantity	Product ID	Name	Price (ea.)	Total Price
🗑	1	0-060-16010-1	The New York Times Cook Book	$26.00	$26.00
🗑	1	0-688-16199-5	How to Cook Meat	$28.00	$28.00
				Shipping Total	$0.00
				Total	$54.00

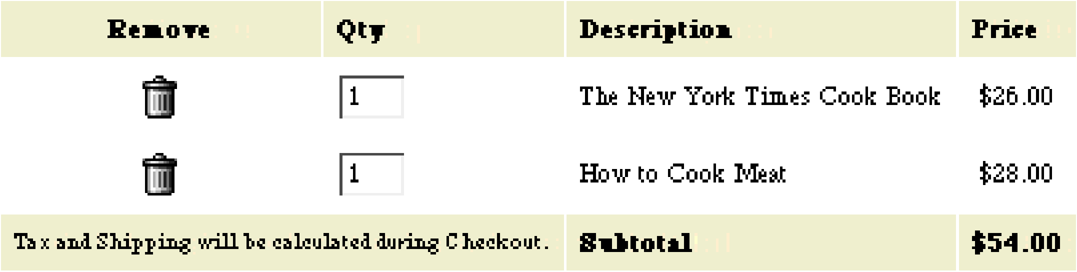
Custom Shopping Basket

```
<form action="shoppingbasket.mgi" method="post">

<p><table cellpadding="5" cellspacing="1" border="0">
<tr bgcolor="#EEEECC">
<td align="center"><font size="-1"
face="helvetica, arial, sans-serif">
<b>Remove</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Qty</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Description</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Price</b></font></td>
</tr>

<mgiShoppingBasket handle="Default">

<tr>
<td align="center">&mgiSBDeleteTrashcan;</td>
<td>&mgiSBModifiableQuantity;</td>
<td><font size="-1" face="helvetica, arial, sans-serif">
&mgiDBFieldName;</font></td>
<td align="center">
<font size="-1" face="helvetica, arial, sans-serif">
$&mgiSBItemPriceTotal;</font></td>
</tr>

</mgiShoppingBasket>

<tr bgcolor="#EEEECC">
```

```
<td colspan="2"><font size="-2"
face="helvetica, arial, sans-serif">
Tax and Shipping will be calculated during Checkout.
</font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>Subtotal</b></font></td>
<td><font size="-1" face="helvetica, arial, sans-serif">
<b>$<mgiGet name="mgiSBTotal"></b></font></td>
</tr>
</table>
```

If the default shopping basket does not fit with your site design, you may customize the shopping basket layout and the content of the shopping basket using the mgiShoppingBasket placeholders and variables. With a custom layout you can add font properties, cell colors, images, etc.

In a custom shopping basket, the code in the body of the mgiShoppingBasket tags is repeated for each item in the shopping basket and the placeholders are replaced with the item's specific information. After the mgiShoppingBasket tags, the shopping basket variables display information calculated from the current shopping basket contents. The custom shopping basket in this example displays this format:

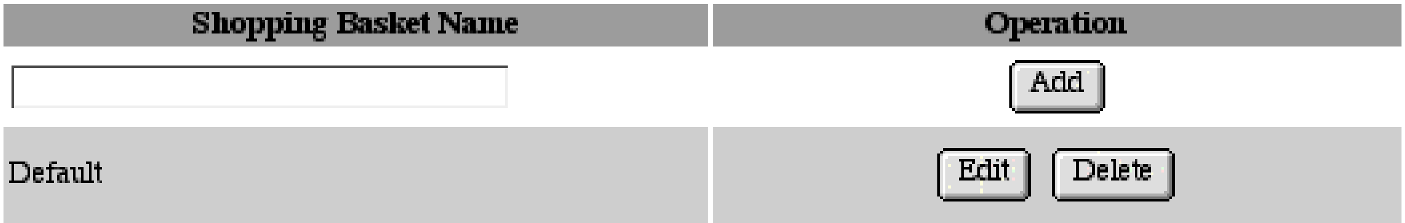


Remove	Qty	Description	Price
🗑	1	The New York Times Cook Book	$26.00
🗑	1	How to Cook Meat	$28.00
Tax and Shipping will be calculated during Checkout.		**Subtotal**	**$54.00**

