Sterling Web™

Implementation Guide

Release 9.0

Last updated in HF1

May 2010

Sterling Commerce
An IBM Company

Contents

Sterling Web: An Overview

Sterling Web™ enables enterprises to sell their products directly to customers through e-commerce Web sites. Customers can browse a product catalog, add products to their cart, verify the prices of products and their availability, view promotions and special offers, and place orders through e-commerce Web sites.

Sterling Web provides the following solutions:

- Profile Management: A user can manage user organizations, create and manage users, create and manage child users, only if the user has the necessary permissions.
- Catalog Management: A user can search for a product and view its details, verify the price and the availability of a product, and compare a product with other products before placing an order for the product.
- Cart and Order Management: A user can add products to a cart and place an order. A user can also search for an existing cart or order and view its details, approve or reject an order that is on hold, send e-mails about an order or a cart, change an order, and cancel an order. The mini cart functionality enables a user to get a bird's eye view of the products in a cart. A user can also create a new cart from an existing cart or a confirmed order.

Sterling Web Users

A user is a person who can perform certain functions depending on the role the user plays in the corresponding organization. A user group is a collection of users who perform similar functions. A user can perform a function only if the user group to which the user belongs has the permission to perform that function. For example, a user belonging to the BUYER-ADMIN user group can manage the details pertaining to a buyer organization, whereas a user belonging to the BUYER-USER user group can only manage the details pertaining to a given user.

Users accessing Sterling Web can be configured as globally unique or unique only within a storefront. Whether the users are globally unique is governed by the **Allow the display User ID to be used across all Enterprises** rule that is configured using the Applications Manager. By default, this rule is enabled, and Sterling Web users are configured as globally unique. For more information about how to configure this rule, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*. Based on the tasks performed by the user, certain business rules are referenced. These business rules are cached in the session of the logged in user. After the user logs out, these rules are removed from the cache.

The following table lists the user groups in Sterling Web as part of the factory setup, and the tasks they can perform based on the default permissions assigned to them:

User Group	Task
BUYER-ADMIN	Users belonging to this user group can manage the details of a buyer organization, create and manage the details of child organizations of the buyer organization, create and manage the details of buyer users, browse the corresponding catalog, place orders for products, and so on.

User Group	Task
BUYER-USER	Users belonging to this user group can manage their own user details, browse the corresponding catalog, place orders for products, and so on. By default, all the buyer users are assigned to the BUYER-USER group. **Note:** Sterling Web does not support users that are not associated with a buyer organization.
BUYER-APPROVER	Users belonging to this user group can approve orders that are on hold, pending approval. Users belonging to this group can also perform all the tasks that can be performed by a user belonging to the BUYER-USER group.
GUEST-USER	Users belonging to this user group can browse catalogs, search for products, add products to a cart, and so on. However, these users cannot place an order.

Note: It is recommended that you do not delete any of the user groups that have been provided as factory setup. However, you can create new user groups or modify the existing user groups. For more information about creating and modifying a user group, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Terminology

The following table describes some of the terms that are used frequently in the Sterling Web application:

Term	Description
Storefront	A storefront is an organization that has the role of both an Enterprise and a Seller, and has a theme associated with it.
Cart	A cart is the basic unit of e-commerce. Buyer users purchase products from an e-commerce Web site by adding them to a cart and placing an order. A cart is also called a draft order in the Sterling Selling and Fulfillment Suite.
Guest user	A guest user is a user who can access the Sterling Web application without logging in to the application. A guest user can browse a product catalog, view the prices, and verify the availability of products, but cannot place an order. A guest user is also called an anonymous user in the Sterling Selling and Fulfillment Suite.
Buyer organization	A buyer organization is an organization that purchases products from a storefront. A buyer organization is also known as a customer in the Sterling Selling and Fulfillment Suite.
Buyer user	A buyer user is a user belonging to a buyer organization who can purchase products from a storefront on behalf of the buyer organization. A buyer user is also called a customer contact in the Sterling Selling and Fulfillment Suite.
Buyer administrator	A buyer administrator is a buyer user with administrator privileges.
Product	A product is the most basic component of the catalog hierarchy. A product is also called an item in the Sterling Selling and Fulfillment Suite.

The following table describes a few item-specific terms used in the Sterling Web application.

