SoftSlate Commerce

User Guide

1.0.14 Edition

SoftSlate, Inc.

Copyright © 2003-2006 SoftSlate, Inc.

SoftSlate Commerce: User Guide

SoftSlate, Inc.
Copyright © 2003-2006 SoftSlate, Inc.

Table of Contents

List of Tables

List of Examples

Part I. Guide for Administrators

Table of Contents

Chapter 1. Installing SoftSlate Commerce

Overview and System Requirements

The process of installing SoftSlate Commerce consists of first deploying the application to a compatible application server, and then running the Installer tool to configure key settings and initialize the data objects in the database.

The following is a list of system requirements for the server running SoftSlate Commerce. Please make sure you have downloaded and installed the following pieces of software before installing SoftSlate Commerce.

System Requirements for SoftSlate Commerce

- **Java Compiler and Runtime Environment.**

  SoftSlate Commerce has been tested with the Sun JDK, versions 1.4.0, 1.4.2, and 5.0, and may work on other versions. The Sun JDK may be downloaded free of charge from http://java.sun.com/j2se/downloads.html.

- **Java Application Server.**

  SoftSlate Commerce has been tested with Apache Tomcat 4.0, 4.1, 5.0, and 5.5, and may work on other versions. Apache Tomcat may be downloaded free of charge from http://jakarta.apache.org/tomcat.

  It has also been tested with BEA WebLogic 8.1, which is available from the BEA Web site: http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/server.

  SoftSlate Commerce may work on other compliant J2EE Web Containers.

- **Database.**

  SoftSlate Commerce has been tested with MySQL 3.23 and 4.0, and may work on other versions. MySQL may be downloaded free of charge from http://www.mysql.com/downloads.

  It has also been tested with Microsoft SQL Server 2000. Information is available on the Microsoft Web site: http://www.microsoft.com/sql.

  Note that the database used by SoftSlate Commerce need not be installed on the same server that runs the application itself. During the Installer, you can specify the host name of the database server.

- **Operating System.**

  SoftSlate Commerce has been tested on Windows XP and Red Hat Linux but should work well on any operating system supported by Java.

Download SoftSlate Commerce

You should have received a URL, and possibly a user name and password to use to download SoftSlate

Commerce. If you don't know where to find the downloads, contact sales or support.

There are three file formats available for you to download. Choose any one of these three files to download; they contain the exact same contents.

- **`softslate-standard-x.x.x.war`.**

  Contains the application in .war (Web application archive) format. Tomcat and other Java application servers can unpackage and deploy .war files automatically. Note: for technical reasons, you cannot use a .war file to install SoftSlate Commerce on BEA WebLogic.

- **`softslate-standard-x.x.x.tar.gz`.**

  Contains the application in tar.gz format. Most commonly used when installing on a Unix or Linux platform.

- **`softslate-standard-x.x.x.zip`.**

  Contains the application in ZIP format. Most commonly used when installing on a Windows platform.

Create an Empty Database

Create an empty database for SoftSlate Commerce to use in MySQL or Microsoft SQL Server 2000.

Note

It is possible to install SoftSlate Commerce in an existing database. All of the database tables used by SoftSlate Commerce begin with the prefix `npc`. If your existing database has no tables that would conflict, you can use it. This may be advantageous in situations where you need to integrate SoftSlate Commerce with other applications.

Example 1.1. Creating an Empty Database in MySQL

1. To, create an empty database in MySQL, log into the MySQL monitor:

```
[root@server root]# mysql -u root -p
Enter password:
```

2. Run the **create database** command. (You may name the database whatever you wish. You will be asked for the name by the Installer tool.)

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 978 to server version: 3.23.58

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```
mysql>create database softslate;
```

Example 1.2. Creating an Empty Database in Microsoft SQL Server

You can use the following script to create a database in Microsoft's Query Analyzer application (or a modified form of this script):

```
USE master
GO
create database softslate
on
(
        name='DATA_softslate',
        filename='C:\Program Files\Microsoft SQL Server\MSSQL\Data\DATA_softslate.
        size=2,
        filegrowth=10%
)
log on
(
        name='XLOG_softslate',
        filename='C:\Program Files\Microsoft SQL Server\MSSQL\Data\XLOG_softslate.
        size=1,
        filegrowth=10%
)
GO
```

Important

It's often wise to create a separate database user for each application handled by your database. Please refer to the documentation for MySQL or Microsoft SQL Server 2000 for instructions on how to do this.

Upload and Deploy SoftSlate Commerce

The following example describes in detail how to deploy SoftSlate Commerce on a Linux server running Tomcat. The steps are largely the same for Windows. For other Java application servers, please refer to their documentation for how to install a Web application on their platform.

Example 1.3. Deploying SoftSlate Commerce with Tomcat on Linux

You may either use the `softslate-standard-x.x.x.tar.gz` file or the `softslate-standard-x.x.x.war` file to install SoftSlate Commerce on Linux. (For Windows, consider using the `.zip` file.) In the first case, you will unpack the application manually. In the second case, you can place the .war file in Tomcat's `webapps` directory and it will unpack the application.

Note

For technical reasons, SoftSlate Commerce will not install correctly on BEA WebLogic when using the `.war` file.

Deploying Using the `.tar.gz` File.

1. If you are using the `.tar.gz` file, change into the directory you wish to install SoftSlate Commerce in, and copy the `.tar.gz` file there. This must be a directory served by Tomcat. For this example, we'll assume we are installing the application in the `/usr/local/jakarta-tomcat/webapps` directory.

```
[root@server root]# cd /usr/local/jakarta-tomcat/webapps
[root@server webapps]# cp /root/softslate-standard-1.0.5.tar.gz .
```

2. Untar the `.tar.gz` file. A directory named `softslate` should appear with the contents of the application.

```
[root@server webapps]# tar -zxf softslate-standard-1.0.5.tar.gz
```

3. For the Installer tool to run successfully, the owner of the Tomcat process must have *writable* access to the `/WEB-INF/conf/keys` directory and the `/WEB-INF/classes/appSettings.properties` file. If necessary, use **chown** and **chmod** to give the Tomcat user permission.

```
[root@server webapps]# chown <anyuser>.tomcat softslate/WEB-INF/conf/keys
[root@server webapps]# chown <anyuser>.tomcat softslate/WEB-INF/classes/appSett
[root@server webapps]# chmod g+w softslate/WEB-INF/conf/keys
[root@server webapps]# chmod g+w softslate/WEB-INF/classes/appSettings.properti
```

4. Start the SoftSlate Commerce Web Application. With Tomcat you can use the Manager application to run the start command. For example, if you deployed SoftSlate Commerce under the `/softslate` context and you have Tomcat running on port 8080, you would use the following URL to start the application:

```
http://<yourhostname>:8080/manager/start?path=/softslate
```

Deploying Using the `.war` File.

1. If you are using the `.war` file, first *change the name of the .war file to the name you'd like the application to run under*. For example if you'd like the application to use the context `/softslate`, change the name of the `.war` file to `softslate.war`.

```
[root@server root]# mv softslate-standard-1.0.5.war softslate.war
```

2. Change into the directory you wish to install SoftSlate Commerce in, and copy the `.war` file there. This must be a directory served by Tomcat and configured to automatically deploy `.war` files. For this example, we'll assume we are installing the application in the `/usr/local/jakarta-tomcat/webapps` directory.

```
[root@server root]# cd /usr/local/jakarta-tomcat/webapps
[root@server webapps]# cp /root/softslate.war .
```

   Tomcat should automatically unpack and deploy the application, and a directory named `softslate` should appear. (The directory name will correspond to whatever name you gave to the `.war` file.)

   Since Tomcat has deployed the application, there is no need to adjust the permissions or ownership.

Run the SoftSlate Commerce Installer

1. Navigate to the Installer tool using a browser. Visit the `installer` directory under the SoftSlate Commerce installation you've deployed. For example, if you deployed SoftSlate Commerce under the `/softslate` context and you have Tomcat running on port 8080, you would go to this URL: `http://<yourhostname>:8080/softslate/installer`

2. Follow the instructions given by the Installer tool to finish installing SoftSlate Commerce. The Installer tool will ask you how to connect to the database and will ask you to provide key configuration settings. It will then attempt to connect to the database, install all the database objects, and initialize the settings.

3. If you have installed using the `.tar.gz` file, following a successful installation you may wish to change the permissions of the `/WEB-INF/conf/keys` directory and the `/WEB-INF/classes/appSettings.properties` file back for better security.

```
[root@server root]# cd /user/local/jakarta-tomcat/webapps
[root@server webapps]# chown <restricteduser>.<restricteduser> softslate/WEB-IN
[root@server webapps]# chown <restricteduser>.<restricteduser>
softslate/WEB-INF/classes/appSettings.properties
[root@server webapps]# chmod g-w softslate/WEB-INF/conf/keys
[root@server webapps]# chmod g-w softslate/WEB-INF/classes/appSettings.properti
```

4. After a successful installation, please consider visiting the Getting Started chapter to begin working with SoftSlate Commerce.

Installation Troubleshooting

- **After deploying, the Installer tool does not come up in my browser.**

  There are many reasons why deploying an application might fail. Please refer to the instructions and documentation for the Java application server you are using to troubleshoot any issues. If you are using Tomcat, the Tomcat log file, which is usually located at `<tomcat_home>/logs/catalina.out` or `<tomcat_home>/logs/stdout.log` can provide much-needed clues.

- **The Installer tool fails to connect to the database.**

  If the Installer tool fails to connect to the database, double-check the database name, user name, and password. If you are using MySQL, you may need to associate the database user with your server's host name for the connection to work. (You can refer to the section that describes how to do this under the Installation Procedure for Ensim Servers section.)

- **The Installer tool finished successfully, but I cannot log into the Administrator.**

  It is possible to successfully run the installer after making an error when filling out the "Administrator URL" setting. This is a problem, because the Administrator itself is typically what you would use to make changes to the URL settings.

  If logging into the Administrator takes you to the wrong place and you must change the "Administrator URL" setting after finishing the Installer, you must first unlock the Installer tool. Do this by editing the `WEB-INF/classes/appSettings.properties` file located under the SoftSlate Commerce installation directory. Look for the line beginning "lockInstaller", and delete that line.

  Next, navigate back to the Installer tool, and click the link to "Switch to Advanced Installer". From the Advanced Installer tool, click the link to "Store Settings" on the left side menu. Fill in the correct Administrator URL (along with all the other required fields on that screen), and click "Continue". Test logging into the Administrator; the URL should be updated. If successful, please go back to the Advanced Installer tool and click the link to "Lock Installer", and click "Continue" to lock the Installer tool again.

- **The Installer tool finished successfully, but I need to run it a second time.**

  If for whatever reason you need to rerun the Installer tool, you have to do two things: unlock the Installer if it has been locked, and empty the database you are using so the data objects can be reinserted without an error.

  To unlock the Installer tool, edit the `WEB-INF/classes/appSettings.properties` file located under the SoftSlate Commerce installation directory. Look for the line beginning "lockInstaller", and delete that line.

  The easiest way to empty the database is often to simply delete it and recreate it. For example, in MySQL, you could run the following commands to create a new, empty database:

  ```
  mysql>drop database softslate;
  mysql>create database softslate;
  ```

- **Installing SoftSlate Commerce manually, without the Installer tool.**

  It is possible to install SoftSlate Commerce manually without using the Installer tool.

Note

If you need to use two-way encryption, the only way to create the two-way encryption key is through the Installer tool. You cannot install manually if you plan to use two-way encryption to store customer passwords, administrator passwords, or payment information. For more information, refer to the section on Security Settings.

To configure the database settings manually, simply edit the `WEB-INF/classes/appSettings.properties` file located under the SoftSlate Commerce installation directory. Add the database name, username, password, and other settings.

To initialize the database objects, run each of the `create*.sql` files followed by the `insert.sql` files under the `WEB-INF/classes/resources` directory. These files create and insert the database objects needed by SoftSlate Commerce.

To initialize key settings required by the application, you must update the following records in the `npcSetting` database table. This example is for a MySQL database.

```
mysql>use softslate;
mysql>update npcSetting set mediumValue='<base customer URL>' where code = 'cust
mysql>update npcSetting set mediumValue='<base secure customer URL>' where code
mysql>update npcSetting set mediumValue='<base administrator URL>' where code =
```

To create and initial administrator login, you must insert records into the `npcAdministrator` and `npcAdministratorRole` database tables, placing the administrator in the `superuser` role.

Using a Web Server To Handle Non-Java Requests

A common practice is to use a Java application server such as Tomcat in conjunction with an HTTP Web server such as Apache HTTP Server. You can set up application so that Tomcat handles only those requests that involve Java processing, while Apache handles all images, stylesheets, and HTML files.

If you are doing this, you need to map the following URL patterns to the Tomcat server. You would put these mappings into Apache's configuration file, `httpd.conf`, or the Apache-Tomcat Connector's configuration file, `workers2.properties`. Please refer to the Apache-Tomcat Connector's documentation for complete instructions. Information is available at http://jakarta.apache.org/tomcat/connectors-doc.

These are the patterns that SoftSlate Commerce needs to use for requests involving Java processing.

SoftSlate Commerce's URL Mappings

- `/*.jsp`
- `/*.do`
- `/softslate/do/*`

If you are using a Web server to handle non-Java requests, another critical configuration step is to prevent access to the `WEB-INF` directory. That directory holds a number of files that, if exposed to outside users, would compromise security.

Warning

If you using a Web server to handle non-Java requests with SoftSlate Commerce, *it is critical that you prevent access to SoftSlate Commerce's `WEB-INF` directory.*

Here is an example of a directive for Apache to prevent access to the `WEB-INF` directory. This example assumes that you have installed SoftSlate Commerce in a directory named `/var/www/html`. (Replace that with the corresponding directory on your server.) You would place this directive in the appropriate place in Apache's configuration file, `httpd.conf`.

```
<Directory /var/www/html/softslate/WEB-INF/>
  AllowOverride None
  Deny from all
</Directory>
```

Installation Procedure for Ensim Servers

The following steps are provided if you need to install SoftSlate Commerce on a server running Ensim. These instructions were tested specifically on a server running Ensim Pro for Linux, version 4.0. It may work for other versions as well.

Example 1.4. Installing SoftSlate Commerce on an Ensim Server

1. **Provision the Tomcat Service for the Site You're Setting Up.**

   ### Note

   You must set up the site `.policy.custom` file *before* provisioning Tomcat for the site. Otherwise, the site's `.policy.custom` file will not take effect.

   a. Log on to the server through ssh.

   b. Change to the Tomcat site policies configuration directory:

   ```
   [root@server root]# cd /var/tomcat4/conf/sites.policies.d
   ```

   c. Create a file in this directory named `site15.policy.custom`, whose contents are the following. (Replace "site15" with the Site ID of the site you are setting up.)

```
permission java.security.AllPermission;
```

d. Log in to Ensim's Appliance Administrator control panel.

e. Click on the "Site Manager" link and find the site you are installing SoftSlate Commerce on.

f. Click the pencil icon next to the site name.

g. The site must have Tomcat 4 provisioned for it. If it is not already checked, mark the checkbox next to Tomcat 4, which is under the "Web Server" heading.

h. Tomcat 4 will not be assigned to the site if its "Security Level" is set to "High". If the security level is set to "High", you must change it to "3.1 Compatibility".

2. **Create an Empty Database for SoftSlate Commerce.**

a. Log in to Ensim's Appliance Administrator control panel.

b. Click on the "Site Manager" link and find the site you are installing SoftSlate Commerce on.

c. Click the pencil icon next to the site name.

d. The site must have at least one available MySQL database. Under the "Database Server" heading, make sure the checkbox next to "MySQL" is checked and that the site has at least one database.

e. Click "Save" to record your changes.

f. Navigate back to the "Site Manager" list.

g. Click on the name of the site you are setting up, and click yes to become the Site Administrator.

h. From the Site Administrator control panel, click on Services.

i. Click on the pencil icon next to MySQL.

j. Click on the "Create Database" link.

k. Create a database with "softslate" as the suffix. E.g. `www_sitename_com_-_softslate`.

l. If you have made a change to the site's database password, you may need to log back into Ensim's Appliance Administrator control panel and restart MySQL from the Services menu for the changes to take effect.

3. **In MySQL, Associate the Database User with the Proper Host.**

Note

On some servers, this step may not be necessary. If SoftSlate Commerce's Installer is having trouble making a connection to the database, you may need to perform these steps. MySQL must have an entry in the `user` table that associates the user of the site's database with the host name of your server. In some cases, this is taken care of when you set up a site.

a. Log on to the server through ssh and look up the hostname of your server by running the **hostname** command:

```
[root@server root]# hostname
server.domainname.com
[root@server root]#
```

b. Log in to Ensim's Appliance Administrator control panel.

c. Navigate to "Services" and click on the pencil icon next to MySQL

d. Click the link to "MySQL Administration Tool"

e. Log into MySQL with the root username and password.

f. In phpMyAdmin, select mysql under the "Databases..." menu.

g. Click on the link to the `user` table.

h. Click on the "Insert" tab to add a record to the `user` table.

i. For Host use the hostname of your server from step 1.

j. For User use the MySQL user corresponding to the site you are setting up. (Click on the Browse tab if you need to find it.)

k. For Password, select "PASSWORD" in the drop down menu under the Function column. Enter in the user's password for database access in the Password field.

l. Leave all the radio buttons marked as "N" for No.

m. Click the Go button at the bottom of the page to add the record.

n. Navigate back to the Ensim control panel home, and go back to the "Services" menu.

o. Click the restart icon next to MySQL to restart the database and load your changes.

Alternatively, you may add the user record through the MySQL command line administration interface, and restart MySQL from the command line:

```
[root@server root]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 978 to server version: 3.23.58

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> insert into user values('server.domainname.com', 'sitedbusername',
password('sitedbuserpassword'), 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N
'N', 'N', 'N', 'N');
mysql> \q
```

```
Bye
[root@server root]# /etc/init.d/mysqld restart
```

4. **Add Mappings to the Site's Apache Configuration.**

Note

By default, Ensim will redirect URL's that match *.jsp to Tomcat to process. Since SoftSlate Commerce uses *.do and /do/* as well, you must add these mappings to the Site's Apache configuration.

a. Log on to the server through ssh.

b. Change to the Site's Apache configuration directory. Assuming the Site's ID is 15, change to the following directory:

```
[root@server root]# cd /etc/httpd/conf/site15
```

c. Create a file in this directory named `softslate`, whose contents are the following. (Replace "site15" with the Site ID of the site you are setting up.)

```
<Directory /home/virtual/site15/fst/var/www/html/softslate/WEB-INF/>
  AllowOverride None
  Deny from all
</Directory>

<IfModule mod_jk.c>
  JkMount /*.do  ajp13
  JkMount /softslate/do/*  ajp13
</IfModule>
```

d. Make sure to remove any extraneous files in the directory. (E.g., if you used Emacs, be sure to remove the `softslate~` file, if it exists. Apache will attempt to load all the files in the directory.)

e. Restart Apache for the new configuration to take effect:

```
[root@server root]# /etc/init.d/httpd restart
```

5. **Upload and Unpack SoftSlate Commerce.**

a. Upload the SoftSlate Commerce `.tar.gz` file to the server.

b. Log on to the server through ssh.

c. Change into the Site's `html` directory, and copy the `.tar.gz` file there. (Replace "site15" with the Site ID of the site you are setting up.)