Admin Mode

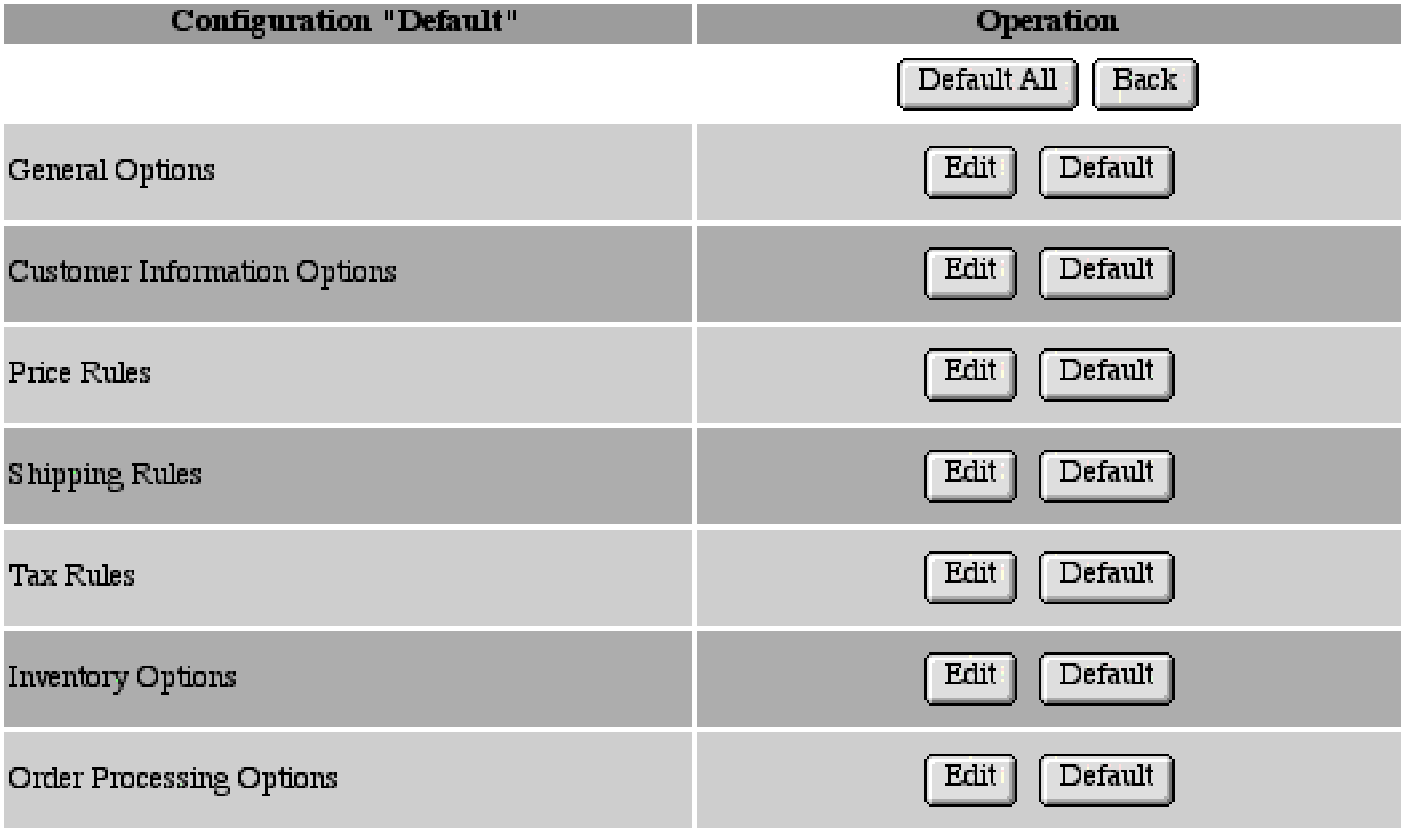
```
<mgiShoppingBasket mode="admin">
</mgiShoppingBasket>
```

In this example, the mgiShoppingBasket tag will display the admin interface.



Shopping Basket Name	Operation
	Add
Default	Edit Delete

In the admin mode of mgiShoppingBasket you create shopping basket configurations (handles) with options for customer checkout information, price rules, shipping rules, tax rules, inventory, and order processing.



Configuration "Default"	Operation
	Default All Back
General Options	Edit Default
Customer Information Options	Edit Default
Price Rules	Edit Default
Shipping Rules	Edit Default
Tax Rules	Edit Default
Inventory Options	Edit Default
Order Processing Options	Edit Default

Suggested Usage

- Shopping Basket

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiStop Tag

Tag Behavior

Use the mgiStop tag to stop all processing of MGI code on a page. All other code (HTML, etc.) on the page continues to be processed. (see also mgiAbort)

Tag Syntax

The mgiStop tag has no required parameters and no optional parameters. The tag form is:

```
<mgiStop>
```

Required Parameters:

- **None**.

Optional Parameters:

- **None**.

Example Usage and Output

```
<mgiIf lhs={mgiGet name="Products"} relationship="isEmpty">
<mgiStop>
</mgiIf>
```

In this example, an mgiStop tag is used in conjunction with a conditional comparison to stop the processing of shopping basket tags when the shopping basket is empty.

Suggested Usage

- Dynamic Forms
- Conditional Comparisons
- Site Development and Testing

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

The mgiString Tag

Tag Behavior

Use the mgiString tag to perform operations on text. (see also mgiInlineString).

Tag Syntax

The mgiString tag has twenty two modes. Each mode has different required and optional parameters. The modes of mgiString are:

- **urlEncode** - The urlEncode mode replaces URL reserved characters with encoded versions of those characters.
- **urlDecode** - The urlDecode mode replaces encoded URL characters with URL reserved characters.
- **htmlEncode** - The htmlEncode mode replaces HTML reserved characters with an encoded version of those characters.
- **htmlDecode** - The htmlDecode mode replaces encoded HTML characters with HTML reserved characters.
- **toUpper** - The toUpper mode changes the string to all capital letters.
- **toLower** - The toLower mode changes the string to all lowercase letters.
- **substring** - The substring mode extracts and displays a string of characters.
- **replace** - The replace mode finds a string and replaces it with another string.
- **insert** - The insert mode inserts a new string of text at a specified location in the string.
- **append** - The append mode adds a string of characters to the end of the string.
- **prepend** - The prepend mode adds a string of characters to the beginning of the string.
- **padleft** - The padLeft mode adds a specified number of characters to the beginning of the string.
- **padright** - The padRight mode adds a specified number of characters to the end of the string.
- **trimLeft** - The trimLeft mode removes whitespace from the beginning of a string.
- **trimRight** - The trimRight mode removes whitespace from the end of a string.
- **trim** - The trim function removes whitespace from the beginning and end of a string.

- **getLength** - The getLength mode displays the total length of the string.
- **split** - The split mode extracts values with a delimiter.
- **regularExpression** - The regularExpression mode searches a string for a regular expression pattern.
- **getWordCount** - The getWordCount mode displays the total number of words in a string.
- **getCharacterCount** - The getCharacterCount mode displays either the total number of characters in a string that are not carriage returns or linefeeds or the total number of times a specified character appears in a string..
- **noOperation** - The noOperation mode does not change the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

URLEncode Mode

The urlEncode mode of the mgiString tag has a beginning tag with one required parameter and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="urlEncode" quoteResult="Yes/No">
String to Encode
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**urlEncode**" mode, the mgiString tag replaces all URL reserved characters with an encoded version of the character (e.g., spaces in the URL are replaced with %20 characters).