Term	Description
Up-sell	Up-sell items refer to higher priced products or services suggested to a customer who is considering a purchase. It involves inducing a customer to purchase more expensive items, upgrades, or other add-ons in an attempt to make a more profitable sale. Example: Suggesting that customers opt for more RAM or a larger hard drive when servicing their computer.
Cross-sell	Cross-sell items refer to related products or services suggested to a customer who is considering buying something. It also involves selling an additional product or service to an existing customer. This is carried out either to increase the income derived from existing clients or to protect the relationship with the clients. Example: While buying tennis racquets, a customer may be offered a bag, balls, tennis lessons, or accessories.
Bundle	Bundle items are logically grouped items, with inventory being maintained for individual items. A typical example of a bundle is a video game set containing a pack of CDs.
Physical Kit	Physical kits are physically assembled items, with inventory being maintained for the parent item. Typical examples are a bouquet of flowers and a preconfigured computer.
Alternative	Alternative items are items suggested when inventory is not available for the original item. They refer to any association type configured as “replace”.
Complimentary	Complimentary items complement the item you are buying. They refer to any association type configured as “relate”. For example, when you buy a mobile phone, a car charger can be a complimentary item.

For an exhaustive list of the terms used in the Sterling Web application, refer to the *Sterling Selling and Fulfillment Suite Glossary*.

Before You Begin

This topic describes the tasks and configurations that have to be performed before you begin to implement the functionalities provided by Sterling Web™. It is assumed that the Sterling Web application has been installed and deployed before you start the implementation process.

Note: Ensure that your appserver is configured to require cookies.

Configurations to Be Performed in Applications Manager

To utilize the basic functionalities provided by Sterling Web, you must perform all of the following tasks in the Applications Manager:

- Create storefront: A storefront is an organization that has roles of an Enterprise and a Seller and has a theme associated with it. For more information about creating an organization, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- Create basic configuration for a storefront: A storefront must have certain configurations set up to utilize the basic functionality. To configure a storefront, you must create payment types, payment rules, currency details, and define sourcing and scheduling rules. For more information about creating payment types, payment rules, currency details, and defining sourcing and scheduling rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- Create buyer organization: A buyer organization is a business customer who purchases products from a storefront. For more information about creating a customer definition, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- Create buyer administrator: A buyer administrator is a user belonging to the buyer organization, who can manage the buyer organization's details and the other users of the buyer organization. A buyer administrator is created as a customer contact of the customer. For more information about defining customer contacts, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- Create guest user and subscribe this user to the GUEST-USER user group: A guest user is a user who can access the Sterling Web application without logging in. A guest user's ID and Password is guest_<*Organization Code of the storefront*>. For example, the User ID and Password of a guest user for storefront XYZ with Organization Code ABC be guest_ABC. For more information about creating users, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- Generate catalog search index: To generate the catalog search index, you first configure the Catalog Index Build transaction, run the agent server, and then trigger the agent configured for the transaction. The Catalog Index Build transaction builds the catalog index file that is used during catalog search. This index file ensures that product searches are accomplished with fast response time. For more information about defining transactions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide.*

  Notes:

  - Ensure that the `searchindex.rootDirectory` property is set appropriately to point to the directory in which the generated search index files should be placed.
  - Ensure that you perform the steps described in "Configurations to Be Performed in Business Center" before generating a catalog search index.

- Define currency conversions: If a storefront wants to support multiple currencies, appropriate currency conversions must be defined for that storefront. For more information about defining currency conversions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- Define carrier services: The service a carrier provides for the delivery of an order is known as carrier service. For more information about defining carrier services, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- Use display user ID across enterprises: A user can determine whether a Sterling Web user is globally unique or unique only within a particular storefront. For more information about setting the display user ID to be used across all enterprises, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- Confirm resetting the password: A storefront can enable a user to reset the password only after a confirmation e-mail is sent to the user. To enable a user to reset the password only after a confirmation e-mail is sent, a storefront must configure the appropriate password policy for the user's organization. For more information about configuring a rule to send a confirmation to users on password reset and setting the password policy, refer to the *Selling and Fulfillment Foundation: Password Policy Management Guide*.
- Set order modification rules: A user can perform modifications to an order based on the status of the order. For more information about order modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- Validate item: A user can validate whether a product is valid for a cart when the product is added to a cart. For more information about Validate Item rule, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- Validate item during order creation or modification: A user can validate whether a product is valid for an order during the checkout process. For more information about validating item during creation/modification, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- Synchronize dates: A user can synchronize the requested dates and the expected ship dates for the order, order header, order line, and order line schedules. The requested dates synchronize with the requested ship dates, requested delivery dates, and cancel dates in the order line or header. The expected dates synchronize with the order schedules. For more information about how to synchronize dates between master order dates and dates on order line and schedules, refer to the *Sterling Distributed Order Management: Configuration Guide*.

  Note: If synchronization is not enabled, the synchronization between dates is not possible.