```
[root@server root]# cd /home/virtual/site15/fst/var/www/html/
[root@server html]# cp /root/softslate-standard-1.0.5.tar.gz .
```

d. Untar the `.tar.gz` file:

```
[root@server html]# tar -zxf softslate-standard-1.0.5.tar.gz
```

e. For the Installer tool to run successfully, the owner of the Tomcat process must have *writable* access to the `/WEB-INF/conf/keys` directory and the `/WEB-INF/classes/appSettings.properties` file. Use **chown** and **chmod** to give the `tomcat4` user permission. (Replace "admin15" with the Site ID of the site you are setting up.)

```
[root@server html]# chown admin15.tomcat4 softslate/WEB-INF/conf/keys
[root@server html]# chown admin15.tomcat4 softslate/WEB-INF/classes/appSetti
[root@server html]# chmod g+w softslate/WEB-INF/conf/keys
[root@server html]# chmod g+w softslate/WEB-INF/classes/appSettings.properti
```

6. **Run the SoftSlate Commerce Installer.**

a. Visit `http://sitedomainname/softslate/installer` in a browser and follow the instructions to install SoftSlate Commerce.

b. If the application does not come up, use the Tomcat log file at `/var/tomcat4/logs/catalina.out` to troubleshoot the issues.

c. If the Installer tool fails to connect to the database, double-check the database name, user name, and password. You may need to associate the database user with your server's host name for the connection to work (see above).

d. Following a successful installation, you should change the permissions of the `/WEB-INF/conf/keys` directory and the `/WEB-INF/classes/appSettings.properties` file back for better security. (Replace "site15" and "admin15" with the Site ID of the site you are setting up.)

```
[root@server root]# cd /home/virtual/site15/fst/var/www/html/
[root@server html]# chown admin15.admin15 softslate/WEB-INF/conf/keys
[root@server html]# chown admin15.admin15 softslate/WEB-INF/classes/appSetti
[root@server html]# chmod g-w softslate/WEB-INF/conf/keys
```