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

URLDecode Mode

The urlDecode mode of the mgiString tag has a beginning tag with one required parameter and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="urlDecode" quoteResult="Yes/No">
String to Decode
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**urlDecode**" mode, the mgiString tag replaces all encoded characters with URL reserved characters (e.g., %20 characters are replaced with spaces).

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

HTMLEncode Mode

The htmlEncode mode of the mgiString tag has a beginning tag with one required parameter and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="htmlEncode" quoteResult="Yes/No">
String to Encode
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**htmlEncode**" mode, the mgiString tag replaces all HTML reserved characters with an encoded version of the character (e.g., quotation marks are replaced with &quot; characters).

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

HTMLDecode Mode

The htmlDecode mode of the mgiString tag has a beginning tag with one required parameter and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="htmlDecode" quoteResult="Yes/No">
String to Decode
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**htmlDecode**" mode, the mgiString tag replaces all encoded characters with HTML reserved characters (e.g., &quot; characters are replaced with quotation marks).

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

ToUpper Mode

The toUpper mode of the mgiString tag has a beginning tag with one required parameter and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="toUpper" quoteResult="Yes/No">
String to Change
</mgiString>
```

Required Parameter:

- **mode -** The mode is the function that the mgiString tag performs. In "**toUpper**" mode, the mgiString tag changes the string to all capital letters.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

ToLower Mode

The toLower mode of the mgiString tag has a beginning tag with one required parameter and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="toLower" quoteResult="Yes/No">
String to Change
</mgiString>
```

Required Parameter:

- **mode -** The mode is the function that the mgiString tag performs. In "**toLower**" mode, the mgiString tag changes the string to all lowercase letters.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

Substring Mode

The substring mode of the mgiString tag has a beginning tag with three required parameters and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="substring" beginningIndex="Number"
length="Number" quoteResult="Yes/No">
String to Search
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**substring**" mode, the mgiString tag extracts and displays a string of characters specified by the numeric location of the first character in the string and the length of the string.
- **beginningIndex -** The beginningIndex is an integer that indicates the numeric location of the first character to extract and display. The value of the beginningIndex value must be less than the total length of the string. For example, to extract the numeric month from the string "19991025", the beginningIndex would be "5".
- **length -** The length is an integer that indicates the length in characters of the string including the beginningIndex. For example, to extract the numeric month from the string "19991025", the beginningIndex would be "5" and the length would be "2".

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

Replace Mode

The replace mode of the mgiString tag has a beginning tag with three required parameters and two optional parameters, a body, and an ending tag. The tag form is:

```
<mgiString mode="replace" stringToReplace="String"
replacementString="String" quoteResult="Yes/No"
caseSensitive="Yes/No">
```

```
String to Search
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**replace**" mode, the mgiString tag finds a string and replaces it with another string.
- **stringToReplace** - The stringToReplace is the text that will be found and replaced by the replacement string. To find the escape characters carriage returns, line feeds, and tabs, use the appropriate Escaped String format.
- **replacementString** - The replacementString is the text that will replace any string found by the stringToReplace parameter. To find the escape characters carriage returns, line feeds, and tabs, use the appropriate Escaped String format.

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.
- **caseSensitive** - The caseSensitive parameter specifies whether the stringToReplace value is checked for case-sensitivity. If the caseSensitive parameter value is "**Yes**", the stringToReplace value is checked for case-sensitivity. If the caseSensitive parameter values is "**No**", the stringToReplace value is not checked for case-sensitivity. The default value is "No".

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

Insert Mode

The insert mode of the mgiString tag has a beginning tag with three required parameters and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="insert" stringToInsert="String"
index="Integer" quoteResult="Yes/No">
String to Search
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**insert**" mode, the mgiString tag inserts string of characters at a specified location in the string.
- **stringToInsert** - The stringToInsert is the text that is inserted at the location specified in the index parameter. To insert escape characters such as carriage returns, line feeds, and tabs, use the appropriate Escaped String format.
- **index** - The index is the integer of the character location where the string is inserted. For

example, in the string "abcdef", "c" is at the 3 index position.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

Append Mode

The append mode of the mgiString tag has a beginning tag with two required parameters and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="insert" stringToAppend="String"
quoteResult="Yes/No">
String
</mgiString>
```

Required Parameter:

- **mode -** The mode is the function that the mgiString tag performs. In "**append**" mode, the mgiString tag adds characters to the end of the string.
- **stringToAppend -** The stringToAppend is the text that is appended to the end of the string. To insert escape characters such as carriage returns, line feeds, and tabs, use the appropriate Escaped String format.

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

Prepend Mode

The prepend mode of the mgiString tag has a beginning tag with two required parameters and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="insert" stringToPrePend="String"
quoteResult="Yes/No">
String
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**prepend**" mode, the mgiString tag adds characters to the beginning of the string.
- **stringToPrepend** - The stringToPrepend is the text that is prepended to the beginning of the string. To insert escape characters such as carriage returns, line feeds, and tabs, use the appropriate Escaped String format.

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

PadLeft Mode

The padLeft mode of the mgiString tag has a beginning tag with one required parameter and two optional parameters, a body, and an ending tag. The tag form is:

```
<mgiString mode="padLeft" character="Character"
length="Integer">
String
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**padLeft**" mode, the mgiString tag adds a specified number of characters to the beginning of the string.

Optional Parameters:

- **character** - The character parameter specifies the type of character that is added to the beginning of the string. To add escape characters such as carriage returns, line feeds, and tabs, use the appropriate Escaped String format. The default character is a blank space.
- **length** - The length parameter specifies the number of specified characters to add to the beginning of the string. The default length is "1".

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

PadRight Mode

The padRight mode of the mgiString tag has a beginning tag with one required parameter and two optional parameters, a body, and an ending tag. The tag form is:

```
<mgiString mode="padRight" character="Character"
```

```
length="Integer">
String
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**padRight**" mode, the mgiString tag adds a specified number of characters to the end of the string.