Configurations to Be Performed in Business Center

To utilize the functionalities provided by Sterling Web to the basic extent, you must perform all of the following tasks in Business Center before you start using the Sterling Web application:

- Create catalog and category: You must create catalogs that list items and their prices. There should be only one active selling catalog at a particular time. Category provides a means to organize items into different groups. After you create a category for a catalog, you can create subcategories, add items, and assign attributes to the category. For more information about creating catalogs and categories, refer to the *Business Center Item Administration Guide.*
- Create products and define details: You must create the products, that make up the storefront's product catalog, and define their details. For more information about creating products, refer to the *Business Center Item Administration Guide.*
- Define customer entitlements: Customer entitlements enable a storefront to define the products its customers can purchase. For example, a storefront may restrict a customer from buying products under the Apparel category, but allow the customer to buy products under the Electronics category. For more information about how to create an entitlement and associate customers to that entitlement, refer to the *Business Center Item Administration Guide*.
- Define price lists: A price list is a list of prices defined for a set of products. A price list is applicable to customers or an enterprise. For more information about defining price lists, refer to the *Business Center Pricing Administration Guide*.
- Configure assets: Assets are electronic media files that are associated with business objects. They can be used to display catalog advertisements, promotions, or special offers in the hot spot regions in the Home page. They can also be associated with products and categories. For more information about defining assets, refer to the *Business Center Item Administration Guide*.
- Define attributes: Attributes are characteristics that define an item. For more information about defining attributes, refer to the *Business Center Item Administration Guide*.

Configurations to Be Performed in Channel Applications Manager

To utilize the functionalities provided by Sterling Web to the maximum extent, you must set the certain business rules appropriately in the Channel Applications Manager before you start using the Sterling Web application. For more information about setting rules specific to Sterling Web, refer to the *Sterling Selling and Fulfillment Suite: Applications Configuration Guide*.

Determining a Storefront’s Appearance

A storefront is an organization that has the role of an Enterprise and a Seller, and has a theme associated with it. Because a storefront is an organization with a Web presence, it is important that the look-and-feel of a storefront is maintained.

Solution

Sterling Web enables you to control a storefront's look-and-feel by using cascading style sheets (CSS). The CSS files are grouped together to form a theme. A default theme is provided in the out-of-the-box implementation of Sterling Web. Each storefront can be configured to have a specific theme as its default theme. A storefront’s theme is defined in the Applications Manager.For every request routed to Sterling Web, the getOrganizationHierarchy API is called to verify whether the value of the sfId request parameter passed in the URL, or the StorefrontID parameter stored in the cookie, pertains to a valid storefront.

End-User Impact

None.

Implementation

Create a storefront by setting it up as an organization that has the role of an Enterprise and a Seller. For more information about creating an organization, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

Understanding the Home Page

Home page is one of the landing pages that is displayed when a user logs in to the Sterling Web™ application.