```
[root@server html]# chmod g-w softslate/WEB-INF/classes/appSettings.properti
```

Chapter 2. Getting Started

Adding Your First Product

When you first install SoftSlate Commerce, it will be fully functional, but there will be no products in the store for your customers to buy. Follow these basic steps to add your first product and category to the store so you can start selling.

Example 2.1. Adding a Product

1. Log into the SoftSlate Commerce administrator using the user name and password you created in the Installer tool. By default the administrator application is located at /softslate/administrator [../../administrator]
2. Once logged in, navigate to Products -> Products [../../administrator/Product.do]. Click the "Add New Record" button on the right hand side of the screen to go to the add new product form.
3. After adding the product, create a category to place the product in, so it will show up in the store's category tree. Navigate to Products -> Categories [../../administrator/Category.do], and click the "Add New Record" button to add a category to the store.
4. Now assign your new product to the category you created:
    a. Navigate to Products -> Categories [../../administrator/Category.do].
    b. You'll see your new category in the table in the center of the screen. Click the "Products" link next to it to view the products assigned and not assigned to the category.
    c. You should see your new product in the table in the center of the screen. Click the "On" button next to "Edit Mode"
    d. Check the checkbox under the "Assigned" column next to the product.
    e. Click "Update/Delete" to save the change.

Placing a Test Order

Now that you've added a product to your store, you are open for business. Try placing a test order to make sure everything's working right.

Example 2.2. Placing an Order

1. Navigate to the welcome screen of SoftSlate Commerce. By default the welcome screen is located at the root of the application, or /softslate [../../]. You should see your new category appearing in the "Browse" box on the left hand side.
2. Click on the name of the category. This should take you to a page where your new product is listed. Click on the product's name. You should arrive on the product page, where you can add the product

to your cart.

3. Click the "Add to Cart" button to add the product to your cart, and then click the "Checkout" link to begin the checkout process.

4. Proceed through the checkout process. Create a test customer account, and enter a test address. When you arrive at the payment screen, use "Visa" as the credit card name, and "4111111111111111" as the credit card number. Finally, confirm your order to finalize it.

Viewing Your Test Order

After placing a test order you can go back into the administrator application to view the order and its details.

Example 2.3. Viewing an Order

1. Log into the SoftSlate Commerce administrator using the user name and password you created in the Installer tool. By default the administrator application is located at /softslate/administrator [../../administrator]

2. Once logged in, navigate to Orders and Customers -> Orders [../../administrator/Order.do]. You should see the test order you placed at the center of the screen. Click the "Details" link next to the order to view the billing information for the order.

3. From the order details screen, you can click the "Deliveries" link to view the order's delivery address and information, and from there you can follow links to view the items that were ordered. Click the "Payments" link to view the order's payment information. And click "Display Invoice" to view the invoice for the order, with all of the items and delivery information on one screen.

Congratulations! You've added your first product to your store, and placed your store's first order.

Chapter 3. Important Settings

Administrators can configure a large number of settings that control various aspects of SoftSlate Commerce, under the Settings menu of the Administrator application. The settings are organized for convenience into various categories, such as Store, System, Logic, Display, and so on.

Each individual setting under these categories is accompanied by a description explaining what its purpose is. In the vast majority of cases, the descriptions should be self-explanatory. But for some key settings, we've written the following sections to provide additional information on how to use them to the best effecct.

Email Template Settings

On the Settings -> Store [../../administrator/SettingsStore.do] screen, there are several settings that provide templates for various emails the store produces:

- Lost Password Email Subject, Text, and HTML templates (for emails sent to users when they fill out the lost password form)
- Invoice Email Subject, Text, and HTML templates (for emails sent to users upon completing an order)
- Notification Email Subject, Text, and HTML templates (for emails sent to the store when a user completes an order)

Each of these templates makes use of placeholders to indicate where within the template dynamic data, such as the user's password, their billing address, and so on, should appear.

For a list of all of the placeholders available for these and other settings in the Administrator, please refer to the Template Placeholders section.

Currency Code Setting

On the Settings -> System [../../administrator/SettingsSystem.do] screen, one of the settings is named "Store Currency Code". Here you must select the three-letter international currency code that the store is offering its products in.

SoftSlate Commerce supports international currency codes in the sense that you can select any ISO 4217 currency code for this setting. The store will look at each user's browser setting to determine the user's locale or origin. It will use the individual's locale to determine how to *format* the prices displayed throughout the store. For example, if your store's currency code is USD, for US dollars, a price would be displayed as `$12.50` for a user whose browser settings indicated he or she is from the US, whereas it would be displayed as `12.50 USD` if their browser indicates he or she is from another country.

There is currently no support in SoftSlate Commerce for selling the same products in multiple currencies and translating their prices through an exchange rate system.

Shipping Selection Required Setting

On the Settings -> Logic [../../administrator/SettingsLogic.do] screen, a common change you will want to make is to change the "Is a Shipping Selection Required?" setting from no to yes.

Settings -> Logic Screen

One of the first things you will probably want to do is define at least one shipping method under the Shipping Configuration -> Shipping Methods [../../administrator/ShippingMethod.do] screen. However, when you have done this your new shipping method will not show up during checkout to users unless you have also switched the "Is a Shipping Selection Required?" setting from no to yes.

SoftSlate Commerce is distributed with this setting turned off so you can place test orders immediately after installation, without having to define a shipping method.

Security Code Required Setting

On the Settings -> Logic [../../administrator/SettingsLogic.do] screen, please take special note of the setting named "Is a Security Code Required for Credit Cards?".

This is a global setting that all of SoftSlate Commerce's payment processors that ask for credit card information, including the Basic Payment Processor and Payflow Link and Payflow Pro processors, use when validating the user's submission of credit card information.

If it's set to "yes", the user must submit a security code with the credit card number. The security code is the three- or four-digit number on the back of the credit card. It is also variously known as the CVV2, CVC2, or CID number.

Default Layout Setting

An important setting for controlling multiple custom layouts is the "Default Layout" setting in the Settings -> Display [../../administrator/SettingsDisplay.do] screen of the administrator.

This setting tells SoftSlate Commerce which layout to give users by default, if the layout is not specified on the URL. Initially, it is set to "custom", meaning that fresh visitors coming to site will get the templates in the `/WEB-INF/layouts/custom` directory, if they exist. However, since the `custom` directory is empty when SoftSlate Commerce is first initialized, in effect visitors see the templates in the `default` directory when the store is first installed.

To clarify, here are the steps SoftSlate Commerce takes to determine which JSP template to use for a given user:

1. SoftSlate Commerce first looks for a request parameter on the URL or posted in a form named *`layout`*. If this parameter exists, its value is placed in the user's session and is carried with the user until the session expires.

2. If the user's *`layout`* has been set, SoftSlate Commerce looks for the JSP template under a directory of the same name, inside the `/WEB-INF/layouts` directory. In other words, if the user's *`layout`* has been set to `special`, the system uses the JSP templates inside the `/WEB-INF/layouts/special` directory.

3. If a given template does not exist under the directory defined by the *`layout`* parameter, the system next looks inside a directory of the same name as the Default Layout setting defined in the Settings -> Display [../../administrator/SettingsDisplay.do] screen. By default, this would be the `/WEB-INF/layouts/custom` directory.

4. If the template still cannot be found, the system uses the `/WEB-INF/layouts/default` directory to find the template.

Tip

For complete details on how to set up multiple custom layouts, refer to the Multiple Custom Layouts section.

Security Settings

The settings defined in the Settings -> Security [../../administrator/SettingsSecurity.do] screen should be changed with caution.

Settings -> Security Screen

SoftSlate Commerce does not store the method of encryption at the time a password or payment information is recorded. This has the following consequences, if you decide to switch the encryption settings after your store has been in operation:

- If you have not been using encryption and you switch the settings to use one- or two-way encryption, the system will treat all passwords, including ones defined previously and stored in plain text, as though they are encrypted. If you have to do this, you should first record the unencrytped passwords offline, and then update the passwords using the Administrator, in the Customers [../../administrator/Customer.do] or Administrators [../../administrator/Administrator.do] area. This will update the customer and adminstrator accounts with the same passwords as before, storing them now in encrypted format.

- Similarly if you have been using two-way encryption for passwords and you switch to no encryption or one-way encryption, none of the previous passwords will be recognized. You should record the passwords offline first, and then update the passwords using the Administrator, in the Customers [../../administrator/Customer.do] or Administrators [../../administrator/Administrator.do] area so they conform to the new encryption style.

- If you have been using one-way encryption for the passwords, there is no way to recover previous passwords if you then decide to switch the encryption settings. Customers and administrators will not be able to log in after making the switch.

- The same issues arise with respect to credit card information. The system will not be able to decrypt credit card numbers correctly if you change the encryption setting, unless you first record the information offline and then update each of the numbers for each order under the Orders [../../administrator/Order.do] area.

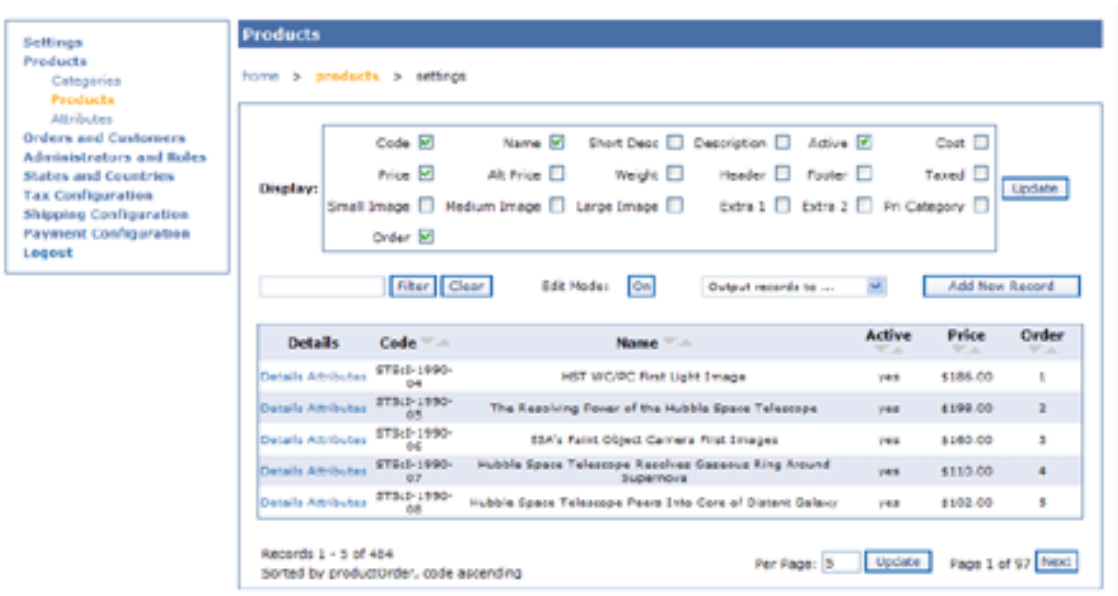
Note

The key used for two-way encryption is a file stored on the Web server. The only way to generate this file is through the Installer [../../installer] application. If you need to switch to two-way encryption, but you did not generate the key when you installed SoftSlate Commerce, you need to unlock the Installer tool and run the routine to generate the two-way key.

Chapter 4. Using the Administrator

The Administrator Control Screen

Throughout the Administrator, the same control screen is used when presenting information on products, categories, attributes, orders, customers, and other types of items. The control screen consists of various forms to manipulate what is being displayed and help you maintain the information. A good share of learning how to manage your store through the Administrator is simply learning how to use the features of the control screen.



The Administrator Control Screen for Products

Here is a summary of each of the forms on the control screen:

- Display fields form: This feature consists of the big box at the top of the screen. It allows you to control which columns are displayed on the screen. For example, on the Products [../../administrator/Product.do] screen, by default each product's code, name, active status, price, and order are displayed. You can change it so every column of information is presented, or only just a few.

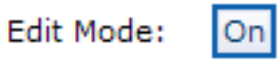
  The Display field form

- Filter form: This consists of the box on the left hand side of the screen next to the "Filter" and "Clear" buttons. The form allows you to filter the items that are displayed to just those that match a given search string. For example, to filter products on the Products [../../administrator/Product.do] screen to just those with the word "Jupiter" in the short description, name, or code, type "Jupiter" into the filter form and click the "Filter" button. This can be extremely useful when grappling with a

large number of records.

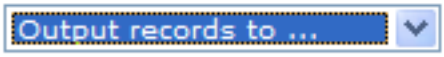
The Filter form

- Edit Mode button: Turning on the Edit Mode button allows you to edit every piece of information that's currently being displayed at once. For example, if you needed to update the prices of all your products at once, you could increase the page size to equal the number of products in the system. (Perhaps you would also want to limit the number of columns being displayed using the Display fields form, to just the Price and Name of the products.) Then click the Edit Mode button. Each of the prices (and any other fields being displayed) are now editable.



The Edit Mode button

- Output records to ... drop down: This feature allows you to create ad hoc reports by outputting the records on the screen, or all the records in the system, to either a printer-friendly format, or to an Excel spreadsheet. For example, to print out a report of the last 50 orders placed in the store, you can go to the Orders [../../administrator/Order.do] screen, enter 50 for the page size, sort the items by the order date in descending order, and then select "Print-Friendly Display" in the Output records to ... drop down menu.



The 'Output Records To ...' menu

- Add New Record button: On each of the screens, you can add a new record to the system--be it a product, order, state, etc.--by clicking the Add New Record button. That will take you to a form where you can add the item, whatever it is.

The Add New Record button

- Sorting arrows: Next to the label of each column, you'll find two grey arrows. Clicking the down arrow sorts the items being displayed in descending order. The up arrow sorts them in ascending order.
- Per Page form/ Next and Previous buttons: The per page form is self-explanatory. It controls the number of items currently being displayed on the screen. Similarly, the Next and Previous buttons help you navigate through the pages.

Page Controls

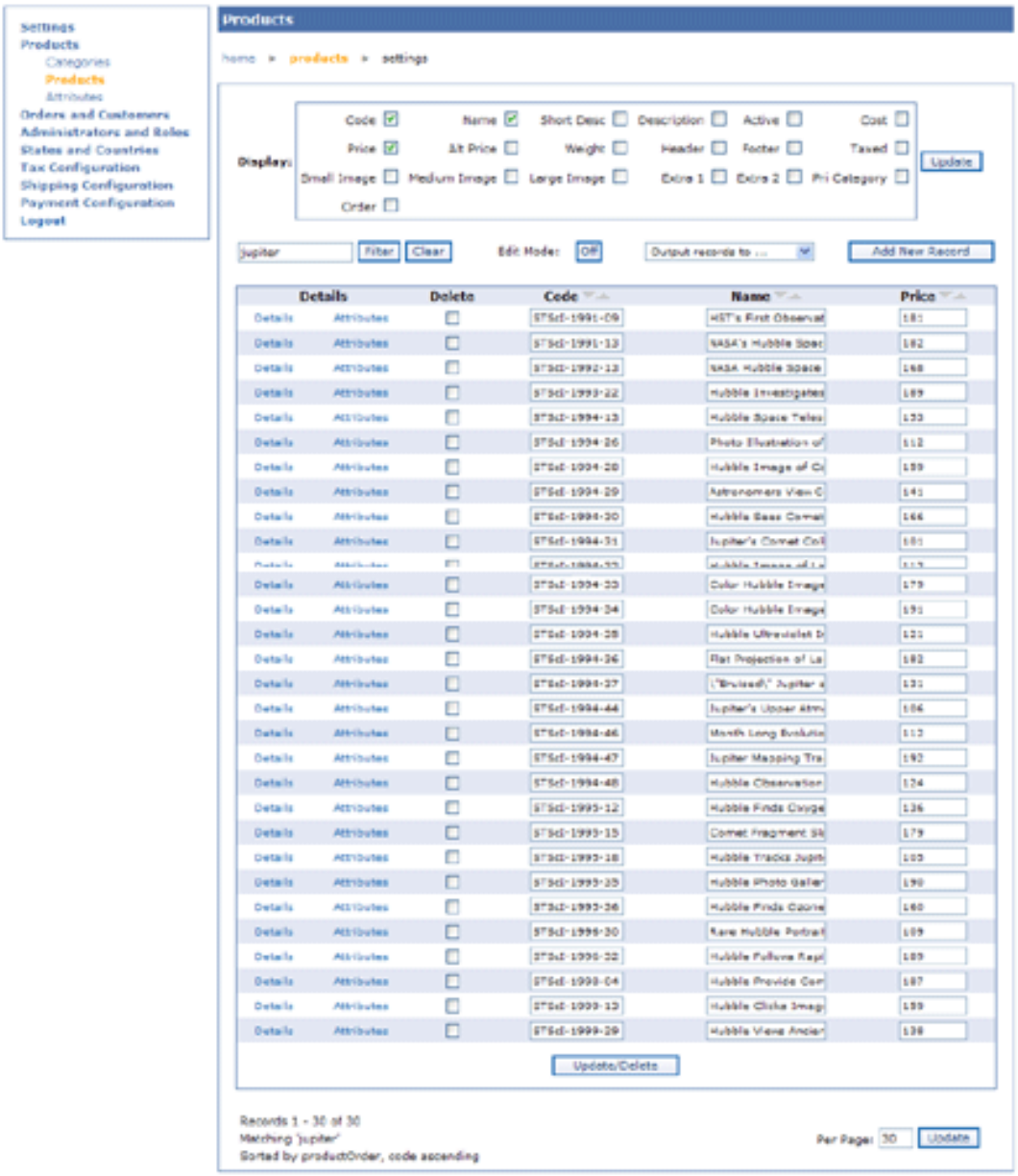
Saving Time with the Administrator Control

Screen

One of the ways the control screen features can save a dramatic amount of time is in the case of updating product prices, particularly if your store has a large number of products. The following example shows how to review and update a large group of products all at once. It uses the products from the demo store to demonstrate what to do.

Example 4.1. Updating Prices for a Group of Products at Once

1. Navigate to the Products -> Products [../../administrator/Product.do] control screen.
2. For convenience, use the Display form to limit the columns being displayed to just display the Products' code, name, and price. Click the "Update" button next to the box of field names to update.
3. Let's say you want to review all the products containing the word "Jupiter" in the name, code, or short description. Type "Jupiter" into the Filter form, and click the "Filter" button. You should get about 30 results matching the filter.
4. To make things easier, let's update the number of items displayed per page to fit everything on the first page. If you had 30 matches for the filter, change the Per Page form to "30", and click the "Update" button next to it. All the matching products should now appear on one page.
5. Now, click the "On" button in the Edit Mode form to make everything on the screen editable.
6. At this point, the control screen should look like this:



Updating Prices for Multiple Products

As you can see, it's very easy to scan through the list of products and review their prices, changing them when necessary. If you need to, use the sorting arrows next to the column names to organize the products further.

7. When you've finished updating the prices, simply hit the big "Update/Delete" button to have all your changes take effect.

You can use this same technique in the Administrator to make many other sorts of updates or to generate ad hoc reports. For example, it can help you:

- Update the statuses of all the recent orders you've received at once.
- Produce an Excel spreadsheet of all the customers in the store, along with their emails and mailing addresses.
- Produce various reports on the orders you've received, sorted by date, status, billing location, etc.
- Much more.

Making Assignments

In several areas of the Administrator, you need to make "assignments"; for example, assigning products to categories, and assigning attributes to products. In either of these cases, the process is essentially the same. The following steps describe how to create an attribute and assign it to a product. (The process of assigning products to categories is specifically covered in the Adding Your First Product section.)

Example 4.2. Creating an Attribute and Options, and Assigning it to a Product

1. Navigate to the Products -> Attributes [../../administrator/Attribute.do] screen.
2. To add a new attribute, click the "Add New Record Button."
3. An Attribute is used to provide customers with the ability to select options for their products, or enter personalized information. In this example, let's define an attribute named "Color".
4. Use "Color" for the name, and "COLOR" for the code, or whatever else you'd like to use.
5. For the type field, to have the Color options displayed next to each other on the product page using radio buttons, select "Expanded (radio buttons)"; otherwise, you can use "Compact (drop down)" to display the options in the form of a drop-down menu.
6. You can optionally enter in a price, cost, weight, description, etc. for the new attribute.
7. Hit "Add New Record" to add the new attribute to the system.
8. Now navigate back to the Products -> Attributes [../../administrator/Attribute.do] screen. You should see your new attribute in the table in the center of the screen.
9. Click the "Options" button next to the new attribute, and on the susequent screen click the "Add New Record" button to add a new Option under the Color attribute.
10. Add a new option named "Blue". Use "Blue" for the name, and "BLUE" for the code. Optionally, set the option's price, and add a description and image path. The price will be added to the customer's subtotal if he selects this option.
11. Hit "Add New Record" to add the new option to the system.
12. Repeat the above two steps for "Red", "Orange", "Yellow", or whatever other colors you like.
13. Now that you've added the Color attribute, and created the color options under it, you need to assign the attribute to a product. Navigate to Products -> Products [../../administrator/Product.do] screen.
14. Click on the "Attributes" link next to the product you wish to add the color attribute to. (If you

haven't created any products, click the "Add New Record" button to create one.)

15. You should now see the new Color attribute (along with any other attributes that were already defined) in the table in the center of the screen. To assign one or more of the attributes to the product, click the "On" button next to "Edit Mode".

16. Check the checkbox under the "Assigned" column next to the Color attribute.

17. Click "Update/Delete" to save the change.

Chapter 5. Configuring Taxes

SoftSlate Commerce is distributed with the ability to set up a tax table based on the location of each customer's delivery address. You can define separate tax rates by country, state, city, and postal code.

Basic Tax Rates

For the following example, let's suppose we operate a store in Boulder, Colorado. Our local laws tell us that we must charge tax for any sales we make to customers located in Colorado, and additional taxes on top of that for customers in Boulder County, and another tax on top of that for customers in the city of Boulder.

Example 5.1. Configuring Taxes for a Store in Boulder, Colorado

1. Navigate to the Tax Configuration -> General Settings [../../administrator/TaxSettings.do] screen.
2. Make sure that the "Are Tax Rates Cumulative" setting is set to yes. This tells the store to apply all the tax rates that match the customer's delivery address.
3. Make sure the "Basic Tax Rates" checkbox is checked.
4. Click "Save" to update the settings.
5. Now navigate to the Tax Configuration -> Basic Tax Rates [../../administrator/BasicTax.do] screen.
6. Click the "Add New Record Button" to begin adding tax rates for your store.
7. Add the first tax rate for all customers in Colorado. For "Country", select "United States" and for "State" select "Colorado". For "Amount", enter "2.9" for 2.9% state sales tax. Leave the other fields as "[Any]".

   #### Note

   The actual tax rates may be different than those used in this example. Please research the current tax rates for your area.

   Click the "Add New Record" button to add the first rate. This first rate will apply to all orders whose delivery addresses are in the United States and Colorado.
8. Now begin adding the tax rates for customers in Boulder County. To do this, we must find out all the postal codes that correspond to Boulder County. Research tells us the following postal codes apply: 80510, 80329, 80328, 80323, 80322, 80321, 80314, 80310, 80309, 80308, 80307, 80306, 80305, 80304, 80303, 80302, 80301, 80038, 80021, 80020, 80025, 80516, 80533, 80455, 80026, 80504, 80503, 80502, 80501, 80028, 80027, 80540, 80466, 80544, 80471, 80481.

   Add a tax rate record for each of the above postal codes. For each one, select "United States" as the country and "Colorado" as the state. For "Amount", enter ".65" for .65% county sales tax. Leave the city field marked as "[Any]".

   Click the "Add New Record" button to add each of the rates for Boulder County.
9. Finally, add a tax rate for the city of Boulder sales tax. For "Country", select "United States", for "State" select "Colorado", and for "City" type in "Boulder". For "Amount", enter "3.41" for 3.41%

sales tax. Leave the postal code field as "[Any]".

Click the "Add New Record" button to add the tax rate for the city of Boulder.

10. With these rates set up, each time a customer comes through checkout and uses a delivery address in Colorado, the tax rate corresponding to Colorado will be applied. If the delivery address's postal code matches one of Boulder County's postal codes, the county tax rate will also be applied. And if the delivery address's city matches "Boulder", the city tax rate will also be applied.

Note

Postal codes do not always line up precisely with county boundaries, and sometimes an address might use a given city for its mailing address but actually be outside that city's tax district. In other words, using Basic Tax Rates may not be 100% accurate for 100% of your customers, but it is likely to be accurate for the vast majority.

Chapter 6. Configuring Shipping

SoftSlate Commerce is distributed with the ability to set up several different kinds of shipping tables to determing the shipping costs for your customers. You can define shipping tables based on the quantity, total price, or total weight of the customer's order. In addition, you can define a flat rate for shipping that applies regardless of the quantity, weight, or price.

Flat Rate Shipping Methods

Let's suppose you operate a store that has very simple rules for shipping costs: if the customer wants their order to be rushed to them, you charge a flat rate of $9.95 for shipping. Otherwise, you charge $5.95. Setting up this type of shipping is quite simple.

Example 6.1. Charging Flat Rates for Shipping

1. Navigate to the Shipping Configuration -> Shipping Methods [../../administrator/ShippingMethod.do] screen.
2. Click "Add New Record" to add the first shipping method, for standard, 5.95 shipping.
3. Use "Standard" for the name of the method, "STD" for the code, leave Minimum Charge at 0, but change Base Charge to 5.95.

   ## Note

   Leave dollar signs or other currency symbols off when entering price values.

4. Under Method Type, select "Flat Rate".
5. Click "Add New Record" to add the new method to the store.
6. Now add the next shipping method, for rushed, 9.95 shipping.
7. Use "Rushed" for the name of the method, "RSH" for the code, leave Minimum Charge at 0, but change Base Charge to 9.95.
8. Under Method Type, again select "Flat Rate".
9. Click "Add New Record" to add the new method to the store.
10. Navigate back to the Shipping Configuration -> Shipping Methods [../../administrator/ShippingMethod.do] screen.
11. You should see both of your methods listed in the table at the center of the screen. Customers will see both options during checkout and can select either Standard (5.95) shipping, or Rushed (9.95) shipping for their order.

Important

No shipping methods will show up during checkout to users unless you have switched the "Is a Shipping Selection Required?" setting from no to yes on the Settings -> Logic

[../../administrator/SettingsLogic.do] screen. For more details, refer to the section on the Shipping Selection Required Setting.

Quantity-Based Shipping Tables

It's common for stores to base the amount they charge for shipping on the quantity of items the customer is ordering. For the following example, let's suppose you operate a store where you would like to charge a base of $5.00 handling for every order. In addition to that, you'd like to charge $1.50 for each of the first three items to be shipped, plus $0.50 for every item after the third. This sort of policy corresponds to a "Cumulative Quantity Table" shipping method type.

Example 6.2. Charging Shipping Based on the Quantity of Items in the Order

1. Navigate to the Shipping Configuration -> Shipping Methods [../../administrator/ShippingMethod.do] screen.
2. Click "Add New Record" to add the shipping method.
3. Use "Standard" for the name of the method, "STD" for the code.
4. Leave Minimum Charge at 0, but change Base Charge to 5.00. This corresponds to the amount of handling charges you want applied to every order.

   ## Note

   Leave dollar signs or other currency symbols off when entering price values.

5. Under Method Type, select "Cumulative Quantity Table".
6. Click "Add New Record" to add the new method to the store.
7. Navigate back to the Shipping Configuration -> Shipping Methods [../../administrator/ShippingMethod.do] screen.
8. You should see your new method listed in the table at the center of the screen.
9. Click the "Rates" link next to the new method to add the two levels of quantities you need.
10. Click "Add New Record" from the "Rates" screen to add the first quantity range.
11. Use 0 for the floor of the first range, 3 as the ceiling, and use 1.50 for the amount of this first quantity range.

    This tells the store to use this rate if the number of items in the user's cart is greater or equal to 0 and less than or equal to 3.

12. Click "Add New Record" to add the first rate to the shipping method.
13. Now enter the next range in the quantity-based shipping table. Use 4 for the floor of the first range, 0 as the ceiling, and 0.50 as the amount.

    This tells the store to use this rate if the number of items in the user's cart is greater or equal to 4, up to infinity (0 as the ceiling means no upper limit).

    ## Important

Ranges for the Cumulative Quantity Table must not overlap. In other words, make sure the floor for one range is larger than the ceiling for the previous range. Otherwise, the amounts from both ranges will be applied when computing a customer's shipping cost.

14. Click "Add New Record" to add the second rate to the shipping method.

As an example of how this works, if a user has 5 items in his or her cart, the store will compute the shipping costs for this method to be 5.00 (from the Base Charge for the method), plus 1.50 for the first item, plus 1.50 for the second item, and plus another 1.50 for the third. At this point the second range will kick in, and the store will add 0.50 for the fourth item and another 0.50 for the fifth. The total shipping cost presented to the customer is 10.50.

Sometimes it's simpler and more convenient to set a static rate for each quantity range, rather than having the costs accumluate for each item as they do in this example. This is done by selecting the regular "Quantity Table" for the Method Type of the shipping method instead of "Cumulative Quantity Table". In that case, the number you enter for the "Amount" of each quantity range is the *flat amount* that will be charged if the user's order falls within that range (plus the Base Charge), rather than the *price per item*.

Important

No shipping methods will show up during checkout to users unless you have switched the "Is a Shipping Selection Required?" setting from no to yes on the Settings -> Logic [../../administrator/SettingsLogic.do] screen. For more details, refer to the section on the Shipping Selection Required Setting.

Weight-Based Shipping Tables

Another common practice is for stores to base the amount they charge for shipping on the total weight of items the customer is ordering. For the following example, let's suppose you operate a store where you would like to charge a base of $5.00 handling for every order. In addition to that, you'd like to charge $1.00 per pound for orders whose weights total between 0 and 10 pounds. For heavier orders totaling more than 10 pounds, you'd like to charge $0.75 per pound. This sort of policy corresponds to a "Per Weight Unit Table" shipping method type.

Example 6.3. Charging Shipping Based on the Weight of Items in the Order

1. Navigate to the Shipping Configuration -> Shipping Methods [../../administrator/ShippingMethod.do] screen.
2. Click "Add New Record" to add the shipping method.
3. Use "Standard" for the name of the method, "STD" for the code.
4. Leave Minimum Charge at 0, but change Base Charge to 5.00. This corresponds to the amount of handling charges you want applied to every order.

   ### Note

   Leave dollar signs or other currency symbols off when entering price values.

5. Under Method Type, select "Per Weight Unit Table".

6. Click "Add New Record" to add the new method to the store.

7. Navigate back to the Shipping Configuration -> Shipping Methods [../../administrator/ShippingMethod.do] screen.

8. You should see your new method listed in the table at the center of the screen.

9. Click the "Rates" link next to the new method to add the two levels of weights you need.

10. Click "Add New Record" from the "Rates" screen to add the first weight range.

11. Use 0 for the floor of the first range, 10 as the ceiling, and use 1.00 for the amount of this first weight range.

    This tells the store to use this rate if the total weight of items in the user's cart is greater or equal to 0 and less than or equal to 10.

12. Click "Add New Record" to add the first rate to the shipping method.

13. Now enter the next range in the weight-based shipping table. Use 10 for the floor of the first range, 0 as the ceiling, and 0.75 as the amount.

    This tells the store to use this rate if the total weight of items in the user's cart is greater or equal to 10, up to infinity (0 as the ceiling means no upper limit).

    ## Important

    Ranges for the Per Weight Unit Table can overlap, in which case the rate of the lower range will be used. In other words, it's ok to make the ceiling for the lower range equal to the floor for the following range. Because weights are computed as decimal values, doing this will ensure that no order will fall through the cracks (i.e. it will make sure all possibilities are accounted for if the order's weight is something like 10.01).

14. Click "Add New Record" to add the second rate to the shipping method.

As an example of how this works, if a user is placing an order and the total weight of all the items in the order is 15.25 pounds, the store will compute the shipping costs for this method to be 5.00 (from the Base Charge for the method), plus 15.25 times 0.75 (because 15.25 falls in the second range). The shipping cost presented to the customer will be 16.44.

Sometimes it's simpler and more convenient to set a static rate for each weight range, rather than having the costs be multiplied out per unit of weight as they are in this example. This is done by selecting the regular "Weight Table" for the Method Type of the shipping method instead of "Per Unit Weight Table". In that case, the number you enter for the "Amount" of each weight range is the *flat amount* that will be charged if the user's order falls within that range (plus the Base Charge), rather than the *price per unit of weight*.

Important

No shipping methods will show up during checkout to users unless you have switched the "Is a Shipping Selection Required?" setting from no to yes on the Settings -> Logic [../../administrator/SettingsLogic.do] screen. For more details, refer to the section on the Shipping Selection Required Setting.

Price-Based Shipping Tables

Many stores find they need to correspond the shipping costs they charge to the total price of the customer's order. For example, let's suppose you would like to reward customers who make large purchases in your store by offering free shipping on orders of $200 or more. You can set this up quite easily by creating a shipping method with a "Price Table" method type.

Example 6.4. Offering Free Shipping on Large Orders

1. Navigate to the Shipping Configuration -> Shipping Methods [../../administrator/ShippingMethod.do] screen.
2. Click "Add New Record" to add the shipping method.
3. Use "Free Shipping for Orders Over $200" for the name of the method, "FREE" for the code.
4. Leave Minimum Charge at 0, and leave the Base Charge at 0.

   ## Note

   Leave dollar signs or other currency symbols off when entering price values.

5. Under Method Type, select "Price Table".
6. Click "Add New Record" to add the new method to the store.
7. Navigate back to the Shipping Configuration -> Shipping Methods [../../administrator/ShippingMethod.do] screen.
8. You should see your new method listed in the table at the center of the screen.
9. Click the "Rates" link next to the new method to add your free shipping rate to the method.
10. Click "Add New Record" from the "Rates" screen to add a range corresponding to the free shipping offer.
11. Use 200 for the floor of the range, and 0 as the ceiling, and use 0 for the amount of this range.

    This tells the store to use this rate if the total price of items in the user's cart is greater or equal to 200, up to infinity (0 as the ceiling means no upper limit).
12. Click "Add New Record" to add the rate to the shipping method.

Now when a customer goes through checkout the store will check to see if the total price of items in the order is greater or equal to 200. If it is, it will display this method, "Free Shipping for Orders Over $200", as one of the options for the customer to choose.

Important

No shipping methods will show up during checkout to users unless you have switched the "Is a Shipping Selection Required?" setting from no to yes on the Settings -> Logic [../../administrator/SettingsLogic.do] screen. For more details, refer to the section on the Shipping Selection Required Setting.

Chapter 7. Configuring Payments

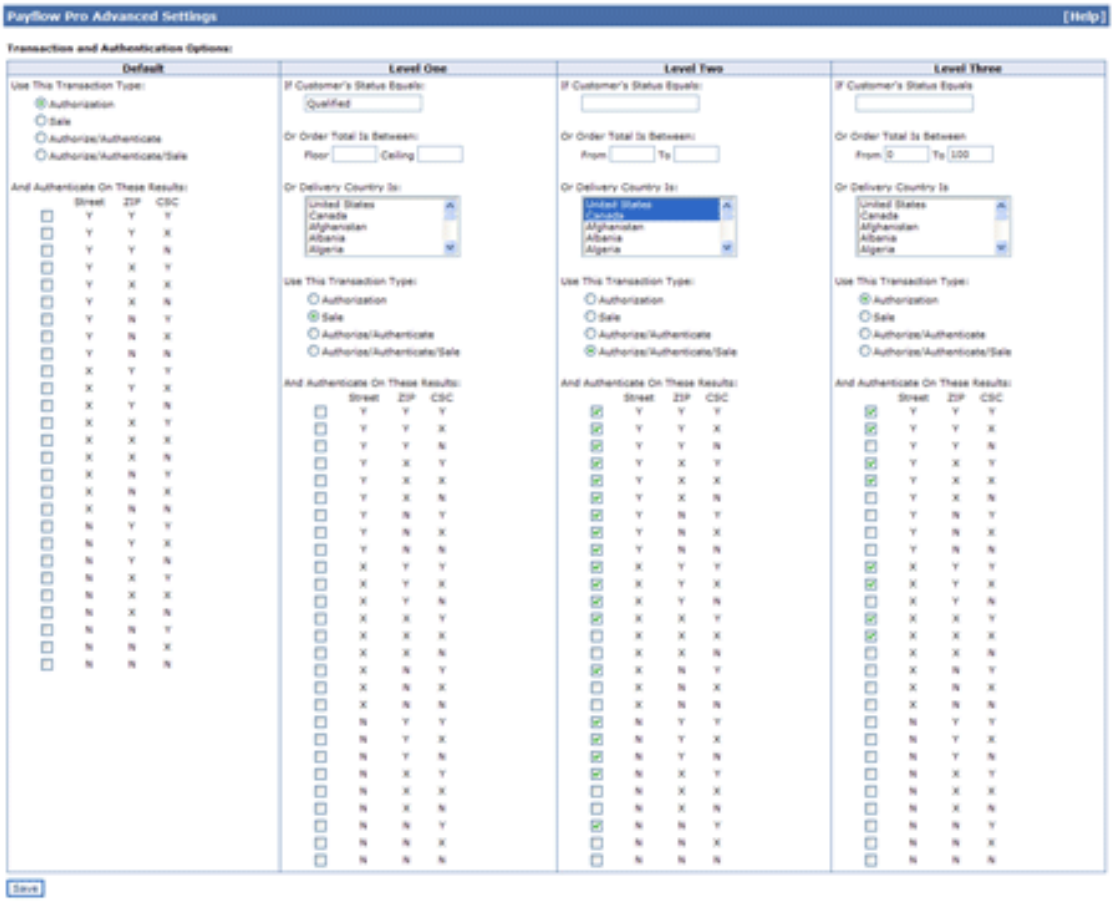
SoftSlate Commerce is distributed with several types of payment processors. The Basic Payment Processor allows you to capture credit card information for each order. When combined with two-way encryption (see the Security Settings [../../administrator/SettingsSecurity.do] screen), you can store the information in the database in encrypted format, and later retrieve it from the Payments screen for each order under the Orders and Customers -> Orders [../../administrator/SettingsSecurity.do] screen. This is useful if your store needs to process credit cards through a separate, offline system.

SoftSlate Commerce also integrates with VeriSign's Payflow Link and Payflow Pro products to provide online credit card processing in real time. You must establish an account with VeriSign and enter login information into the Payflow Link [../../administrator/PayflowLink.do] or Payflow Pro [../../administrator/PayflowPro.do] settings screens for these processors to work.

Note

Several of the fields on the Payflow Link [../../administrator/PayflowLink.do] and Payflow Pro [../../administrator/PayflowPro.do] screens accept placeholders that are replaced with dynamic values for each customer's order. For a list of accepted placeholders, refer to the Template Placeholders Reference section.

Payflow Pro Advanced Settings



Payflow Pro Advanced Settings screen. Larger Image [images/payflowProAdvancedLarge.gif].

The Payflow Pro Processor provides a number of advanced settings allowing you to tailor your online processing to your business, on the Payflow Pro Advanced Settings [../../administrator/PayflowProTransAuth.do] screen. To explain how these advanced settings work, here is a step-by-step guide for how the Payflow Pro Processor handles each order as it is submitted by a customer.

1. The Payflow Pro Processor starts by comparing the customer's order with the settings defined under "Level One" of the Payflow Pro Advanced Settings:

    - *If Customer's Status Equals:* If the customer placing the order is logged in and has a "Status" corresponding to this value, the Level One settings will be applied to the order. All customers' status values can be modified at any time on the Orders and Customers -> Customers [../../administrator/Customer.do] screen. For example, you may want to identify customers who have ordered in the past and label them with a status of "Qualified". When these customers come through to place a new order, you can automatically capture the order by setting the transaction type to "Sale" for their customer status. On the other hand, for all other customers, or for users who are placing an order without an account, you may want to authorize and authenticate their purchases first.

    - *If Order Total Is Between:* If there is no match on the customer's status, but the total price of the items in the current order falls within this range, the Level One settings will be applied to the order. This allows you to treat small orders, which may not be of great concern in terms of fraud, differently than large orders, which may require vetting. You could allow orders under, say, $100 to be captured automatically while larger orders would be authorized only, pending a manual review.

    - *Delivery Country Is:* If there is no match on either the customer's status or the order total, but the delivery address's country matches a selection in this drop down menu, the Level One settings will be applied to the order. The idea is you can treat orders going to different countries differently. Some countries may be a constant source of fraudulent orders, and others may not support AVS or card security code validations. In these cases, you can use this setting to identify those countries you wish to treat differently.

2. If none of the above settings under "Level One" applies, the processor next looks for a match with the "Level Two" settings, and, failing that, a match with the "Level Three" settings. If no match can be found, the processor uses the "Transaction Type" and "Authenticate On These Results" settings under the "Default" column.

3. Having determine which level of settings the given customer falls under, the processor next identifies the Transaction Type to be used for the order:

    - *Authorization:* If this option is selected under the level of settings that is in effect for the customer, an authorization request is sent to Payflow Pro, which places a hold on the customer's credit card account matching the amount of the order. You must capture the funds at a later time.

    - *Sale:* If this option is selected, the funds are immediately captured from the customer's credit card. There is no need to perform a delayed capture at a later time.

    - *Authorize/Authenticate:* First, the processor sends Payflow Pro an authorization request, which places a hold on the customer's credit card account matching the amount of the order. Next, the AVS and card security code results that come back from the authorization are tested against the checkboxes under the "Authenticate On These Results" settings. If the results are acceptable, the customer is considered "Authenticated" and the order is allowed to go through. If the results do not match one of the authentication checkboxes, the transaction is considered declined. An error message is given to the customer on the credit card payment screen asking him or her to double check the billing address and security code he or she has entered.

- *Authorize/Authenticate/Sale:* First, the processor sends Payflow Pro an authorization request, which places a hold on the customer's credit card account matching the amount of the order. Next, the AVS and card security code results that come back from the authorization are tested against the checkboxes under the "Authenticate On These Results" settings. If the results are acceptable, the customer is considered "Authenticated" and *an immediate Sale request is sent to Payflow Pro, capturing the funds that were just authorized*. If the results do not match one of the authentication checkboxes, the transaction is considered declined. An error message is given to the customer on the credit card payment screen asking him or her to double check the billing address and security code he or she has entered.

4. For the "Authorize/Authenticate" and "Authorize/Authenticate/Sale" transaction types, the processor checks the returned values for AVS Street, AVS ZIP, and card security code against the list of checkboxes. Note that a "Y" indicates a successful match for the street address, ZIP code, or security code. A "N" indicates a mismatch, and an "X" indicates that the customer's bank does not support that type of validation.

Chapter 8. Template Placeholders

Template Placeholders Reference

Several settings in the Administrator provide templates where the following placeholders can be used where dynamic data is meant to go. Please refer to the following table when constructing the templates.

Table 8.1. Template Placeholders Reference

Template Placeholder/Token	Description	Java Expression
%%LOGIN_URL%%	The URL used by a customer to log into his or her account. Useful for the Lost Password Email templates.	`(String) base-Form.getSettingsBean().getValue("customerSecureURL") + "/Account.do"`
%%STORE_NAME%%	The name of the store as defined in the Settings -> Store [../../administrator/SettingsStore.do] screen.	`(String) base-Form.getSettingsBean().getValue("storeName")`
%%STORE_ADDRESS1%%	The first line of the store's address, as defined in the Settings -> Store [../../administrator/SettingsStore.do] screen.	`(String) base-Form.getSettingsBean().getValue("storeAddress1")`
%%STORE_ADDRESS2%%	The second line of the store's address, as defined in the Settings -> Store [../../administrator/SettingsStore.do] screen.	`(String) base-Form.getSettingsBean().getValue("storeAddress2")`
%%STORE_CITY%%	The store's city, as defined in the Settings -> Store [../../administrator/SettingsStore.do] screen.	`(String) base-Form.getSettingsBean().getValue("storeCity")`
%%STORE_STATE%%	The store's state or province, as defined in the Settings -> Store [../../administrator/SettingsStore.do] screen.	`(String) base-Form.getSettingsBean().getValue("storeStateOrProvince")`
%%STORE_POSTALCODE%%	The store's postal code, as defined in the Settings -> Store [../../administrator/SettingsStore.do] screen.	`(String) base-Form.getSettingsBean().getValue("storePostalCode")`
%%STORE_COUNTRY%%	The store's country, as defined in the Settings -> Store [../../administrator/SettingsStore.do] screen.	`(String) base-Form.getSettingsBean().getValue("storeCountry")`
%%STORE_PHONE%%	The store's phone number, as defined in the Settings -> Store [../../administrator/SettingsStore.do] screen.	`(String) base-Form.getSettingsBean().getValue("storePhone")`
%%STORE_EMAIL%%	The store's email address, as defined in the Settings -> Store [../../administrator/SettingsStore.do] screen.	`(String) base-Form.getSettingsBean().getValue("storeEmail")`

Template Placeholder/Token	Description	Java Expression
%%STORE_FAX%%	The store's fax number, as defined in the Settings -> Store [../../administrator/SettingsStore.do] screen.	`(String) baseForm.getSettingsBean().getValue("storeFax")`
%%ORDER_ID%%	The orderID from the `npcOrder` database table for the current userw's order, if it is defined.	`Integer.toString(activeUser.getOrder().getOrderID())`
%%CUSTOMER_ID%%	The customerID from the `npcOrder` database table for the current user's order, if it is defined.	`Integer.toString(activeUser.getOrder().getCustomerID())`
%%USER_NAME%%	The userName from the `npcOrder` database table for the current user's order, if it is defined.	`activeUser.getOrder().getUserName()`
%%ORDER_NUMBER%%	The orderNumber from the `npcOrder` database table for the current user's order, if it is defined.	`Integer.toString(activeUser.getOrder().getOrderNumber())`
%%ORDER_TOTAL%%	The formatted order total for the current user's order, if it is defined.	`activeUser.getOrder().getFormattedTotal()`
%%SHIPPING%%	The formatted shipping amount for the current user's order, if it is defined.	`activeUser.getOrder().getFormattedShipping()`
%%TAX%%	The formatted tax amount for the current user's order, if it is defined.	`activeUser.getOrder().getFormattedTax()`
%%ORDER_SUBTOTAL%%	The formatted subtotal (total minus tax and shipping) amount for the current user's order, if it is defined.	`activeUser.getOrder().getFormattedSubtotal()`
%%ORDER_TAXABLESUBTOTAL%%	The formatted taxable subtotal amount for the current user's order, if it is defined.	`activeUser.getOrder().getFormattedTaxableSubtotal()`
%%ORDER_DATE%%	The date the current user's order was completed.	`activeUser.getOrder().getFormattedCompleted()`
%%ORDER_LASTMODIFIED%%	The date the current user's order was last modified.	`activeUser.getOrder().getFormattedLastModified()`
%%ORDER_STATUS%%	The status from the `npcOrder` database table for the current user's order, if it is defined.	`activeUser.getOrder().getStatus()`
%%ORDER_STATUSDETAILS%%	The statusDetails from the `npcOrder` database table for the current user's order, if it is defined.	`activeUser.getOrder().getStatusDetails()`
%%ORDER_TRACKINGNUMBER%%	The trackingNumber from the `npcOrder` database table for the current user's order, if it is defined.	`activeUser.getOrder().getTrackingNumber()`
%%ORDER_WEIGHT%%	The total weight from the `npcOrder` database table for the current user's order, if it is defined.	`Double.toString(activeUser.getOrder().getWeight())`

Template Placeholder/Token	Description	Java Expression
%%ORDER_QUANTITY%%	The total quantity of items from the npcOrder database table for the current user's order, if it is defined.	Double.toString(activeUser.getOrder().getQuantity())
%%BILL_FIRSTNAME%%	The first name of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getFirstName()
%%BILL_LASTNAME%%	The last name of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getLastName()
%%BILL_ORGANIZATION%%	The organization name of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getOrganization()
%%BILL_ADDRESS1%%	The first line of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getAddress1()
%%BILL_ADDRESS2%%	The second line of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getAddress2()
%%BILL_CITY%%	The city of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getCity()
%%BILL_STATE%%	The state or province of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getState()
%%BILL_POSTALCODE%%	The postal code of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getPostalCode()
%%BILL_COUNTRY%%	The country of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getCountry()
%%BILL_PHONE1%%	The daytime phone number of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getPhone1()
%%BILL_PHONE2%%	The evening phone number of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getPhone2()
%%BILL_EMAIL1%%	The primary email address of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getEmail1()
%%BILL_EMAIL2%%	The secondary email address of the billing address for the current user's order, if it is defined.	activeUser.getOrder().getEmail2()
%%BILL_EXTRA1%%	The extra1 field from the npcOrder database table for the current user's order, if it is defined.	activeUser.getOrder().getExtra1()
%%BILL_EXTRA2%%	The extra2 field from the npcOrder database table for the current user's order, if it is defined.	activeUser.getOrder().getExtra2()
%%BILL_NOTES%%	The notes field from the npcOrder database table for the current user's order, if it is defined.	activeUser.getOrder().getNotes()

Template Placeholder/Token	Description	Java Expression
%%DELIVERY_ID%%	The deliveryID from the npcOrderDelivery database table for the current user's delivery record, if it is defined.	Integer.toString(activeUser.getOrder().firstDelivery().getOrderDeliveryID())
%%DELIVERY_TOTAL%%	The formatted total for the current user's primary delivery, if it is defined.	activeUser.getOrder().firstDelivery().getFormattedTotal()
%%DELIVERY_SHIPPING%%	The formatted shipping amount for the current user's primary delivery, if it is defined.	activeUser.getOrder().firstDelivery().getFormattedShipping()
%%DELIVERY_TAX%%	The formatted tax amount for the current user's primary delivery, if it is defined.	activeUser.getOrder().firstDelivery().getFormattedTax()
%%DELIVERY_SUBTOTAL%%	The formatted subtotal (total minus tax and shipping) amount for the current user's primary delivery, if it is defined.	activeUser.getOrder().firstDelivery().getFormattedSubtotal()
%%DELIVERY_TAXABLESUBTOTAL%%	The formatted taxable subtotal amount for the current user's primary delivery, if it is defined.	activeUser.getOrder().firstDelivery().getFormattedTaxableSubtotal()
%%DELIVERY_DATE%%	The date the current user's primary delivery was created.	activeUser.getOrder().firstDelivery().getFormattedCreated()
%%DELIVERY_LASTMODIFIED%%	The date the current user's primary delivery was last modified.	activeUser.getOrder().firstDelivery().getFormattedLastModified()
%%DELIVERY_STATUS%%	The status from the npcOrderDelivery database table for the current user's primary delivery, if it is defined.	activeUser.getOrder().firstDelivery().getStatus()
%%DELIVERY_STATUSDETAILS%%	The statusDetails from the npcOrderDelivery database table for the current user's primary delivery, if it is defined.	activeUser.getOrder().firstDelivery().getStatusDetails()
%%DELIVERY_TRACKINGNUMBER%%	The trackingNumber from the npcOrderDelivery database table for the current user's primary delivery, if it is defined.	activeUser.getOrder().firstDelivery().getTrackingNumber()
%%DELIVERY_WEIGHT%%	The total weight from the npcOrderDelivery database table for the current user's primary delivery, if it is defined.	Double.toString(activeUser.getOrder().firstDelivery().getWeight())

Template Placeholder/Token	Description	Java Expression
%%DELIVERY_QUANTITY%%	The quantity of items from the npcOrderDelivery database table for the current user's primary delivery, if it is defined.	Double.toString(activeUser.getOrder().firstDelivery().getQuantity())
%%DELIVERY_FIRSTNAME%%	The first name of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getFirstName()
%%DELIVERY_LASTNAME%%	The last name of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getLastName()
%%DELIVERY_ORGANIZATION%%	The organization name of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDeliv-ery().getOrganization()
%%DELIVERY_ADDRESS1%%	The first line of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getAddress1()
%%DELIVERY_ADDRESS2%%	The second line of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getAddress2()
%%DELIVERY_CITY%%	The city of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getCity()
%%DELIVERY_STATE%%	The state or province of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getState()
%%DELIVERY_POSTALCODE%%	The postal code of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getPostalCode()
%%DELIVERY_COUNTRY%%	The country of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getCountry()
%%DELIVERY_PHONE1%%	The daytime phone number of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getPhone1()
%%DELIVERY_PHONE2%%	The evening phone number of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getPhone2()
%%DELIVERY_EMAIL1%%	The primary email address of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getEmail1()
%%DELIVERY_EMAIL2%%	The secondary email address of the delivery address for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getEmail2()
%%DELIVERY_EXTRA1%%	The extra1 field from the npcOrderDelivery database table for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getExtra1()
%%DELIVERY_EXTRA2%%	The extra2 field from the npcOrderDelivery database table for the current user's order, if it is defined.	activeUser.getOrder().firstDelivery().getExtra2()
%%DELIVERY_NOTES%%	The notes field from the npcOrd-	act-

Template Placeholder/Token	Description	Java Expression
	`erDelivery` database table for the current user's order, if it is defined.	`iveUser.getOrder().firstDelivery().getNotes()`
%%START_ITEMS%%	Marks the beginning of a loop through the order items for the current user's order.	
%%CODE%%	The code of the order item currently being looped through.	`orderItem.getCode()`
%%NAME%%	The name of the order item currently being looped through.	`orderItem.getName()`
%%DESCRIPTION%%	The name of the order item currently being looped through.	`orderItem.getName()`
%%QUANTITY%%	The quantity of the order item currently being looped through.	`orderItem.getQuantity()`
%%COST%%	The unit cost of the order item currently being looped through.	`orderItem.getFormattedCost()`
%%PRICE%%	The unit price of the order item currently being looped through.	`orderItem.getFormattedPrice()`
%%ALT_PRICE%%	The alternate price of the order item currently being looped through.	`orderItem.getFormattedAltPrice()`
%%WEIGHT%%	The total weight of the order item currently being looped through.	`orderItem.getWeight()`
%%PRODUCT_WEIGHT%%	The unit weight of the order item currently being looped through.	`orderItem.getProductWeight()`
%%TOTAL%%	The total, extended price of the order item currently being looped through.	`orderItem.getFormattedTotal()`
%%SHORT_DESCRIPTION	The short description of the order item currently being looped through.	`orderItem.getShortDescription()`
%%EXTRA1%%	The value of the extra1 field of the order item currently being looped through.	`orderItem.getExtra1()`
%%EXTRA2%%	The value of the extra2 field of the order item currently being looped through.	`orderItem.getExtra2()`
%%EXTRA3%%	The value of the extra3 field of the order item currently being looped through.	`orderItem.getExtra3()`
%%EXTRA4%%	The value of the extra4 field of the order item currently being looped through.	`orderItem.getExtra4()`
%%EXTRA5%%	The value of the extra5 field of the order item currently being looped through.	`orderItem.getExtra5()`
%%START_ATTRIBUTES%%	Marks the beginning of a loop through the attributes for the current order item being looped through.	
%%ATTRIBUTE_CODE%%	The code of the current order item attribute being looped through.	`(String) attribute.get("attributeCode")`

Template Placeholder/Token	Description	Java Expression
%%ATTRIBUTE_NAME%%	The name of the current order item attribute being looped through.	`(String) attribute.get("attributeName")`
%%ATTRIBUTE_VALUE%%	The value chosen or entered by the user for the current order item attribute being looped through.	`(String) attribute.get("attributeValue")`
%%OIA_TOTAL%%	The total price for the current order item attribute being looped through.	`(String) attribute.get("total")`
%%OIA_WEIGHT%%	The total weight for the current order item attribute being looped through.	`(String) attribute.get("weight")`
%%ATTRIBUTE_COST%%	The unit cost of the order item attribute currently being looped through.	`(String) attribute.get("attributeCost")`
%%ATTRIBUTE_PRICE%%	The unit price of the order item attribute currently being looped through.	`(String) attribute.get("attributePrice")`
%%ATTRIBUTE_ALT_PRICE%%	The alternate price of the order attribute item currently being looped through.	`(String) attribute.get("attributeAltPrice")`
%%ATTRIBUTE_WEIGHT%%	The weight of the order item attribute currently being looped through.	`(String) attribute.get("attributeWeight")`
%%OPTION_CODE%%	The code of the option chosen by the user for the current order item attribute being looped through.	`(String) attribute.get("optionCode")`
%%OPTION_NAME%%	The name of the option chosen by the user for the current order item attribute being looped through.	`(String) attribute.get("optionName")`
%%OPTION_COST%%	The unit cost of the option chosen by the user for the order item attribute currently being looped through.	`(String) attribute.get("optionCost")`
%%OPTION_PRICE%%	The unit price of the option chosen by the user for the order item attribute currently being looped through.	`(String) attribute.get("optionPrice")`
%%OPTION_ALT_PRICE%%	The alternate price of the option chosen by the user for the order attribute item currently being looped through.	`(String) attribute.get("optionAltPrice")`
%%OPTION_WEIGHT%%	The weight of the option chosen by the user for the order item attribute currently being looped through.	`(String) attribute.get("optionWeight")`
%%END_ATTRIBUTES%%	Marks the end of a loop through the attributes for the current order item being looped through.	
%%END_ITEMS%%	Marks the end of a loop through the order items for the current user's order.	

Chapter 9. Upgrading SoftSlate Commerce

Upgrading Overview

The steps to upgrade SoftSlate Commerce from one version to another are the same regardless of the version you are upgrading from or to. The process of upgrading consists of first upgrading the application's files on the server, and then running the Upgrades tool in the Administrator to upgrade the data objects in the database.

To find out which version you are currently running, go to the Upgrades screen [../../administrator/Upgrades.do] of the Administrator. That screen tells you the current application version and data object version of the program. (You'll use this same screen to upgrade the database objects.)

- *Application Version:* The version of SoftSlate Commerce corresponding to the files residing on the server. (Specifically, this value is produced by the JSP templates used to produce the Administrator's screens.)
- *Data Objects Version:* The version of SoftSlate Commerce corresponding to the database objects, such as the table struture or the values of database records, for the database SoftSlate Commerce is connected to. (Specifically this value comes from the "dataObjectsVersion" record of the `npcSetting` table.)

Note

Unless you are in the middle of an upgrade, the application version and the data objects version should be exactly the same or unknown behavior may result, and a warning message will appear in the Administrator.

Important: Read This Before Upgrading

Before upgrading your copy of SoftSlate Commerce, please review the following details about how to handle any customizations you may have made to the application. If you have made customizations to your installation of SoftSlate Commerce, in most cases an upgrade will preserve your changes without any manual intervention required. There are some exceptions, however.

Warning

Even if you have made your customizations using the techniques described in this documentation, there is still a small chance an upgrade will cause errors, since the files and code are by necessity interdependant to some degree. For this reason we strongly suggest running the upgrade in a test environment and placing some test orders before doing so on a production installation.

1. **Upgrading After Customizing Configuration Files and JSP Templates.**

   Many configuration files, and any custom JSP templates that are outside the `default` layout directory, are not replaced during the upgrade process. This allows you to preserve the customizations you have made while still leveraging changes to those files you have not customized. Specifically, if your customizations have been limited to the following files and directories, installing an upgrade

will not overwrite them.

Warning

If you have made changes to any other files besides the ones listed here *an upgrade will overwrite the modified files and eliminate your changes*. In these cases, please consider refactoring your changes into one of the following custom files.

Files and Directories Not Overwritten with Upgrades

- `/WEB-INF/classes/log4j.properties` (logging settings)
- `/WEB-INF/classes/appSettings.properties` (database and other application settings)
- `/WEB-INF/conf/keys` (default location of encryption keys)
- `/WEB-INF/layouts/custom` (default location of custom JSP templates)
- All other directories under `/WEB-INF/layouts` other than `default`, which *is* overwritten.
- `/css/style-custom.css` (custom css stylesheet)
- `/WEB-INF/conf/core/tiles-defs-pages.xml` (used for Tiles definitions of custom pages)
- All `struts-config-custom.xml` files under `/WEB-INF/conf` (used for custom Struts mappings)
- All `tiles-defs-custom.xml` files under `/WEB-INF/conf` (used for custom Tiles definitions)
- All `sql-custom.properties` under `/WEB-INF/classes/resources` (used for custom SQL queries)
- All `application-custom.properties` under `/WEB-INF/classes/resources` (used for custom messages)

2. **Upgrading After Customizing SoftSlate Commerce Source Code.**

   The best way to make source code changes to SoftSlate Commerce is to subclass one of the existing classes in the `com.softslate.commerce` Java package, and place the new class in your own Java package. Then, from the Settings -> Components [../../administrator/SettingsComponents.do] screen in the Administrator, you can update the implementing class name for the interface that is affected.

   It is possible, however, to modify the classes that come with SoftSlate Commerce and recompile them to make changes. If you have done this with a class in the `com.softslate.commerce` Java package, *the classes you have modified will be overwritten and replaced with the version of the class from the upgrade.* This pertains to both uncompiled source `.java` files in the `/WEB-INF/src` directory and compiled `.class` files in the `/WEB-INF/classes` directory.

   If this is the case, we suggest you refactor the changes you have made into a subclass as described above. Alternatively you can extract the upgrade and manually merge any changes that have taken place to the class in question, compiling the merged class after you've made the merge.

3. **Upgrading from the Free Edition to the Standard Edition.**

With one exception, the upgrade process described in the following section applies equally well to upgrades from the Free Edition to the Standard Edition, as it does for upgrades within the two editions. If you are upgrading from the Free Edition to the Standard Edition, the only additional step is to remove the `/WEB-INF/lib/softslate.jar` file from your installation before or after the upgrade process. The classes in the `softslate.jar` file will be replaced by `.class` files in the `/WEB-INF/classes` directory after the upgrade to the Standard Edition. This structure for the Standard Edition allows you to easily recompile the application from source code if necessary.

Upgrade Procedure

Note

Some of the following steps are illustrated using Tomcat 5 as the servlet container. If you are using a different application server, please identify the corresponding steps for the application server you are using before you begin.

1. **Prepare for the Upgrade.**

   a. *Make a Backup of the SoftSlate Commerce Application.* Make a back up copy of the current application directory. This is most important step of the entire process. If anything goes wrong you can restore the back up and investigate.

   ## Important

   Please don't skip this step! Making a backup copy of the working application directory is the most important step of the upgrade, because it means you can always go back to the current version of the application if anything goes wrong.

   b. *Review the Upgrade.* Review important release notes for the latest versions at http://www.softslate.com/demo/documentation/html/releaseNotes.html. Examine the Change Log for the new features and bug fixes for the latest release at http://www.softslate.com/demo/documentation/html/changeLog.html.

   c. *Download the Upgrade.* Download the SoftSlate Commerce upgrade from the URL given to you. If you don't know where to find the upgrade, contact sales or support. For Unix, the upgrade file will be a `.tar.gz` file and will look like this: `softslate-standardupgrade-x.x.x.tar.gz` or `softslate-compiledupgrade-x.x.x.tar.gz`. For Windows, the upgrade file will be `.zip` file and will look like this: `softslate-standardupgrade-x.x.x.zip` or `softslate-compiledupgrade-x.x.x.zip`.

   d. *Upload and Extract the Upgrade.* Upload the upgrade to your server and extract its contents into a working directory *outside the application server's area.* (Later, you will copy the contents of the upgrade into the SoftSlate Commerce directory, after bringing the application down.)

      For example, for a Unix platform, run the following command to untar the upgrade into a working directory:

      tar -zxf softslate-standardupgrade-1.0.4.tar.gz

      A directory named `softslate` should appear with the contents of the upgrade inside it.

      On Windows, right-click on the file and select "Extract All..." to unzip the file.

e. *Precompile New JSP Templates (Optional).* Optionally, you can precompile the new JSP templates for better performance after you restart the application. A build function for Tomcat 5 is provided to assist with this (SoftSlate Commerce Standard Edition only).

Use these steps to precompile on a Unix server (Tomcat 5 only):

i. **cd** into the `/WEB-INF/build` directory of SoftSlate Commerce.

ii. Run **./build.sh precompile -Dtomcat.home=<path to Tomcat installation> -Dpath.root=<path to working `softslate` directory>**. The compiled templates will be placed in the `/WEB-INF/classes/org/apache/jsp` directory. For example, if Tomcat is installed in the `/usr/local/jakarta-tomcat` directory and the working directory you've unpacked the files into is `/home/user/softslate`, you would run the following command:

./build.sh precompile -Dtomcat.home=/usr/local/jakarta-tomcat -Dpath.root=/home/user/softslate>

Note: It may take several minutes for the compilation to finish.

Use these steps to precompile on a Windows server (Tomcat 5 only):

i. Go to `Start -> All Programs -> Accessories -> Command Prompt` to open up a command prompt window.

ii. **cd** into the `/WEB-INF/build` directory of SoftSlate Commerce.

iii. Run **build.bat precompile -Dtomcat.home=<path to Tomcat installation> -Dpath.root=<path to working `softslate` directory>**. The compiled templates will be placed in the `/WEB-INF/classes/org/apache/jsp` directory. For example, if Tomcat is installed in the `E:\jakarta-tomcat` directory and the working directory you've unpacked the files into is `E:\softslate`, you would run the following command:

build.bat precompile -Dtomcat.home=E:\jakarta-tomcat -Dpath.root=E:\softslate>

Note: It may take several minutes for the compilation to finish.

f. *Create a Temporary HTML Page.* You should expect the following steps to take at least 10 minutes to complete, during which time SoftSlate Commerce will be unreachable. Consider creating an HTML page explaining to customers that the store will be down for maintenance. This file can be placed in the Web server so that when the application is brought down everyone will see it.

2. **Bring Down the SoftSlate Commerce Web Application.**

a. *Stop the SoftSlate Commerce Web Application.* For example, if you are using Tomcat use the Manager application to run the stop command:

```
http://localhost:8080/manager/stop?path=/softslate
```

b. *Test the Temporary HTML Page.* If you have created a static HTML page to inform users the store is down, verify that it is in place and working.

3. **Copy the Upgrade Over the Existing Application Directory.**

- *Unix.* Next, copy the contents of the upgrade you extracted into the application directory, replacing all existing files of the same name.

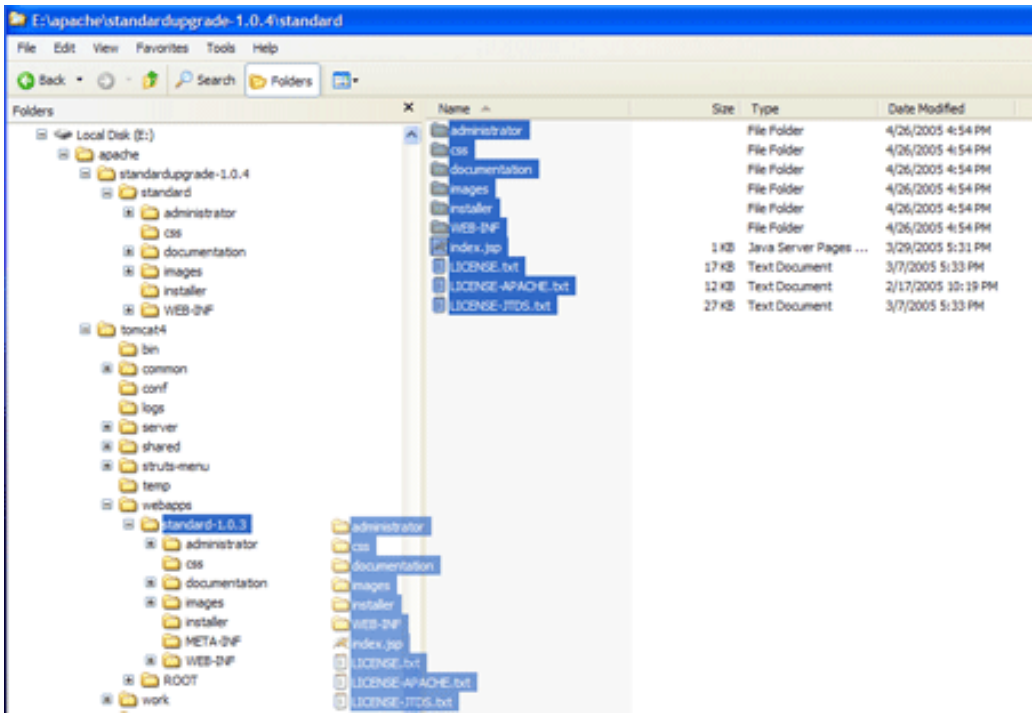
On Unix use **cp -R** to perform the copy. This will replace the contents of the application directory with the contents of the upgrade directory. For example, if you extracted the upgrade into `/home/myuser`, and the application directory is at `/usr/local/jakarta-tomcat/webapps/softslate`, you would run this command to perform the copy:

cp -R --reply=yes /home/myuser/softslate/* /usr/local/jakarta-tomcat/webapps/softslate

After the copy, if necessary, you should **chown** the application directory to the correct owner and group:

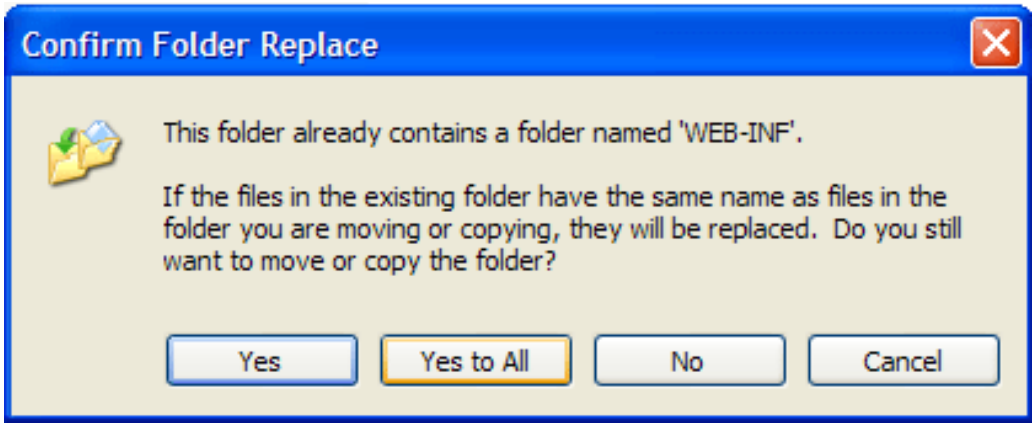
chown -R tomcat.tomcat /usr/local/jakarta-tomcat/webapps/softslate

- *Windows.* On Windows use Windows Explorer to perform the copy:



Copying an Upgrade Over an Existing Application on Windows

When prompted to replace the existing content, select "Yes to All":



4. **Replace Previously Compiled Templates.**

   a. *Delete Previously Compiled JSP Templates.* Delete all the previously compiled JSP templates for the SoftSlate Commerce application. For example, for Tomcat, if the Tomcat installation directory is `/usr/local/jakarta-tomcat`, you would run this command:

      rm -r /usr/local/jakarta-tomcat/work/Catalina/localhost/softslate/org/apache/jsp

   b. *Copy Over Precompiled Templates (Optional).* If you precompiled the new JSP templates (above), copy or move them in to place now.

      Use these steps to copy the precompiled templates on a Unix server (Tomcat 5 only):

      i. Copy the precompiled templates into the `work` under the Tomcat distribution:

         mv /usr/local/jakarta-tomcat/webapps/softslate/WEB-INF/classes/org/apache/jsp / usr/local/jakarta-tomcat/work/Catalina/localhost/softslate/org/apache

      ii. Change the ownership of the precompiled templates to the owner of the Tomcat process:

         chown -R tomcat.tomcat / usr/local/jakarta-tomcat/work/Catalina/localhost/softslate/org/apache/jsp.

      On Windows, use Windows Explorer to copy the precompiled templates into the `work` directory under the Tomcat distribution. For example, for Tomcat, the compiled templates should go in this directory:

      `<tomcat-home>/work/Catalina/localhost/softslate/org/apache/jsp`

5. **Restart SoftSlate Commerce.**

   a. *Start the SoftSlate Commerce Application.* For example, if you are using Tomcat use the Manager application to run the start command:

      `http://localhost:8080/manager/start?path=/softslate`

6. **Upgrade the Database Objects.** Now that you've upgraded the application files for SoftSlate Commerce, you must upgrade the database objects that go with them. Until you do so, upgraded and fixed features will not work properly.

   a. *Log In to the Administrator*. Log in to the Administrator [../../administrator/Welcome.do]. You should see a warning message indicating that the application is out of sync with the database

objects.

b. *Run the Uprades Tool.* Navigate to the Upgrades screen [../../administrator/Upgrades.do], and click the "Run Upgrades" button to perform the database upgrades needed. If the upgrades are successful, you should see a success message and the warning should go away.

7. **Post-Upgrade Testing.**

a. *Place a Test Order.* Test the application by placing a test order and logging into the Administrator.

b. *Test the Upgraded Features.* Test the new features and bug fixes for the upgrade. For a list of the new and fixed features, visit http://www.softslate.com/demo/documentation/html/changeLog.html.

Appendix A. Change Log