Optional Parameters:

- **character** - The character parameter specifies the type of character that is added to the end of the string. To add escape characters such as carriage returns, line feeds, and tabs, use the appropriate Escaped String format. The default character is a blank space.
- **length** - The length parameter specifies the number of specified characters to add to the end of the string. The default length is "1".

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

TrimLeft Mode

The trimLeft mode of the mgiString tag has a beginning tag with one required parameter and two optional parameters, a body, and an ending tag. The tag form is:

```
<mgiString mode="trimLeft" quoteResult="Yes/No"
character="Character">
String to Trim
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**trimLeft**" mode, the mgiString tag removes all whitespace (including spaces, tabs and returns) from the beginning of a string.

Optional Parameters:

- **character** - The character to be trimmed.
- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

TrimRight Mode

The trimRight mode of the mgiString tag has a beginning tag with one required parameter and two optional parameters, a body, and an ending tag. The tag form is:

```
<mgiString mode="trimRight" quoteResult="Yes/No"
character="Character">
String to Trim
</mgiString>
```

Required Parameter:

- **mode -** The mode is the function that the mgiString tag performs. In "**trimRight**" mode, the mgiString tag removes all whitespace (including spaces, tabs and returns) from the end of a string.

Optional Parameters:

- **character -** The character to be trimmed.
- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

Trim Mode

The trim mode of the mgiString tag has a beginning tag with one required parameter and two optional parameters, a body, and an ending tag. The tag form is:

```
<mgiString mode="trim" quoteResult="Yes/No"
character="Character">
String to Trim
</mgiString>
```

Required Parameter:

- **mode -** The mode is the function that the mgiString tag performs. In "**trim**" mode, the mgiString tag removes all whitespace (including spaces, tabs and returns) from the beginning and end of a string.

Optional Parameters:

- **character -** The character to be trimmed.
- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

GetLength Mode

The getLength mode of the mgiString tag has a beginning tag with one required parameter and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="getLength" quoteResult="Yes/No">
String to Search
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**getLength**" mode, the mgiString tag displays the total length of the string.

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

Split Mode

The split mode of the mgiString tag has a beginning tag with three required parameters and two optional parameters, a body, and an ending tag. The tag form is:

```
<mgiString mode="split" delimiter="Character"
index="Integer" quoteResult="Yes/No"
caseSensitive="Yes/No">
String to Search
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**split**" mode, the mgiString tag extracts values with a delimiter.
- **delimiter** - The delimiter is the character that separates each part of the string. To specify escape characters such as carriage returns, line feeds, and tabs as delimiters use the appropriate Escaped String format.
- **index** - The index is the number of the value to extract. For example, in the following comma-delimited list, an index of "2" will extract the second value of "Smith".

  ```
  Bob,Smith,919-225-6329,919-225-6330
  ```

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

- **caseSensitive -** The caseSensitive parameter specifies whether the delimiter value is checked for case-sensitivity. If the caseSensitive parameter value is "**Yes**", the delimiter value is checked for case-sensitivity. If the caseSensitive parameter values is "**No**", the delimiter value is not checked for case-sensitivity. The default value is "No".

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

RegularExpression Mode

The regularExpression mode of the mgiString tag has a beginning tag with three required parameters and two optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="regularExpression" regularExpression="RE Value"
beginningIndex="Integer" quoteResult="Yes/No"
resultVariableName="Name">
String to Search
</mgiString>
```

Required Parameter:

- **mode -** The mode is the function that the mgiString tag performs. In "**regularExpression**" mode, the mgiString tag searches a string for a regular expression pattern.
- **regularExpression -** The regularExpression is the regular expression pattern to search for in the string. Create a regular expression pattern using regular expression symbols.
- **beginningIndex -** The beginningindex is an integer that indicates the numeric location in the string to begin searching. The value of the beginningIndex value must be less than the total length of the string. For example, a beginningIndex of "5" would start searching at the "e" in the string "abcdefghij".

Optional Parameters:

- **quoteResult -** The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.
- **resultVariableName -** The resultVariableName is the name prepended to result variables that contain information about the search. You may choose any name for the result variable prefix. To display result variables use the mgiGet tag. In the following list of available result variables, the resultVariableName prefix is "**resultvar**". For example, if your resultVariableName is "Email", then the page variable that contains the length of the full match would be named "Email_0_length". Result variables are created and available anywhere on the page after the mgiInlineString tag. The pound signs (#) represent the

regular expression subgroup number (from 1 to 9). Zero (0) is the main subgroup that is matched. (Use the mgiGet tag to display variable information.)

- **resultvar** - The full match returned by the regular expression.
- **resultvar_0_index** - The beginning index of the full match.
- **resultvar_0_length** - The length of the full match.
- **resultvar_#** - The # subgroup of the match. # can be 1 through 9.
- **resultvar_#_index** - The beginning index of the # subgroup match.
- **resultvar_#_length** - The length of the # subgroup match.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

GetWordCount Mode

The getWordCount mode of the mgiString tag has a beginning tag with one required parameter and one optional parameter, a body, and an ending tag. The tag form is:

```
<mgiString mode="getWordCount" quoteResult="Yes/No">
String to Search
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**getWordCount**" mode, the mgiString tag displays the total number of words in a string

Optional Parameters:

- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

GetCharacterCount Mode

The getCharacterCount mode of the mgiString tag has a beginning tag with one required parameter and three optional parameters, a body, and an ending tag. The tag form is:

```
<mgiString mode="getCharacterCount" character="Character"
quoteResult="Yes/No" caseSensitive="Yes/No">
String to Search
</mgiString>
```

Required Parameter:

- **mode** - The mode is the function that the mgiString tag performs. In "**getCharacterCount**" mode, the mgiString tag displays the total number of characters in a string that are not carriage returns or linefeeds.