Solution

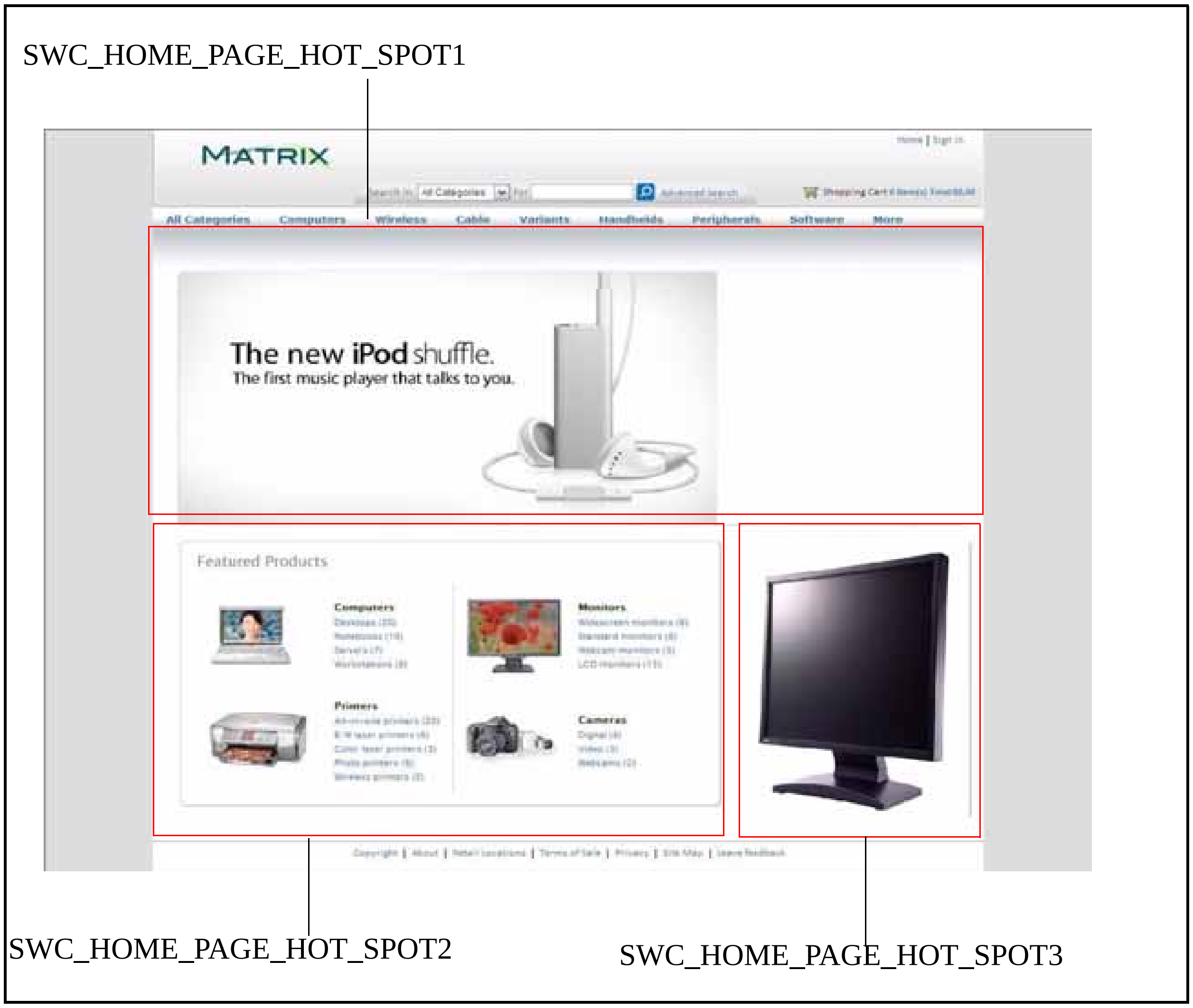
Sterling Web provides two different views of the Home page based on whether a user is a guest user or a buyer user who has logged in after providing the login credentials. The searchCatalogIndex API is called to retrieve the categories and items to be displayed on the Home page.

The Home page comprises the following areas displayed to the users based on their role:

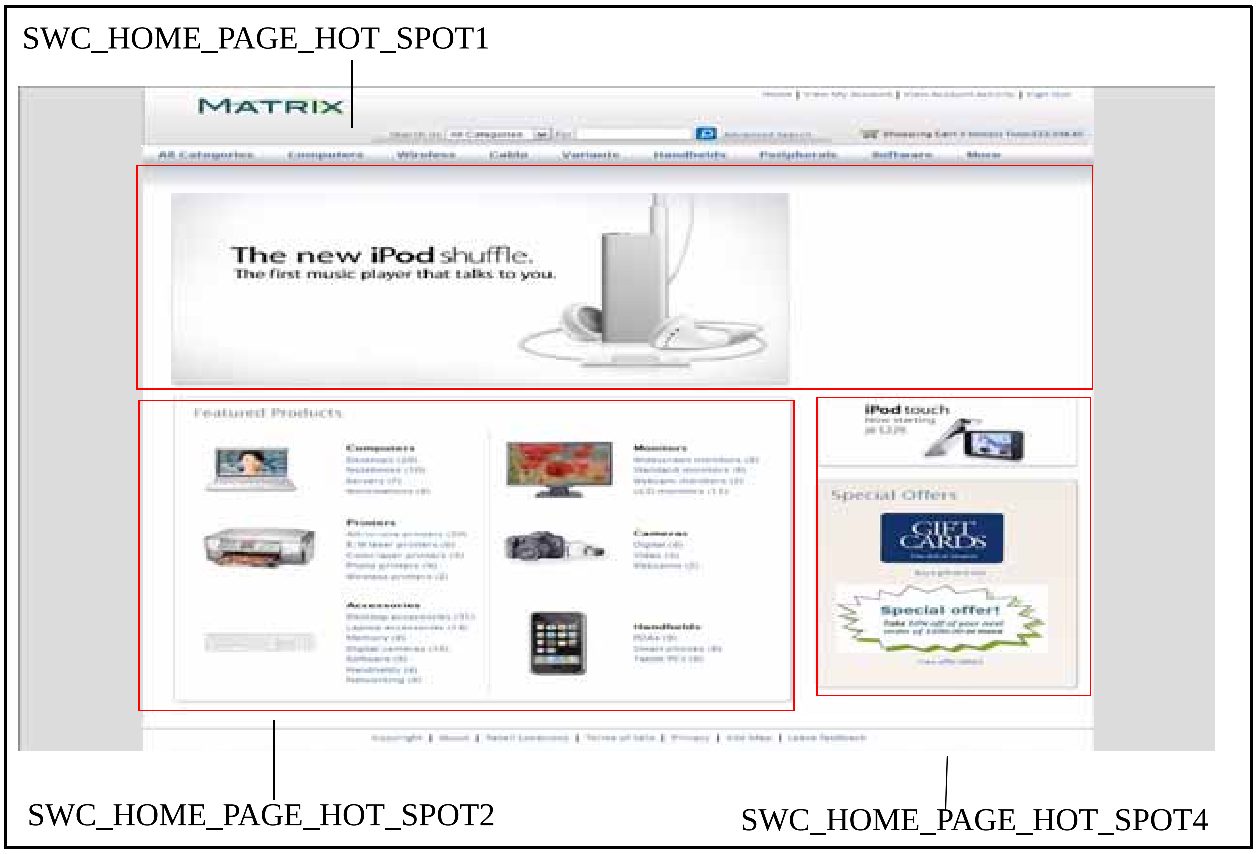
- Catalog Search: This enables all the users to search for products in the catalog.
- Catalog Navigation Bar: This enables all the users to view the product categories the user is entitled to. Users can use the catalog navigation bar to browse the categories.
- Mini Cart: This provides a quick view of the contents of a user’s default cart.
- Sign In hyperlink: This enables a user to log in to the Sterling Web application as a Buyer user.