```
all         1.0.14 Free Edition product limit increased from 25 to 100
all         1.0.14 Improvements and clarifications in log4j.properties file
customer             1.0.14 State field not required if user chooses a country tha
customer             1.0.14 Improvements in email validations
customer             1.0.14 Fixed bug where taxes on shipping could be applied mor
customer             1.0.14 Fixed error message when an invalid quantity is added
customer             1.0.13 Added image order item field, several attribute fields
customer             1.0.13 Fixed bugs related to using the 'Other' field for the
documentation 1.0.13 Updated template placeholder documentation
documentation 1.0.13 New design for documentation header
administrator 1.0.12 Fixed bug with reset value for Category.do in the Administrat
customer             1.0.12 Fixed bug in end user's build.xml file related to jar
customer             1.0.12 Fixed bug with account menu not going away immediately
customer             1.0.12 Switched in new SoftSlate logos for header and footer
customer             1.0.12 Added the log in screen to the checkout breadcrumbing
all         1.0.11 Interface changes: adding Map return values, etc. (see release
all         1.0.11 Product limit for Free Edition changed to 25
documentation 1.0.11 Simple rules for customizations
all         1.0.10 Short description, extra fields added to order item object
all         1.0.10 Additional extra fields for orders, order delivs, customers, an
all         1.0.10 Custom product settings and category settings
customer             1.0.10 Customers can edit attribute selections on the edit ca
customer             1.0.10 Fixed bug with shipping options on the single screen o
administrator 1.0.9 Added ids for display for most objects
administrator 1.0.9 Incorporated new NetPush Order drivers
administrator 1.0.9 Delete function on the edit screen for all objects
administrator 1.0.9 Fixed bug when initializing states and countries after update
administrator 1.0.9 On/off Edit Mode button more intuitive
administrator 1.0.9 Fixed bug when viewing the orders for a given customer
administrator 1.0.9 Editing an item in the administrator takes you back to the con
administrator 1.0.9 Admin control screen form parameters stored in session, and se
administrator 1.0.9 Filtering items on the control screen reverts display to first
all         1.0.9 Product limit check for compiled build
all         1.0.9 Adjustments to SEO feature to allow for one subdirectory level i
customer             1.0.9 Fixed bug so the create new account form repopulates on
installer            1.0.9 Fixed bug in basic installer regarding encryption type
administrator 1.0.8 Address field settings set to empty string rather than null if
administrator 1.0.8 Delete expired orders function
all         1.0.8 Upgraded MySQL driver to 3.0.16-ga
all         1.0.8 Renamed mysql, jtds, common-codec, and struts-menu .jar files
all         1.0.8 Integrated Apache commons codec for MD5 hashes
customer             1.0.8 Fixed bug when adding inactive or nonexistant product t
customer             1.0.8 fixed bug when processing single screen order form and
customer             1.0.8 Forcing order total to 2 decimal places after taxes
customer             1.0.8 Request attribute "loggedIn" representing if the user i
customer             1.0.8 Sensitive data removed from debug level log messages
customer             1.0.8 CategoryBean made serializable for session's current ca
customer             1.0.8 Links around the powered by logo
customer             1.0.8 Bug with single-page order form when a payment error oc
installer            1.0.8 Removing index.jsp from the upgrade distributions so it
administrator 1.0.7 Prevent PayflowPro advanced processing/ logout processing upon
administrator 1.0.7 Product and category additional extra fields
administrator 1.0.7 JavaScript error on login form fixed
administrator 1.0.7 Hitslink interface
customer             1.0.7 Payflow Pro integration (testing completed)
customer             1.0.7 Product and category page keywords and descriptions
customer             1.0.7 Fixed single-screen order form
customer             1.0.7 JavaScript to prevent duplicate form submissions
```