Optional Parameters:

- **character** - The character or set of characters to be counted.
- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.
- **caseSensitive** - The caseSensitive parameter specifies whether the character value is checked for case-sensitivity. If the caseSensitive parameter value is "**Yes**", the character value is checked for case-sensitivity. If the caseSensitive parameter values is "**No**", the character value is not checked for case-sensitivity. The default value is "No".

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

NoOperation Mode

The noOperation mode of the mgiString tag has a beginning tag with no required parameters and two optional parameters, a body, and an ending tag. The tag form is:

```
<mgiString mode="noOperation" quoteResult="Yes/No">
String
</mgiString>
```

Required Parameter:

- **None**.

Optional Parameters:

- **mode** - The mode is the function that the mgiString tag performs. In "**noOperation**" mode, the mgiString tag does not change the string. The noOperation function can be used in conjunction with the quoteResult parameter to only enter quote marks around the string.
- **quoteResult** - The quoteResult parameter determines if the processed string is wrapped in quotes. If the quoteResult parameter value is "Yes", then quote marks are added to the beginning and end of the string. If the quoteResult parameter value is "No", then the quote marks are not added to the string.

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

Example Usage and Output

HTMLEncode Mode

```
<mgiString mode="htmlEncode">
<mgiPostArgument name="NewsStory">
</mgiString>
```

In this example, a news story from a form submission is HTML encoded for display on a page.

Split Mode

```
<mgiString mode="split" delimiter="," index="3">
blue,orange,green,purple,red
</mgiString>
```

In this example, the third value in a comma-delimited list of values is displayed. The third value from the list in this example is "green".

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

Suggested Usage

- Form Processing
- Dynamic Content

Return to the mgiString Mode Menu | Examples and Output | Suggested Usage

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiSwitch Tag

Tag Behavior

Use mgiSwitch tag to perform conditional comparisons to known cases. The syntax for the mgiSwitch tag is similar to that of a c-style switch statement. (see also mgiIf and mgiInlineIf)

Tag Syntax

The mgiSwitch tag has a beginning tag with one required parameter and one optional parameter, a body, and an ending tag. The mgiCase tag is used within the mgiSwitch tag to define comparison cases. The mgiCase tag has a beginning tag with no required parameters and one optional parameter, a body , and an ending tag (The mgiCase tag may have multiple beginning tags for any single ending tag). The tag forms are:

```
<mgiSwitch value="Value" caseSensitive="Yes/No">
  <mgiCase value="Comparison Value">
  Value if comparison to case(s) is true
  </mgiCase>
  <mgiCase>
  Default value if no comparison is true or
  if no comparison exists
  </mgiCase>
</mgiSwitch>
```

mgiSwitch

Required Parameters:

- **value -** The value in mgiSwitch is the string tested in the conditional comparison with the mgiCase values. In order to use special reserved characters such as tabs, backslashes, and ASCII characters, you must use the appropriate Escaped String format. The case value is not case-sensitive unless you specify a specific case using the Escaped String format or include the caseSensitive parameter.

Optional Parameters:

- **caseSensitive -** The caseSensitive parameter specifies whether the switch values and case values are checked for case-sensitivity. If the caseSensitive parameter value is "**Yes**", the switch and case values are checked for case-sensitivity. If the caseSensitive parameter values is "**No**", the switch and case values are not checked for case-sensitivity. The default value is "No".

mgiCase

Required Parameters:

- **None.**

Optional Parameters:

- **value** - The value in mgiCase is the string compared to the value in the mgiSwitch tag. The body of the first mgiCase value that is matched is displayed. Cases are tested until an ending mgiCase tag is found. If the value parameter is not included, then the case is considered to be the default result. The value parameter of the mgiCase tag is not preprocessed by MGI (i.e., MGI tags are not processed within the value parameter). In order to use special reserved characters such as tabs, backslashes, and ASCII characters, you must use the appropriate Escaped String format. The case value is not case-sensitive unless you specify a specific case using the Escaped String format or include the caseSensitive parameter in the mgiSwitch tag. All case values are constant, therefore you cannot dynamically populate case values by embedding MGI tags.

Example Usage and Output

The mgiSwitch equivalent to the c-style switch statement:	C-style switch statement:

`<mgiSwitch value="checkvalue">`
`    <mgiCase value="3">`
`        3`
`    <mgiCase value="2">`
`        2`
`    <mgiCase value="1">`
`        1`
`    </mgiCase>`
`    <mgiCase>`
`        `**`default`**
`    </mgiCase>`
`</mgiSwitch>`
	`c++ (pseudo-code):`
	
	`    switch(checkvalue)`
	`    {`
	`        case "3":`
	`            3;`
	`        case "2":`
	`            2;`
	`        case "1":`
	`            1;`
	`            break;`
	`        default:`
	`            "default";`
	`            break;`
	`    }`

```
Your letter grade is:
<mgiSwitch value={mgiGet name="Score"}>
```

```
    <mgiCase value="100">
    <mgiCase value="95">
    <mgiCase value="90">
        A
    </mgiCase>

    <mgiCase value="85">
    <mgiCase value="80">
        B
    </mgiCase>

    <mgiCase value="75">
    <mgiCase value="70">
        C
    </mgiCase>

    <mgiCase value="65">
        D
    </mgiCase>

    <mgiCase>
        F
    </mgiCase>

</mgiSwitch>
```

In this example, a calculated quiz score is compared to possible quiz scores in order to determine the student's letter grade. If the student's quiz score is "95", then their letter grade will appear as:

Your letter grade is: A

```
<mgiSwitch value={mgiRequest name="Username"}>
  <mgiCase value="Member">
    <mgiIncludeFile name="member.mgi">
  </mgiCase>
  <mgiCase value="Admin">
    <mgiIncludeFile name="admin.mgi">
  </mgiCase>
  <mgiCase>
    <mgiIncludeFile name="index.mgi">
  </mgiCase>

</mgiSwitch>
```

In this example, the content of a page is dynamically displayed based on a user's login information. If a user enters "Member" for their login, then the member.mgi page is included and displayed. If a user enters "Admin" for their login, then the admin.mgi page is included and displayed. If any other login is entered, then the index.mgi page is included and displayed.

Suggested Usage

- Conditional Comparisons
- Dynamic Content

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiTime Tag

Tag Behavior

Use the mgiTime tag to display the current time in hours and minutes of the server or offset from Greenwich Mean Time. (See also mgiDate).

Tag Syntax

The mgiTime tag has no required parameters and four optional parameters. The tag form is:

```
<mgiTime html="HTML" format="hour" text="Text" GMTOffset="Hours">
```

Required Parameters:

- **None.**

Optional Parameters:

- **format -** The format is the format of the time components. If the format parameter is not specified, the default display of the mgiTime tag is the server's time format. You may include multiple format parameters in one mgiTime tag. Valid formats for the mgiTime tag are:
  - **Hour -** Displays the numeric hour between 1 and 12.
  - **PaddedHour -** Displays the numeric hour between 01 and 12.
  - **MilitaryHour -** Displays the numeric hour between 00 and 23.
  - **Minute -** Displays the minutes of an hour between 00 and 59.
  - **Second -** Displays the seconds of a minute between 00 and 59.
  - **AmPm -** Displays an "am" or "pm" label.
- **gmtOffset -** The gmtOffset is the whole or half number of hours the time is offset from Greenwich Mean Time preceeded by a positive (+) or negative (-) sign (e.g., "+5" or "-3.5"). Only half hours may be entered. Other tenths of an hour may not be entered. When the GMTOffset parameter is not present, the local time of the server is displayed. A global GMTOffset can be enabled in the region preferences for your domain. If you do not have access to the region preferences for your domain, contact the server administrator. Including a gmtOffset parameter overrides the gmtOffset in the region preferences.
- **html -** The html is the HTML code and text that is decoded and displayed relative to the specified date formats. Any HTML in the value of the html parameter should already be encoded (e.g., quotation marks should be entered as "&quot;"). You may include multiple html parameters in one mgiTime tag.

- **text -** The text is the string of characters that is decoded and displayed relative to the specified time formats. Text in the value of the text parameter is not decoded prior to display. You may include multiple text parameters in one mgiTime tag.

Example Usage and Output

```
<mgiTime>
```

If the current server time is 8:45 in the evening, the default mgiTime tag might display as:

8:45 pm

```
<mgiTime format="paddedHour" text=" " format="amPm">
```

If the current server time is 6:13 in the morning, the mgiTime tags in this example would display as:

06 am

```
<mgiTime html="Local Time &lt;b&gt;" format="hour"
text=":" format="minute" text=" "
format="amPm" html="&lt;/b&gt;" GMTOffset="-8">
```

If the current GMT time is 9:15 pm, but you want the local time for Los Angeles, California, then the GMTOffset indicated in this example would display as:

Local Time **1:15 pm**

Suggested Usage

- Server Side Includes
- Guestbook
- Form Processing
- Database Submission

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiToken Tag

Tag Behavior

Use the mgiToken tags to append a token to URLs (HREFs and ACTIONs) between the beginning and ending tags. A token number can be used as a visitor's unique shopping basket identification. A token is composed of the visitor's IP number and time interval so that each token is absolutely unique.

Tag Syntax

The mgiToken tag has a beginning tag with no required parameters and three optional parameters, a body, and an ending tag. The tag form is:

```
<mgiToken mode="Mode" pathArgumentName="Name"
pathArgumentValue="Value">
URLs to be tokenized
</mgiToken>
```

Required Parameters:

- **None**.

Optional Parameters:

- **mode -** The mode is the function that the mgiToken tag performs. In "**addTokenOnly**" mode, the mgiToken tag appends the mgiToken path argument and unique token value to URLs. In "**addTokenWithPathArguments**" mode, the mgiToken tag appends the token and all path arguments to URLs. In "**addPathArgumentsOnly**" mode, the mgiToken tag appends all path arguments to URLs. The default value is "addTokenOnly".
- **pathArgumentName -** The pathArgumentName is the name of a path argument that is always appended to tokenized URLs. If this parameter is not followed by a pathArgumentValue parameter in the tag, it is ignored.
- **pathArgumentValue -** The pathArgumentValue is the value of the specified path argument that is always appended to tokenized URLs. If this parameter is not preceded by a pathArgumentName parameter in the tag, it is ignored.

Technical Notes

- **Token Exceptions** - The mgiToken tag does not append a token to "mailto" HREFs or URLs with an ending slash (/). URLs with an ending slash usually indicate a link to an external site.

Example Usage and Output

```
<mgiToken>
<A HREF="purchase.html">Purchase Our Products!</A>
</mgiToken>
```

When this example token tag is processed through an MGI server, the HREF is appended with a unique token number such as:

```
<A HREF="purchase.html?mgiToken=CD982415B2F65040">
Purchase Our Products!</A>
```

Suggested Usage

- Shopping Baskets
- Affiliate Systems

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

The mgiValidateData Tag

Tag Behavior

Use the mgiValidateData tag to check text for specified formats.