```
administrator 1.0.6 Support for application rebranding
customer             1.0.6 Support for application rebranding
documentation 1.0.6 Updated Ensim install docs
documentation 1.0.6 Support for application rebranding
installer            1.0.6 Support for application rebranding
administrator 1.0.5 Ported all JSP templates to work with Tomcat 4.0
administrator 1.0.5 Ported all JSP templates to work with WebLogic 8.1
customer             1.0.5 Streamlined styles.jsp and styles.css
customer             1.0.5 Ported all JSP templates to work with Tomcat 4.0
customer             1.0.5 Ported all JSP templates to work with WebLogic 8.1
customer             1.0.5 Product page can use bean named 'product' or 'productFo
documentation 1.0.5 Expanded documentation for installing SoftSlate Commerce
installer            1.0.5 Linking installer to the documentation
installer            1.0.5 Fixing line endings for shell scripts
installer            1.0.5 Create streamlined, basic installer vs. 'advanced' vers
installer            1.0.5 Ported all JSP templates to work with Tomcat 4.0
administrator 1.0.4 Upgrades screen, for database objects upgrades
administrator 1.0.4 General tax settings screen
administrator 1.0.4 General shipping settings screen
administrator 1.0.4 Payflow Pro settings and advanced settings
administrator 1.0.4 Integration of documentation with the Administrator
administrator 1.0.4 Eliminate lazy images in admin menu
administrator 1.0.4 General payment settings screen
administrator 1.0.4 Payflow link settings screen
customer             1.0.4 Billing and delivery address field validations (zip, ph
customer             1.0.4 Payflow Link integration
customer             1.0.4 Payment processors now return a Map that includes a 're
customer             1.0.4 Fix visited link color
documentation 1.0.4 Guide for Administrators first version completed
installer            1.0.4 Created insert.sql files for each module, to eliminate
installer            1.0.3 If key already exists, don't overwrite, add message exp
installer            1.0.3 Eliminate 'other' fields in the db settings screen
administrator 1.0.2 Making warning on security settings page red and bold
administrator 1.0.2 Clicking Unassigned/Assigned buttons should reset page to firs
administrator 1.0.2 Change "top menu" style label to "nav bar menu"
administrator 1.0.2 Change "Header BG Color" to "Grid Header BG Color"
administrator 1.0.2 Explain more precisely what the button styles affect (what's a
administrator 1.0.2 Bug using filter on orders screen in MySQL
customer             1.0.2 Make links from order history to product page unsecure
customer             1.0.2 Tomcat 4 compatibility changes
customer             1.0.2 Adding sql-custom.properties files that will not be ove
customer             1.0.2 If basket total is 0, do not require payment
customer             1.0.2 Add hot links to product names in order history
customer             1.0.2 Compute taxes to two, not four decimal places
customer             1.0.2 Powered by SoftSlate Commerce logo in default footer
customer             1.0.2 In attributesAndOptions.jsp, remove HTML filter from at
customer             1.0.2 In attributesAndOptions.jsp, display an attribute price
customer             1.0.2 Moving the final order processing away from BasePayment
customer             1.0.2 Eliminate body link styles in style.css/ add style-cust
installer            1.0.2 Create stylesheet color db settings under 'styles' not
administrator 1.0.1 Dashboard statistics - all time orders - fixed
administrator 1.0.1 Dashboard statistics - last 30 days - fixed
administrator 1.0.1 Built-in categories reloaded after product assignments
administrator 1.0.1 Message/error background uses static stylesheet
administrator 1.0.1 Updating an order now keeps the customerID with the record
customer             1.0.1 Custom layouts fixed - new tag for tiles:insert looks f
customer             1.0.1 Problem in attributesAndOptions.jsp fixed for drop down
customer             1.0.1 Comment identifying the file name in layout.jsp fixed
customer             1.0.1 HTML filter set to false for product and category names
documentation 1.0.1 Getting started -> adding first product: split out last step i
documentation 1.0.1 Using styles.css: change to explain how to use styles-custom.c
installer            1.0.1 Key generation and test connection proceed even if erro
```

Appendix B. Release Notes

Version 1.0.11

Interface Changes.

Several interface changes were made to make the API more flexible. If you have created your own custom Java classes please take note of the following interface changes. If you have created an implementation of any of these interfaces, please update your implementations as you upgrade your application to 1.0.11.

1. `AdministratorProcessor.processLogin` now returns a Map rather than void
2. `CustomerProcessor.processRegister` now returns a Map rather than void
3. `CustomerProcessor.processLogin` now returns a Map rather than void
4. `CustomerProcessor.updateAddresses` now returns a Map rather than void
5. `CustomerProcessor.updateCustomer` now returns a Map rather than void
6. `CustomerProcessor.loadOrderHistory` now returns a Map rather than void
7. `CartProcessor.processRemoveItem` now returns a Map rather than void
8. `CartProcessor.processCheckoutAddresses` now returns a Map rather than void
9. `CartProcessor.processOrderComplete` now takes a Map as an argument rather than no arguments
10. `ShippingProcessor.processShipping` now returns a Map rather than void
11. `TaxProcessor.processTax` now returns a Map rather than void
12. `AdministratorGatewayDAO.processLogin` now returns a Map rather than void
13. `CustomerGatewayDAO.processRegister` now returns a Map rather than void
14. `OrderGatewayDAO.processOrderComplete` now takes an Order and a Map as arguments rather than just an Order

Version 1.0.8

Name Changes for .jar Files.

The following .jar files were renamed to eliminate the version numbers from the file names. This will allow for smoother upgrades of these files in future versions. Before upgrading to 1.0.8, please *delete* the following files from the application's `/WEB-INF/lib` directory. When you copy in the files for the upgrade, the new, renamed versions will take their place.

1. `jtds-1.0.2.jar`
2. `struts-menu-2.3.jar`

3. `mysql-connector-java-2.0.14-bin.jar`
4. `commons-codec-1.3.jar`

Part II. Guide for Designers

Table of Contents

Chapter 10. Making Basic Customizations

Dynamic Font Styles

When you first install SoftSlate Commerce it is configured by default to use the font and color styles defined dynamically in the administrator, under the Settings -> Styles [../../administrator/SettingsStyles.do] screen. If you only need to make limited changes to the store's look-and-feel, consider simply using this screen in the administrator.

Settings -> Styles Screen

To see these settings at work, follow these steps to make a sample change:

Example 10.1. Changing the Body Font Size

1. Log into the SoftSlate Commerce administrator and navigate to the Settings -> Styles [../../administrator/SettingsStyles.do] screen.
2. Make sure the very first setting, "Use Dynamic Styles" is set to yes. If it is set to no, none of your changes will take effect.
3. As an example, change the "Body Font Size" setting to a larger size, for example 12 or 13, and click "Save".
4. Navigate to the store's welcome screen (/softslate [../../]). You should see that the font size for much of the text on the screen is now larger.
5. In a similar way, you can make changes to all the various color settings, font weight settings ("bold" or "normal"), and font-style settings ("italic" or "normal").

Tip

Explore the various types of fonts defined in the Settings -> Styles [../../administrator/SettingsStyles.do] screen. Specific text on various screens throughout the store is often controlled by a separate set of font settings.

Using CSS Stylesheets Instead of Dynamic Styles

When you first install SoftSlate Commerce it is configured by default to use the font and color styles defined in the administrator, under the Settings -> Styles [../../administrator/SettingsStyles.do] screen.

However, if you are familiar with CSS (Cascading Style Sheets) it is best to turn off dynamic styles and use a CSS stylesheet directly to define your store's styles.

SoftSlate Commerce comes with a stylesheet named `style.css`, which is located inside the `/css` directory of SoftSlate Commerce. This stylesheet contains all the default style settings used in the store.

To change the store so that it uses `style.css` directly follow these steps:

Example 10.2. Setting the Store to Use CSS Stylesheets

1. Log into the SoftSlate Commerce administrator and navigate to the Settings -> Styles [../../administrator/SettingsStyles.do] screen.
2. Change the very first setting, "Use Dynamic Styles" to no, and click "Save".

You *could* edit `style.css` directly to change your font styles. *However*, we strongly recommend you leave `style.css` unmodified. If you upgrade your copy of SoftSlate Commerce in the future, the `style.css` file may be modified or overridden by the upgrade process. Instead, place your custom styles in the `styles-custom.css` file, also located in the `/css` directory. You can do so safely without fear that your changes will be overridden with a future upgrade. Since the `style-custom.css` file is included after the `style.css` file, styles you define there will take precedence.

Using the CSS stylesheets gives your store a performance benefit, since most browsers will cache CSS files, whereas with dynamic styles, a second request must be made to the server with each page hit to retrieve the dynamic styles produced from the administrator settings.

Caution

Initially the `style.css` stylesheet matches the dynamic settings defined in the administrator when SoftSlate Commerce was first installed. It does not carry over any changes you may have made to the settings in the administrator.

Display Settings

Several other look and feel settings can be controlled in the administrator, under the Settings -> Display [../../administrator/SettingsDisplay.do] screen.

Settings -> Display Screen

Here are a few of the display settings you're most likely to want to change:

- The "Header Background Color" setting controls the color displayed as the background for the headers of various grids in the store, such as the list of products on the Category Screen, or the list cart items on the Cart Screen.
- Similarly, the "Alternate Grid Background Color" affects the background color of the alternating rows in the grids of the store.
- The "Left Side Width" setting controls the width of the column on the left-hand side of the screen, which contains the search box and the category tree.
- The "Layout Space Size" setting is used throuhout the default templates to add spacing around the various elements of the screens--for example, to separate the store's search box from the category tree.

Note

All of the settings defined under the "Settings" menu in the administrator are stored in the database in the `npcSetting` database table.

Chapter 11. Creating a Custom Layout

Chances are before too long you will want to make more extensive changes to your store's look and feel than simply modifying the font styles and miscellaneous display settings. Fortunately SoftSlate Commerce gives you the ability to customize every piece of HTML produced by the application. It does so by allowing you to create your own custom JSP templates and to use them to replace the default templates that come with the application. Some familiarity with HTML is necessary for you to create your own templates. Familiarity with JSP and Struts tags is also a benefit, though not a necessity.

Setting Up Your Design Environment

Before making your first customizations, consider the design environment you'd like to use when creating your custom screens. There are many software products out there that provide features for editing HTML and JSP documents. Among the most popular are Macromedia's Dreamweaver [http://www.macromedia.com/software/dreamweaver] and the open-source editor jEdit [http://www.jedit.org/]. However, any text editor can be used to edit the JSP templates, including Microsoft's Notepad, or the open-source editor Emacs [http://www.gnu.org/software/emacs].

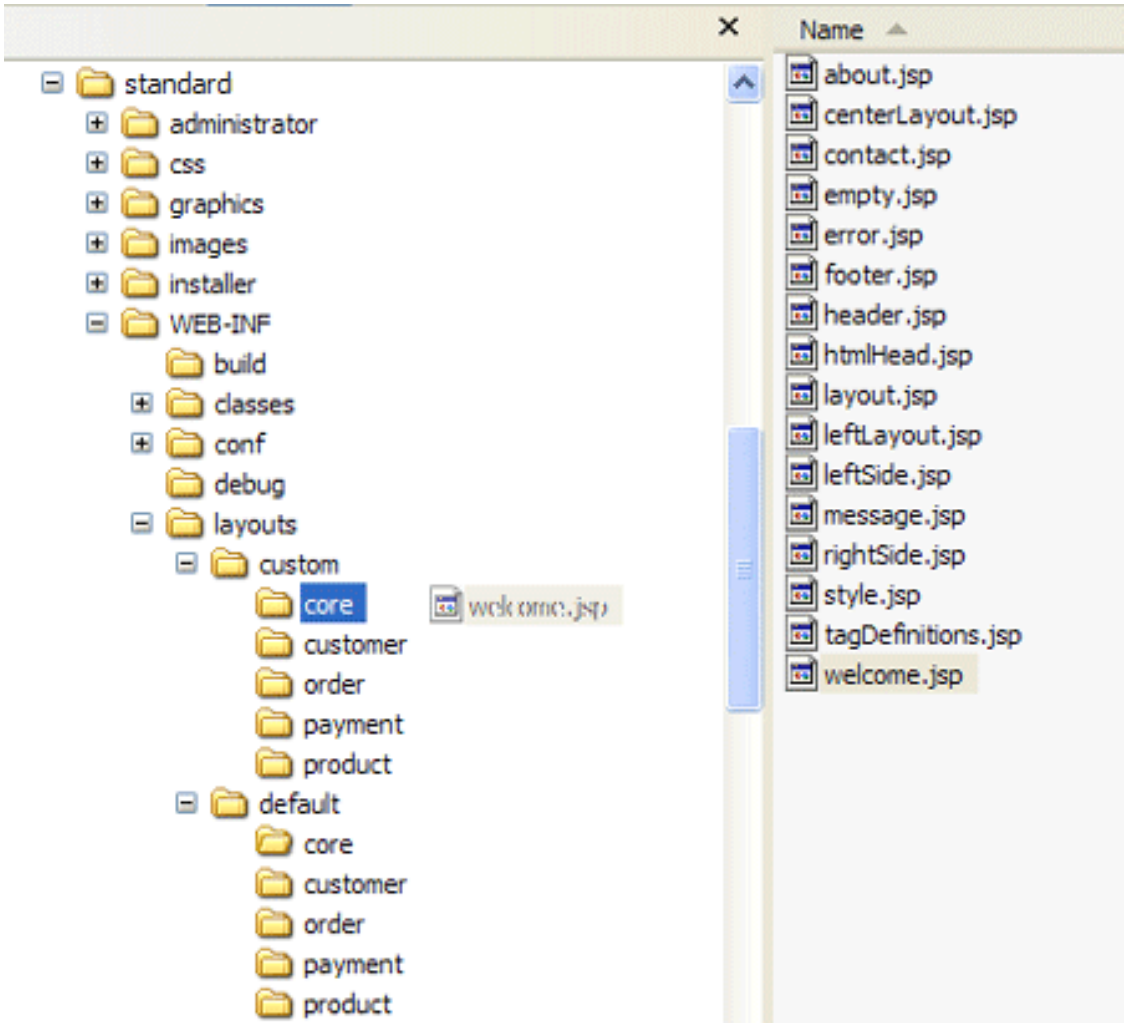
We strongly recommend setting up a separate installation of SoftSlate Commerce that you'll use only for development purposes. The SoftSlate Commerce license allows you to install the application on an unlimited number of development machines, so long as they are not accessible by the public through the Internet. It also allows you to install the application on a publicly-accessible *staging* server, which can be used to test your changes before going live.

Tip

An inexpensive way to set up a development installation of SoftSlate Commerce is to install Apache Tomcat [http://jakarta.apache.org/tomcat] and MySQL [http://www.mysql.com] on whatever machine you'll be using for development. Both Tomcat and MySQL are free, open-source products that work well with SoftSlate Commerce.

Custom and Default JSP Templates

The JSP templates that come with the application are all located in the `/WEB-INF/layouts/default` directory. Inside there, they are organized into subdirectories representing the major functional areas of your store: `core`, `product`, `order`, `customer`, and `payment`.



Copying a template from the `default` directory to the `custom` directory

Custom templates that you create will go in the `/WEB-INF/layouts/custom` directory, which contains the same five subdirectories. If a JSP template exists in the `custom` directory, the store will automatically detect and use it in place of the corresponding template under the `default` directory. This system allows you to create custom templates for just those areas of the store you need to modify, while other areas can be left untouched so that they use the default templates.

It's important not to modify the templates in the `/WEB-INF/layouts/default` directory. These files may be overridden when you install a future upgrade of SoftSlate Commerce. Instead, please place a copy of the template you wish to change into the corresponding subdirectory under `/WEB-INF/layouts/custom`. The store will automatically detect the new file and begin using it immediately. You can then feel free to make whatever changes you wish to the copied file.

Important

Do not modify the templates in the `/WEB-INF/layouts/default` directory. They may be overridden with future upgrades of SoftSlate Commerce. Instead, copy the template you wish to change into the corresponding subdirectory under `/WEB-INF/layouts/custom`.

Customizing Your First Screen

One of the first changes you're likely to make is to customize the Welcome Screen to display your own message to your visitors, featured products, or other information. Follow these steps to create a custom template for the Welcome Screen in place of the default template.

Example 11.1. Customizing the Welcome Screen

1. Refer to the Outline of the Default Welcome Screen. As you can see from the outline, the body of the Welcome Screen where the default heading and message appears, is produced by the `core/welcome.jsp` file. Find this file under the `/WEB-INF/layouts/default/core` directory.

2. You could make changes directly in this file, but you should *not*. The files inside the `/WEB-INF/layouts/default` directory may be overwritten when you install a future upgrade of SoftSlate Commerce. Instead, copy the `welcome.jsp` file into the `/WEB-INF/layouts/custom/core` directory.

3. Now you can safely make whatever changes you need to the copied `welcome.jsp` file under the `custom` directory. Open up the `welcome.jsp` you copied to the `custom` directory. It should look something like this:

```
<%@ include file="/WEB-INF/layouts/default/core/tagDefinitions.jsp" %>
<!-- Start welcome.jsp -->
<table width="100%" border="0" cellspacing="0" cellpadding="4">
        <tr>
                <td class="welcomeHeading">
                        <bean:message key="welcome.heading"/>
                </td>
        </tr>
        <tr>
                <td class="welcomeMessage">
                        <bean:message key="welcome.message"/>
                </td>
        </tr>
</table>
<!-- End welcome.jsp -->
```

4. Now replace the default welcome message with your own unique message:

```
<%@ include file="/WEB-INF/layouts/default/core/tagDefinitions.jsp" %>
<!-- Start welcome.jsp -->
<table width="100%" border="0" cellspacing="0" cellpadding="4">
        <tr>
                <td class="welcomeHeading">
                        <bean:message key="welcome.heading"/>
                </td>
        </tr>
        <tr>
                <td class="welcomeMessage">
                        Hello customers! So glad you decided to stop by!
                </td>
        </tr>
</table>
<!-- End welcome.jsp -->
```

5. Save the file. If necessary, upload the new file to the server you have running SoftSlate Commerce. Then navigate in your browser to the Welcome Screen [../../Welcome.do] (refresh the page if your browser is already there). You should see your changes on the screen.

Congratulations! You've just made your first customization to your store's screens.

Note

The <bean:message key="welcome.message"/> tag that was replaced in the above example is a tag that pulls in text from the `application.properties` files under the `/WEB-INF/classes/resources` directory, which are used to assist with internationalization and translations. For more information, refer to the SoftSlate Commerce and Internationalization section.

Customizing the Core Layout Files

Customizing a particular screen to add your own HTML, text, or graphics is one thing. But to implement a complete design, you will no doubt need to customize the core layout files that control the layout of each of the various screens in SoftSlate Commerce. These files provide the overall structure of each of the screen, while the other files in the application fill in the "guts" of the screens' content.

To customize one of the core layout files, use the same method as customizing any other JSP template: copy the default template from the `default` directory to the `custom` directory, and then make whatever changes you need to.

Note

You may find your design calls for using the Center Layout on all of the screens that by default use the store's Left Layout, or vice-versa. A simple technique for making this happen is to copy both `centerLayout.jsp` and `leftLayout.jsp` into the custom directory. Then, if you want to use the Center Layout in place of the Left Layout, you can simply replace the contents of `leftLayout.jsp` with the following:

```
<jsp:include page="centerLayout.jsp"/>
```

There are three layouts employed by the application: Center Layout, Left Layout, and Basic Layout. These layouts are produced by `core/centerLayout.jsp`, `core/leftLayout.jsp`, and `core/layout.jsp` respectively. Use the following table to find out which layout file corresponds to which screens.

Table 11.1. SoftSlate Commerce's Core Layouts and Their Screens

Layout	Template	Description	Screens
Center Layout	`core/centerLayout.jsp`	Features a column on both the left-hand and right-hand side of the screen. By default the left-hand side displays the store's category tree, a search box, and the user's cart. The right-hand column displays the store's Built-In Categories.	Welcome Screen
			About Screen
			Contact Screen
			Search Screen
			Cart Item Edit Screen
Left Layout	`core/leftLayout.jsp`	Features a column on the left-hand of the screen. By default the left-hand side displays the store's category tree, a search box, and the user's cart.	Category Screen
			Product Screen
			Product List Screen
			Search Results Screen
			Cart Screen
			Login Screen
			Register Screen
			Lost Password Screen
			Checkout Invite Login Screen
			Checkout Force Login Screen
			Checkout Invite Register Screen
Basic Layout	`core/layout.jsp`	Simply displays the screen's content in the middle of the page, between the header and footer. Used by default for the store's checkout screens and the customer account screens.	Error Screen
			Order Form Screen
			Account Addresses Screen
			Account Password Screen
			Account History Screen
			Account History Details Screen
			Checkout Addresses Screen
			Checkout Payment Screen
			Checkout Combo Screen
			Checkout ConfirmScreen
			Checkout Thank You Screen

Example 11.2. Customizing the Layout of the Checkout Screens

1. If you need to change the layout of the all screens in the checkout process, refer to the above table. As you can see the layout to customize is the Basic Layout, which is controlled by the `layout.jsp` file.

2. Make a copy of the `layout.jsp` file that appears in the `/WEB-INF/layouts/default/core` directory and place it in the `/WEB-INF/layouts/custom/core` directory. (As with the other JSP templates, you should place copies in the `custom` directory because the files in the `default` directory may be overwritten with a future upgrade.)

3. Now, feel free to make whatever changes you need to to the copied file. Upload the file into the SoftSlate Commerce application, and you're changes should take effect immediately, on all the screens that correspond to the layout file.

Customizing Screen Sections: Working with the Tiles Framework

SoftSlate Commerce uses the Tiles Framework [http://struts.apache.org/userGuide/dev_tiles.html] to organize its JSP templates. Throughout SoftSlate Commerce's JSP templates, you'll see tags like this, which invoke a Tiles definition:

```
<tiles:insert attribute="header"/>
```

If you aren't familiar with Tiles, you don't have to be to customize the templates. However, some understanding of Tiles and the Tiles definitions will help you find your way around and give you opportunities to customize your store in simpler ways.

"Tiles" can be thought of as different blocks on a page displayed in the browser. Which templates produce the different blocks are defined in Tiles definition files, which are xml files located in the `/WEB-INF/conf` folder of SoftSlate Commerce. For example, the Tiles definition for the Center Layout and for the Welcome Screen is found in `/WEB-INF/conf/core/tiles-defs.xml`. They look like this:

```
<definition name="core.baseCenterLayout"
        path="/WEB-INF/layouts/default/core/centerLayout.jsp"
        controllerUrl="/LayoutAction.do">
        <put name="htmlHead" value="/WEB-INF/layouts/default/core/htmlHead.jsp" />
        <put name="header" value="/WEB-INF/layouts/default/core/header.jsp" />
        <put name="error" value="/WEB-INF/layouts/default/core/error.jsp" />
        <put name="message" value="/WEB-INF/layouts/default/core/message.jsp" />
        <put name="leftSide" value="core.leftSide" />
        <put name="rightSide" value="core.rightSide" />
        <put name="subMenu" value="customer.accountMenu" />
        <put name="body" value="/WEB-INF/layouts/default/core/empty.jsp" />
        <put name="footer" value="/WEB-INF/layouts/default/core/footer.jsp" />
</definition>

<definition name="core.welcome" extends="core.baseCenterLayout">
        <put name="pageTitleKey" value="page.welcome"/>
        <put name="body" value="/WEB-INF/layouts/default/core/welcome.jsp" />
</definition>
```

These two definitions tell the Tiles Framework which JSP templates to use when displaying the Welcome Screen. They indicate that the `/WEB-INF/layouts/default/core/centerLayout.jsp` file is used to control the overall layout of the screen. Within that file, the definitions tell Tiles to include `/WEB-INF/layouts/default/core/header.jsp` where the `<tiles:insert attribute="header"/>` tag appears. Similarly, the `core.welcome` definition tells Tiles to include `/WEB-INF/layouts/default/core/welcome.jsp` when it comes across the `<tiles:insert attribute="body"/>` tag. In this way, the Welcome Screen is constructed by the Tiles Framework, using the JSP templates defined in the Tiles definition file.

Note

SoftSlate Commerce extends the Tiles framework to first look inside the `custom` directory each time a given template is called for, before using the template found in the `default` directory. This is how a definition that calls for `/WEB-INF/layouts/default/core/welcome.jsp` will include `/WEB-INF/layouts/custom/core/welcome.jsp` instead, if it exists.

There are some situations where the simplest way to customize a section of a screen is to change the Tiles definition for the screen rather than the JSP templates themselves. As an example, let's say you want your store's Welcome Screen to have a different header than the rest of your store's screens. By default, all of the store's screens include `/WEB-INF/layouts/default/core/header.jsp` to render the header. But you want the Welcome Screen to use a different template (maybe because you want to enlarge your logo, or include a different set of navigation links).

Example 11.3. Replacing the Welcome Screen's Header

1. The file `/WEB-INF/conf/core/tiles-defs-custom.xml` is included in SoftSlate Commerce to help with creating custom Tiles definitions. A custom definition will let us identify a special header template file for the Welcome Screen. Find the `/WEB-INF/conf/core/tiles-defs-custom.xml` file and open it in a text editor.

2. Add the following custom definition to `/WEB-INF/conf/core/tiles-defs-custom.xml`, inside the `<tiles-definitions>` tag:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE tiles-definitions PUBLIC
        "-//Apache Software Foundation//DTD Tiles Configuration 1.1//EN"
        "http://jakarta.apache.org/struts/dtds/tiles-config_1_1.dtd">


<!-- Tiles definitions for custom pages. -->

<tiles-definitions>

        <definition name="core.welcome" extends="core.baseCenterLayout">
                <put name="pageTitleKey" value="page.welcome"/>
            <put name="body" value="/WEB-INF/layouts/default/core/welcome.jsp"
            <put name="header"
                value="/WEB-INF/layouts/default/core/homeHeader.jsp" />
        </definition>
```

```
</tiles-definitions>
```

3. What this will do is override the default definition for the Welcome Screen. Where the default definition calls for `/WEB-INF/layouts/default/core/header.jsp` to be used for the header, this change will call up `/WEB-INF/layouts/default/core/homeHeader.jsp` instead, for just Welcome Screen. Save the `/WEB-INF/conf/core/tiles-defs-custom.xml` file and if necessary upload it to your installation of SoftSlate Commerce.

4. Of course, you now need to create the `homeHeader.jsp` file itself, with the changes you want to include on the Welcome Screen's header. Since this file is new and not part of the SoftSlate Commerce distribution, you can safely place it in the `default` directory. Doing so will let all of your custom layouts use the file as a default (which will come in handy if you end up creating Multiple Custom Layouts, described below). Create the `homeHeader.jsp` file now and if necessary upload it into the `/WEB-INF/layouts/default/core` directory.

5. Finally, whenver you make a change to a Tiles definition file, you must reload the SoftSlate Commerce application in your servlet container for the change to take effect.

6. After reloading the application, refresh the Welcome Screen in your browser. You should see the content of the `homeHeader.jsp` file appearing in place of the regular header.

Caution

The `/WEB-INF/conf/core/tiles-defs-custom.xml` file is configured to be read in by SoftSlate Commerce after all of the other Tiles definition files, so it can be used to override the existing definitions (as described above). Please use this technique to change a definition, and avoid modifying the other definition files. These other files may be modified or replaced when you install a future upgrade of SoftSlate Commerce.

Important

The three layout definitions, `core.baseLayout`, `core.baseLeftLayout`, and `core.baseCenterLayout` cannot be overridden using the `/WEB-INF/conf/core/tiles-defs-custom.xml` file. These definitions are extended by other definitions before the `/WEB-INF/conf/core/tiles-defs-custom.xml` file is read. The three definitions are exactly the same, so they can be interchanged. Consider using a JSP include command to include additional files from one of these core layout files.

Note

There is much more information about Tiles available in books and online. Refer to the Struts Web site [http://struts.apache.org/userGuide/dev_tiles.html] for more details.

Chapter 12. Multiple Custom Layouts

We've seen examples of how to create a custom layout by copying the default JSP templates that come with the application into the `custom` directory and then making modifications. SoftSlate Commerce lets you take things one step further and create multiple layouts consisting of multiple sets of customized JSP templates.

In short, you can create as many directories under the `/WEB-INF/layouts` directory as you wish *next to* the `custom` and `default` directories, each one representing a completely separate look and feel for your store. Which layout a given visitor to your store sees can be controlled by a simple URL parameter (named *`layout`*). If the *`layout`* URL parameter matches the name of a directory under `/WEB-INF/layouts`, the store will use the templates in that directory, when they exist, to render the screens. (As with the `custom` layout, if a template does not exist, the store will use the corresponding template in the `default` directory.)

Adding an Additional Custom Layout

Example 12.1. Creating an Additional Custom Layout for Visitors Coming from an Affiliate

Let's say we want to create a special look and feel for people who visit the store from a link on a particular affiliate's site. (Perhaps we want to include the affiliate's logo in the store's header, or change the text or colors around a bit. Any customization on any or all screens is possible.)

1. The first thing to do is give a name to your new layout. For our purposes (and for lack of a more original name) let's call our new layout the "special" layout (a one-word term is best).

2. This name serves as both the value of the *`layout`* URL parameter for visitors coming to the site, and as the name of the directory under `/WEB-INF/layouts` that holds the new layout's customized templates. So, go ahead and create a subdirectory under `/WEB-INF/layouts` named `special`. While you're at it, within that directory create the same subdirectories found under the `custom` and `default` directories: `core`, `product`, `order`, `customer`, and `payment`.

3. Tell your affiliate to add the *`layout`* URL parameter, with a value of "special", to any links that take visitors from their site to your store. For example, the link to your Welcome Screen would look something lie this: `http://www.storesdomainname.com/softslate/Welcome.do?layout=special`. When SoftSlate Commerce sees that parameter on the URL, it knows to use the "special" layout throughout the user's session.

4. At this point you can follow the same techniques for creating a custom layout described in the Creating a Custom Layout chapter to create the new "special" layout. Copy files you want to customize from the `default` directory to the `special` directory. If the visitor is assigned to the "special" layout, the store will first look in the `special` directory to include each template. If it does not exist there, it will then include the corresponding template in the `default` directory.

Important

An important setting for controlling multiple custom layouts is the "Default Layout" setting in the Settings -> Display [../../administrator/SettingsDisplay.do] screen of the administrator. This setting tells SoftSlate Commerce which layout to give users by default, if the layout is not specified on the URL. For

complete details, refer to the section on the Default Layout Setting

Chapter 13. More Customization Examples

SoftSlate Commerce's Built-In Categories

SoftSlate Commerce has four special Built-In Categories that you can use to display special products throughout the store. If you've created any of these categories in your store, SoftSlate Commerce will automatically load their products into memory, so you can use them to display special products on any screen. By default, the categories will be displayed on the right-hand side of the Welcome Screen.

The four Built-In Categories that SoftSlate Commerce looks for have the following category codes:

- _features
- _popular
- _favorites
- _new

These categories can be used, respectively, for featured products, most popular or bestselling products, favorites, and new products. (You can actually name the categories whatever you like, so long as the category code is one of the four codes.

As an example, let's add the "_features" category to the store, and then add some products to it.

Example 13.1. Adding a Built-In Category

1. To add a Built-In Category to your store, log into the administrator and navigate to the Products -> Categories [../../administrator/Category.do] screen. Click the "Add New Record" button on the right hand side of the screen to go to the add category form.
2. Add a category whose name is "Featured Products" and whose code is "_features". Make sure the code is entered correctly, including the underscore "_" at the beginning. If you enter it incorrectly, SoftSlate Commerce will not think it's one of the Built-In Categories and won't load its products into memory. For now leave the rest of the text fields empty. At the bottom of the form, for the "On Category Tree" field, select no, so the category does not appear on the store's category tree. Be sure to leave the "Active" field marked yes. Click "Add New Record" when you're done.
3. Now that you've added the category, let's assign some products to it. We'll assume you have already added one or more products, but if you haven't, refer to the Adding Your First Product section.
4. Navigate to Products -> Categories [../../administrator/Category.do]. You'll see the new "_features" category in the table in the center of the screen. Click the "Products" link next to it to view the products assigned and not assigned to the category. Click the "On" button next to "Edit Mode", check the checkboxes under the "Assigned" column next to the product you want to assign to the category, and click "Update/Delete" to save the change.

Chapter 14. SoftSlate Commerce's Screens

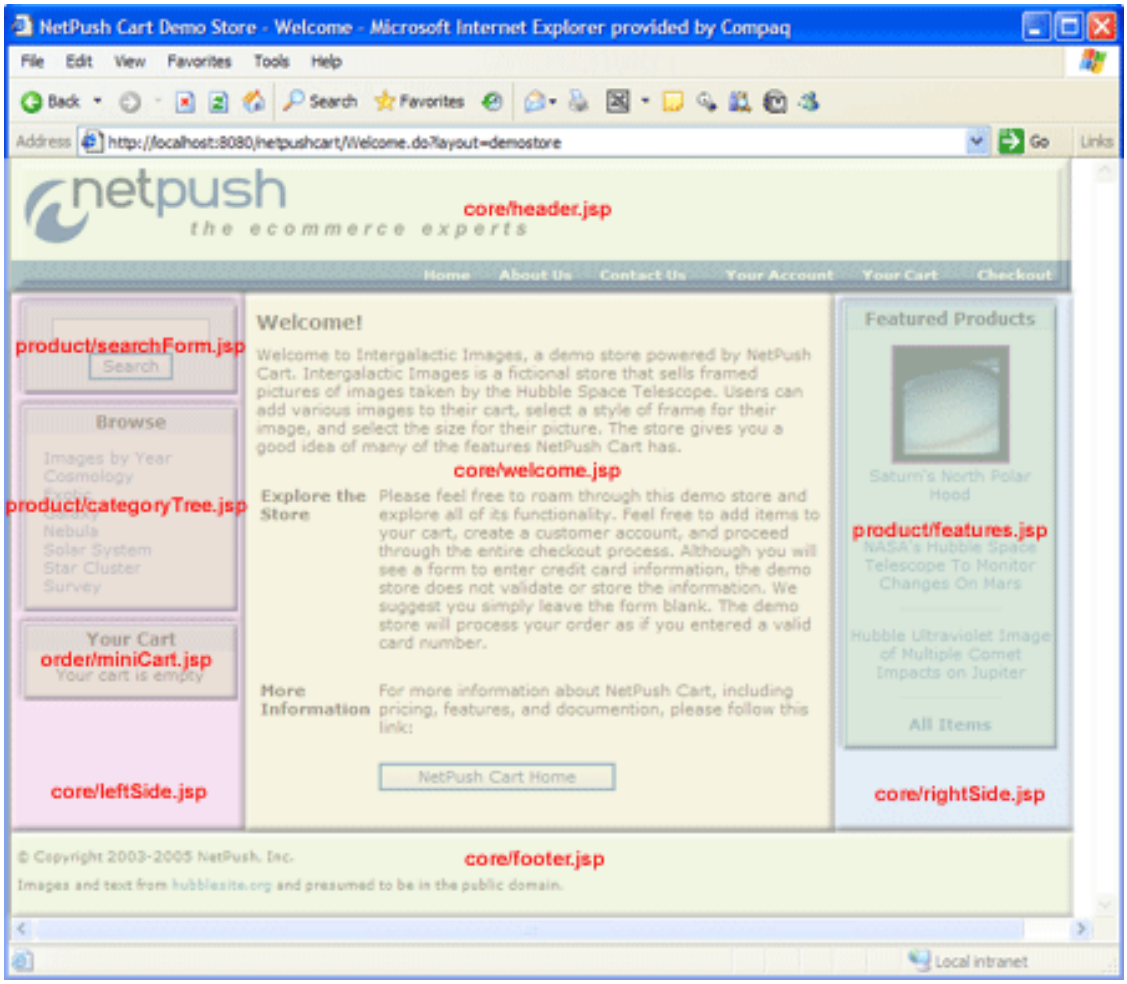
The following sections describe each of the screens in the customer interface for SoftSlate Commerce. When customizing or extending the application, it's essential to be familiar with the various screens and the roles they play. Use the following tables and sections as a reference when making changes.