Tag Syntax

The mgiValidateData tag has four types. Each type has different required and optional parameters. The mgiValidateData types are:

- **string -** Validates text values.
- **integer -** Validates integer values.
- **real -** Validates real number values.
- **date -** Validates date values

String Type

The string type of mgiValidateData has one required parameter and four optional parameters. The tag form is:

```
<mgiValidateData data="Text" type="string" failureURL="URL"
format="Format" regularExpression="RE Value">
```

Required Parameters:

- **data -** The data is the text to validate.

Optional Parameters:

- **type -** The type is the type of information that is validated. The "**string**" type validates the format of a text string. "String" is the default type of the mgiValidateData tag. If the data does not match the type given or the data value is empty, then an automatic failure occurs.
- **failureURL -** The failureURL is the page where failed validations are redirected. The default behavior is to display the words "invalid data" if the data does not meet the specified criteria and to display the data you are validating if the data does meet the specified criteria.
- **format -** The format is the pattern that is validated in the data. The "**url**" format compares the data to a URL pattern. The "**emailAddress**" format compares the data to an email address pattern. The "**regularExpression** format compares the data with the expression

provided in the regularExpression parameter. There is no default value.

- **regularExpression** - The regularExpression is the regular expression pattern validated in the data. Create a regular expression pattern using regular expression symbols.
- **display** - Whether or not the tag should display the result or the "invalid data" message after being processed. Valid values are "yes" and "no". The default value is "yes". *(Note: This parameter, if set to "no", should be used in conjunction with the **failureURL*** parameter to catch invalid data.)

Integer Type

The integer type of mgiValidateData has two required parameter and four optional parameters. The tag form is:

```
<mgiValidateData data="Number" type="integer" failureURL="URL"
upperBound="Integer" lowerBound="Integer">
```

Required Parameters:

- **data** - The data is the number to validate.
- **type** - The type is the type of information that is validated. The "**integer**" type validates the format of a whole number.

Optional Parameters:

- **failureURL** - The failureURL is the page where failed validations are redirected. The default behavior is to display the words "invalid data" if the data does not meet the specified criteria and to display the data you are validating if the data does meet the specified criteria.
- **upperBound** - The upperBound is the maximum value of the data.
- **lowerBound** - The lowerBound is the minimum value of the data.
- **display** - Whether or not the tag should display the result or the "invalid data" message after being processed. Valid values are "yes" and "no". The default value is "yes". *(Note: This parameter, if set to "no", should be used in conjunction with the **failureURL*** parameter to catch invalid data.)

Real Type

The real type of mgiValidateData has two required parameter and four optional parameters. The tag form is:

```
<mgiValidateData data="Number" type="real" failureURL="URL"
upperBound="Number" lowerBound="Number">
```

Required Parameters:

- **data** - The data is the number to validate.
- **type** - The type is the type of information that is validated. The "**real**" type validates the format of a real number (positive or negative with decimals).

Optional Parameters:

- **failureURL** - The failureURL is the page where failed validations are redirected. The default behavior is to display the words "invalid data" if the data does not meet the specified criteria and to display the data you are validating if the data does meet the specified criteria.
- **upperBound** - The upperBound is the maximum value of the data.
- **lowerBound** - The lowerBound is the minimum value of the data.
- **display** - Whether or not the tag should display the result or the "invalid data" message after being processed. Valid values are "yes" and "no". The default value is "yes". *(Note: This parameter, if set to "no", should be used in conjunction with the* ***failureURL*** parameter to catch invalid data.)

Date Type

The date type of mgiValidateData has two required parameter and eight optional parameters. The tag form is:

```
<mgiValidateData data="Date" type="date" failureURL="URL"
upperBound="Date" lowerBound="Date" upperSpan="Integer"
lowerSpan="Integer"
dateFormat="Format">
```

Required Parameters:

- **data** - The data is the date to validate.
- **type** - The type is the type of information that is validated. The "**date**" type validates the format of a date.

Optional Parameters:

- **failureURL** - The failureURL is the page where failed validations are redirected. The default behavior is to display the words "invalid data" if the data does not meet the specified criteria and to display the data you are validating if the data does meet the specified criteria.
- **upperBound** - The upperBound is the maximum date of the data. The upperBound date must be in the form mm-dd-yyyy unless specified otherwise in the dateFormat parameter.
- **lowerBound** - The lowerBound is the minimum date of the data. The lowerBound date must be in the form mm-dd-yyyy unless specified otherwise in the dateFormat parameter.

- **upperSpan -** The upperSpan is the number of days after the current date when the data is valid. The upperSpan value must be a positive integer.
- **lowerSpan -** The lowerSpan is the number of days before the current date when the data is valid. The lowerSpan value must be a positive integer.
- **dateFormat -** The dateFormat is the required format of the data. Valid values of the dateFormat parameter are "mm-dd-yyyy" and "dd-mm-yyyy". The default value is "mm-dd-yyyy".
- **delimiter -** The delimiter that must separate the month, day, and year values. If no delimiter is specified, dashes('-'), back slashes('/'), and periods('.') will be accepted as valid delimiters.
- **display -** Whether or not the tag should display the result or the "invalid data" message after being processed. Valid values are "yes" and "no". The default value is "yes". *(Note: This parameter, if set to "no", should be used in conjunction with the* ***failureURL*** parameter to catch invalid data.)

Example Usage and Output

String Type

```
<mgiValidateData data={mgiPostArgument name="Email"}
type="string" failureURL="emailerror.mgi" format="emailAddress">
```

In this example, an email address from a form submission is verified. If the "Email" field is left empty or if the email address is not formatted properly, then the visitor is redirected to the emailerror.mgi page.

Integer Type

```
<mgiValidateData data={mgiPostArgument name="Quantity"}
type="integer" failureURL="quantityerror.mgi" lowerBound="1">
```

In this example, the quantity of a product is verified. The quantity must be a whole number greater than 1, otherwise the visitor is redirected to the quantityerror.mgi page.