Welcome Screen

By default the Welcome Screen uses `core/centerLayout.jsp` to control the overall layout of the page. Below is an outline of the Welcome Screen, with the JSP templates that it includes highlighted and labelled. You can use this outline to identify the areas of the Center Layout and of the Welcome Screen that you need to customize.

Table 14.1. Welcome Screen Reference

Path	Welcome.do [../../Welcome.do]
Required URL Parameters	None
Description	The opening screen of the store, and the target of the "Home" link in the default header.
Layout File	`core/centerLayout.jsp`
Key Included Files	`core/welcome.jsp`



Outline of the default Welcome Screen using the Center Layout. Larger Image [images/welcomeScreenLarge.gif].

Contact Screen

Table 14.2. Contact Screen Reference

Path	contactPage.do [../../contactPage.do]
Required URL Parameters	None
Description	A page intended to display the store's contact information. The target of the "Contact" link in the default header.
Layout File	`core/centerLayout.jsp`
Key Included Files	`core/contact.jsp`

About Screen

Table 14.3. About Screen Reference

Path	aboutPage.do [../../aboutPage.do]
Required URL Parameters	None
Description	A page intended to display general information about the store. The target of the "About" link in the default header.
Layout File	`core/centerLayout.jsp`
Key Included Files	`core/about.jsp`

Search Screen

Table 14.4. Search Screen Reference

Path	searchPage.do [../../searchPage.do]
Required URL Parameters	None
Description	Displays the search box form on its own page.
Layout File	`core/centerLayout.jsp`
Key Included Files	`product/search.jsp`

Cart Item Edit Screen

Table 14.5. Cart Item Edit Screen Reference

Path	CartItemEditForm.do
Required URL Parameters	*`orderItemID`*
Description	Displays information about one of the items in the user's cart, with buttons to edit or delete it.
Layout File	`core/centerLayout.jsp`
Key Included Files	`order/cartItemEdit.jsp`

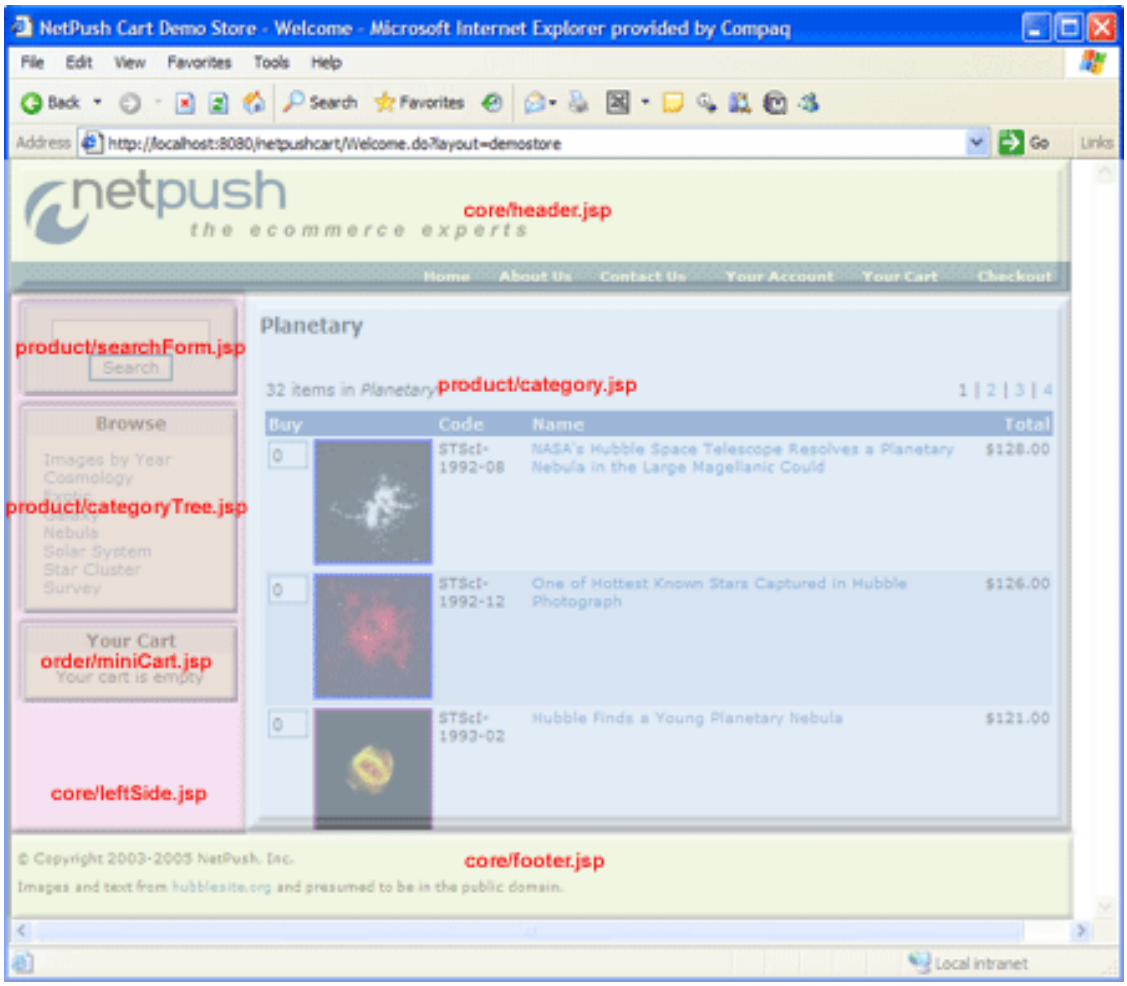
Category Screen

By default the Category Screen uses `core/leftLayout.jsp` to control the overall layout of the page. Below is an outline of the Category Screen, with the JSP templates that it includes highlighted and labelled. You can use this outline to identify the areas of the Left Layout and of the Category Screen that you need to customize.

Table 14.6. Category Screen Reference

Path	Category.do
Required URL Parameters	*`code`*

Description	Displays all the information related to a given category, including a list of its subcategories, if any, and a list of its products.
Layout File	`core/leftLayout.jsp`
Key Included Files	`product/category.jsp`
	`product/subcategories.jsp`
	`product/productList.jsp`



Outline of a default Category Screen using the Left Layout. Larger Image [images/categoryScreenLarge.gif].

Product Screen

Below is an outline of the Product Screen, with the JSP templates that it includes highlighted and labelled. You can use this outline to identify the areas of the Left Layout and of the Product Screen that you need to customize.

Table 14.7. Product Screen Reference

Path	Product.do

Required URL Parameters	*code*
Description	Displays all the information related to a given product. Provides an "Add To Cart" button to add the product to the cart. Also includes the product's attributes and options.
Layout File	`core/leftLayout.jsp`
Key Included Files	`product/product.jsp`
	`product/attributesAndOptions.jsp`

Outline of a default Product screen using the Left Layout. Larger Image [images/productScreenLarge.gif].

Product List Screen

Table 14.8. Product List Screen Reference

Path	ProductList.do [../../ProductList.do]
Required URL Parameters	None
Description	Displays a list of *all* the active products in the store. Useful for stores with a small number of products.
Layout File	`core/leftLayout.jsp`
Key Included Files	`product/productListLayout.jsp`
	`product/productList.jsp`
	`product/pagination.jsp`

Search Results Screen

Table 14.9. Search Results Screen Reference

Path	Search.do
Required URL Parameters	*`searchString`*
Description	Displays a list of the active products matching a search string provided by the user in the search box.
Layout File	`core/leftLayout.jsp`
Key Included Files	`product/searchResultsLayout.jsp`
	`product/productList.jsp`
	`product/pagination.jsp`

Cart Screen

Table 14.10. Cart Screen Reference

Path	Cart.do [../../Cart.do]
Required URL Parameters	None
Description	Displays the current user's cart and its contents.
Layout File	`core/leftLayout.jsp`
Key Included Files	`order/cart.jsp`

Account Login Screen

Table 14.11. Account Login Screen Reference

Path	AccountLogin.do [../../AccountLogin.do]
Required URL Parameters	None
Description	Displays a form for a customer to log in to his or her account, and links to create a new account or retrieve a lost password.

Layout File	`core/leftLayout.jsp`
Key Included Files	`customer/loginLayout.jsp`
	`customer/login.jsp`
	`customer/lostPasswordLink.jsp`
	`customer/registerLink.jsp`

Register Screen

Table 14.12. Register Screen Reference

Path	RegisterForm.do [../../RegisterForm.do]
Required URL Parameters	None
Description	Displays a form for a user to create a new customer account.
Layout File	`core/leftLayout.jsp`
Key Included Files	`customer/registerLayout.jsp`
	`customer/register.jsp`

Lost Password Screen

Table 14.13. Lost Password Screen Reference

Path	LostPasswordForm.do [../../LostPasswordForm.do]
Required URL Parameters	None
Description	Displays a form for a customer to have a lost password sent to him or her.
Layout File	`core/leftLayout.jsp`
Key Included Files	`customer/lostPassword.jsp`

Checkout Invite Login Screen

Table 14.14. Checkout Invite Login Screen Reference

Path	CheckoutInviteLoginForm.do [../../CheckoutInviteLoginForm.do]
Required URL Parameters	None, but user must have an appropriate session status to access the checkout screens.
Description	Displays a form for a customer to log in to his or her account, and links to create a new account, retrieve a lost password, or continue through checkout without an account.
Layout File	`core/leftLayout.jsp`

Key Included Files	`order/inviteLoginLayout.jsp`
	`customer/login.jsp`
	`customer/lostPasswordLink.jsp`
	`customer/registerLink.jsp`
	`order/declineLogin.jsp`

Checkout Force Login Screen

Table 14.15. Checkout Force Login Screen Reference

Path	CheckoutForceLoginForm.do [../../CheckoutForceLoginForm.do]
Required URL Parameters	None, but user must have an appropriate session status to access the checkout screens.
Description	Displays a form forcing the customer to log in to his or her account, or to create a new account before proceeding through checkout. Also displays a link to retrieve a lost password.
Layout File	`core/leftLayout.jsp`
Key Included Files	`order/forceLoginLayout.jsp`
	`customer/login.jsp`
	`customer/lostPasswordLink.jsp`
	`customer/registerLink.jsp`

Checkout Invite Register Screen

Table 14.16. Checkout Invite Register Screen Reference

Path	CheckoutInviteRegisterForm.do [../../CheckoutInviteRegisterForm.do]
Required URL Parameters	None, but user must have an appropriate session status to access the checkout screens.
Description	Displays a form for a user to create a new customer account.
Layout File	`core/leftLayout.jsp`
Key Included Files	`order/inviteRegisterLayout.jsp`
	`customer/register.jsp`

Error Screen

Table 14.17. Error Screen Reference

Path	Error.do [../../Error.do]
Required URL Parameters	None
Description	Displays any unexpected errors generated by the application, including any exceptions thrown from the request processing.
Layout File	`core/layout.jsp`
Key Included Files	`core/error.jsp`

Order Form Screen

Table 14.18. Order Form Screen Reference

Path	OrderForm.do [../../OrderForm.do]
Required URL Parameters	None
Description	Displays a standalone order form where the user can select which products to buy and enter in address, shipping and payment information in one step.
Layout File	`core/layout.jsp`
Key Included Files	`order/orderFormLayout.jsp`
	`order/orderProductList.jsp`
	`order/billingFormGuts.jsp`
	`order/deliveryFormGuts.jsp`
	`order/shippingFormGuts.jsp`
	`order/paymentFormGuts.jsp`

Account Addresses Screen

Table 14.19. Account Addresses Screen Reference

Path	AccountAddresses.do [../../AccountAddresses.do]
Required URL Parameters	None, but customer must be logged in.
Description	Displays a form for the customer to enter or modify his or her billing and delivery addresses.
Layout File	`core/layout.jsp`
Key Included Files	`customer/addresses.jsp`
	`order/billingFormGuts.jsp`
	`order/deliveryFormGuts.jsp`

Account Password Screen

Table 14.20. Account Password Screen Reference

Path	AccountPassword.do [../../AccountPassword.do]
Required URL Parameters	None, but customer must be logged in.
Description	Displays a form for the customer to modify his or her user name and password.
Layout File	`core/layout.jsp`
Key Included Files	`customer/password.jsp`

Account History Screen

Table 14.21. Account History Screen Reference

Path	AccountHistory.do [../../AccountHistory.do]
Required URL Parameters	None, but customer must be logged in.
Description	Displays a list of the previous orders placed by the customer.
Layout File	`core/layout.jsp`
Key Included Files	`customer/historyLayout.jsp`
	`customer/historyList.jsp`

Account History Details Screen

Table 14.22. Account History Details Screen Reference

Path	OrderDetails.do
Required URL Parameters	*`orderNumber`*
Description	Displays the details of a particular order the customer has placed previously.
Layout File	`core/layout.jsp`
Key Included Files	`customer/historyDetailsLayout.jsp`
	`customer/orderDetails.jsp`

Checkout Addresses Screen

Table 14.23. Checkout Addresses Screen Reference

Path	CheckoutAddresses.do [../../CheckoutAddresses.do]
Required URL Parameters	None, but user must have an appropriate session status to access the checkout screens.
Description	Displays a form for the user to enter in billing and delivery address information, and to select a shipping option.
Layout File	`core/layout.jsp`

Key Included Files	`order/checkoutAddressesLayout.jsp`
	`order/checkoutBreadcrumbs.jsp`
	`order/billingFormGuts.jsp`
	`order/deliveryFormGuts.jsp`
	`order/shippingFormGuts.jsp`

Checkout Payment Screen

Table 14.24. Checkout Payment Screen Reference

Path	CheckoutPayment.do [../../CheckoutPayment.do]
Required URL Parameters	None, but user must have an appropriate session status to access the checkout screens.
Description	Displays a form for the user to enter in payment information.
Layout File	`core/layout.jsp`
Key Included Files	`order/paymentLayout.jsp`
	`order/checkoutBreadcrumbs.jsp`
	`order/reviewOrder.jsp`
	`order/paymentForms.jsp`

Checkout Combo Screen

Table 14.25. Checkout Combo Screen Reference

Path	CheckoutCombo.do [../../CheckoutCombo.do]
Required URL Parameters	None, but user must have an appropriate session status to access the checkout screens.
Description	Displays a combined form for the user to enter in billing and delivery address information, to select a shipping option, and to enter in payment information.
Layout File	`core/layout.jsp`
Key Included Files	`order/comboFormLayout.jsp`
	`order/checkoutBreadcrumbs.jsp`
	`order/billingFormGuts.jsp`
	`order/deliveryFormGuts.jsp`
	`order/shippingFormGuts.jsp`
	`order/paymentFormGuts.jsp`

Checkout Confirm Screen

Table 14.26. Checkout Confirm Screen Reference

Path	CheckoutConfirm.do [../../CheckoutConfirm.do]
Required URL Parameters	None, but user must have an appropriate session status to access the checkout screens.
Description	Displays a summary of the user's current order, and a button for the user to confirm and finalize it.
Layout File	`core/layout.jsp`
Key Included Files	`order/checkoutConfirmLayout.jsp`
	`order/checkoutBreadcrumbs.jsp`
	`order/reviewOrder.jsp`

Checkout Thank You Screen

Table 14.27. Checkout Thank You Screen Reference

Path	None - this screen is reached when the user has successfully finished the checkout process.
Required URL Parameters	User must have an appropriate session status to complete the checkout process.
Description	Displays an invoice of the order the user has just completed..
Layout File	`core/layout.jsp`
Key Included Files	`order/thankYou.jsp`
	`order/orderDetails.jsp`

Part III. Guide for Developers

Table of Contents

Chapter 15. How to Customize SoftSlate Commerce

7 Simple Rules for Making Customizations

For any customization you make to SoftSlate Commerce, at all times try to follow these rules. It will save you work and spare you from issues that may occur when upgrading to a future version.

1. **Custom Java Classes.**

   Override existing Java classes by subclassing the appropriate class from the `com.softslate.commerce` package with your own new class. (Requires application reload.) Then, change the setting under the Settings -> Components [../../administrator/SettingsComponents.do] screen of the Administrator (`npcSetting` database table) to tell the application to use your new class instead of the default one. Do not modify any of the existing classes as they may change with an upgrade.

   For more information, refer to SoftSlate Commerce's Java API documentation available at http://www.softslate.com.

2. **Custom SQL Queries.**

   Override existing SQL queries or create ones in the appropriate `sql-custom.properties` file found under the subdirectories of the `/WEB-INF/classes/resources` directory. (Requires application reload.) Since the `sql-custom.properties` files are the last files read into memory, they will override the queries defined in the other `sql*.properties` files. Do not modify any of the other `sql*.properties` files as they may change with an upgrade.

3. **Custom Struts Configurations.**

   Override existing Struts action mappings and form bean definitions, or create new ones, in `/WEB-INF/conf/core/struts-config-custom.xml`. (Requires application reload.) Since this is the last file read into memory, it will override the mappings and form beans defined in the other Struts XML files. Do not modify any of the other Struts XML files as they may change with an upgrade.

4. **Custom Tiles Definitions.**

   Override existing Tiles definitions or create new ones in `/WEB-INF/conf/core/tiles-defs-custom.xml`. (Requires application reload.) Since this is the last file read into memory, it will override the definitions defined in the other Tiles XML files. Do not modify any of the other Tiles definition XML files as they may change with an upgrade.

   For more information, refer to the section on Working with the Tiles Framework

5. **Custom JSP Templates.**

   Override existing JSP templates by copying them from the `/WEB-INF/layouts/default` directory over to the `/WEB-INF/layouts/custom` directory and modifying them there. Create any new JSP templates in the `custom` directory as well. Do not modify the templates in the `default` directory as they may change with an upgrade. (In a similar way, to customize templates used in the Administrator, copy templates from `/WEB-INF/templates/administrator/default` to `/`

`WEB-INF/templates/administrator/custom`.)

For more information, refer to the section on Creating a Custom Layout.

6. **Custom Application Messages.**

   Create new application messages in `/WEB-INF/classes/resources/core/application-custom.properties`. (Access the message using `<bean:message key="[your key]" bundle="/custom"/>` in a JSP page. Requires application reload.) Do not modify any of the other `application*.properties` files as they may change with an upgrade. (Note: it is not possible to override an existing application message in the `application-custom.properties` file, you can only create new messages.)

7. **Custom CSS Styles.**

   Override existing CSS styles or create new ones in `/css/styles-custom.css`. Do not modify `/css/styles.css` as it may change with an upgrade.

   For more information, refer to the section on Using CSS Stylesheets.