Real Type

```
<mgiValidateData data={mgiPostArgument name="Price"} type="real"
failureURL="searcherror.mgi" upperBound="1000000.00">
```

In this example, search criteria for the price of new homes is verified. The entered price must be a real number less than 1 million dollars, otherwise the visitor is redirected to the searcherror.mgi page.

Date Type

```
<mgiValidateData data={mgiGet name="Appointment Date"}
type="date" failureURL="daterror.mgi" upperSpan="5">
```

In this example, an appointment date is verified. The appointment date must be anywhere from the current date to five days after the current date, otherwise the visitor is redirected to the dateerror.mgi page.

Suggested Usage

- Form Processing
- Identification Verification

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

MGI Escaped Strings

Syntax

In order to use special reserved characters, you must use the escaped string format. Escaped strings use the backslash ("\") to denote special characters that normally cannot be entered. The supported escaped characters are:

- \\ - Inserts the backslash "\" character
- \r - Inserts a carriage return
- \n - Inserts a new line (linefeed)
- \t - Inserts a tab
- \v - Inserts a vertical tab
- \xyz - Inserts an ASCII value "xyz" is the character's three digit ASCII character value. Ex: \065 is 'A'.

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

MGI Regular Expressions

Syntax

MGI supports UNIX extended regular expression syntax. The syntax for regular expression symbols are:

Symbol	Function
\	Marks the next character as a special escaped character
^	Matches/anchors the beginning of line
$	Matches/anchors the end of line
?	Matches the preceding pattern or group zero or one times
*	Matches the preceding pattern or group zero or more times
+	Matches the preceding pattern or group one or more times
.	Matches any single character except \r and \n
{x,y}	Matches the preceding pattern at least x times, no more than y times
b{7}	Matches exactly 7 characters
b{7,}	Matches at least 7 characters
b{7,9}	Matches at least 7 characters but no more than 9 characters
(expression)	Brackets or tags an expression as a group (there may be up to 9 groups)
abc|xyz	Matches abc OR xyz
(abc|xyz)	Matches abc OR xyz
[xyz]	A character set. Matches any characters between brackets
[^xyz]	A negative character set. Matches any characters NOT between brackets
\f	Matches a form-feed character
\n	Matches a linefeed character
\r	Matches a carriage return character
\t	Matches a tab character
\v	Matches a vertical tab character
\c	Matches a character. Equivalent to [A-Za-z]
\C	Matches a noncharacter. Equivalent to [^A-Za-z]
\d	Matches a digit. Equivalent to [0-9]
\D	Matches a nondigit. Equivalent to [^0-9]
\s	Matches any white space including space, tab, formfeed, etc but not newline. Equivalent to [\f\t\v]
\S	Matches any nonwhite space character but not newline. Equivalent to [^ \f\t\v]
\w	Matches any word character including underscore. Equivalent to [A-Za-z0-9 _]
\W	Matches any nonword character. Equivalent to [^A-Za-z0-9 _]

Note - ^ refers to the character '^' NOT Control Key + value

Special Character Requirements:

To search for a `'['` place it inside a character set or escape it with a backslash

To search for a `']'` it must be the first character inside the character set or escaped with a backslash

To search for `.,+*?$(){}` they must be inside a character set or escaped with a backslash

To search for `'^'` it must not be the first character on a line or the first character in a set or it must be escaped with a backslash

To search for `'-'` in a character set it must be first or last character in the set

Example Usage and Output

`m.n` matches "man", "men", "min" but not "moon"

`Te?st` matches "tst", "test" BUT NOT "teest", "teeest" etc.

`Te+st` matches "test", "teest", "teeeest" etc. BUT NOT "tst"

`Te*st` matches "test", "teest", "teeeest" etc. AND "tst"

`[aeiou]` matches every lowercase vowel

`[,.?]` matches a literal ",", "." or "?"

`[0-9a-z]` matches any digit, or lowercase letter

`[^0-9]` matches any character except a digit (^ means NOT the following)

You may search for an expression A or B as follows: `(John|Jane)`

[Understanding MGI Menu] [Using MGI Menu] [Referencing MGI Menu]

[MGI Guides Main Menu] [User Guide Main Menu]

PDF Version History

The PDF version of the MGI Admin Guide and MGI User Guide is dated with the month, day and year of release. The original PDF document was released on 23 August 2001. The following is a list of sections that have been added or updated since the original release.

If you have a printed version of the PDF guide, you may simply print and replace updated sections or print and add new sections.

Date	Section	Updates or Additions
07-10-02	User | Referencing | mgiRequest	Added queryString data type.
	User | Referencing | mgiSendMail	Added headerName and headerValue parameters
06-25-02	User | Referencing | Index	Added mgiDynamicList, mgiIncrement and mgiInfoDumper links.
	User | Referencing | mgiDynamicPopup	Updated documentation of database source.
	User | Referencing | mgiDynamicList	Added new tag description.
	User | Referencing | mgiEditDatabase	Updated technical notes.
	User | Referencing | mgiEncryptURL	Updated documentation of url parameter.
	User | Referencing | mgiIncrement	Added new tag description.
	User | Referencing | mgiInfoDumper	Added tag description.
	User | Referencing | mgiJulianDay	Updated with new AmPm parameter. Updated with new daySuffix format.
	User | Referencing | mgiRequest	Updated with new pageID data type.
	User | Referencing | mgiString	Updated with new parameters.
	User | Using | Counters	Updated example download link.
09-28-01	User | Referencing | Index	Added mgiRollOver link.
	User | Referencing | mgiCreditCard	Updated note for spaces/hyphens.
	User | Referencing | mgiIncludeHTTP	Updated note for secure URLs.
	User | Referencing | mgiRollOver	Added new tag description.
	User | Referencing | mgiSendMail	Updated replyFrom address requirements.
	PDF Version History	Added PDF Version History
08-23-01	All	Original release of PDF.