Apart from these areas, the Home page comprises multiple hot spot regions. You can use the hot spot regions to display catalog advertisements, promotions, or special offers by defining assets and associating them with the hot spot regions. For more information about defining assets, refer to the *Business Center Item Administration Guide*.

A sample Home page accessed by a guest user, comprising the hot spot regions and the asset types associated with these regions, is shown in the following figure:



A sample Home page displayed to a buyer user after login, indicating the hot spot regions and the asset types associated with these regions, is shown in the following figure:



By default, the following asset types are defined for the Home page for a buyer user:

- SWC_HOME_PAGE_HOT_SPOT1
- SWC_HOME_PAGE_HOT_SPOT2
- SWC_HOME_PAGE_HOT_SPOT4

By default, the following asset types are defined for the Home page for a guest user:

- SWC_HOME_PAGE_HOT_SPOT1
- SWC_HOME_PAGE_HOT_SPOT2
- SWC_HOME_PAGE_HOT_SPOT3

In the preceding figures, the SWC_HOME_PAGE_HOT_SPOT1 asset type is used to display a catalog advertisement, SWC_HOME_PAGE_HOT_SPOT2 asset type to display the featured categories, SWC_HOME_PAGE_HOT_SPOT3 asset type to display the special offers for a guest user, SWC_HOME_PAGE_HOT_SPOT4 asset type to display the special offers for a buyer user.

End-User Impact

This section explains the end-user impact of this functionality:

- For a user, either the Home Page or the Account Activity page is displayed as the landing page based on the user preference set in the User Detail page.
- The categories displayed in the catalog navigation bar are based on the entitlements that are either assigned or dynamically applicable to the user who is accessing the application. For example, if the guest user is not entitled to access category XYZ, this category will not be displayed in the catalog navigation bar when a guest user accesses the application.
- The actions that a user can perform from the Home page are based on the permissions granted to the user's user group. A guest user can view the Sign In page. Additionally, a guest user can perform the following tasks:
  - Browse product catalog.
  - Search for products in catalog.
  - View mini cart.

  In addition to the tasks listed previously, a buyer user can, after logging in to the application, perform the following tasks:
  - Navigate to the Account Activity page.
  - Navigate to the User Detail page.
  - Sign out of the Sterling Web application.

Implementation

This section explains the configurations for this functionality:

- You must set the preferences for the post-login landing page as the Home page in the User Detail page to enable a user to view the Home page as the landing page.
- You must define customer entitlements to either grant or restrict guest users' or buyer users' access to the categories in a product catalog. For more information about how to create a customer entitlement rule, refer to the *Business Center Item Administration Guide*.
- You must define the appropriate entitlement strategies for the user at the organization level so that the customer entitlements can be applied. For more information about defining customer entitlements, refer to the *Sterling Distributed Order Management: Configuration Guide.*
- You must define the assets and asset types to be used to display catalog advertisements, promotions, or special offers in the hot spot regions. For more information about defining assets, refer to the *Business Center Item Administration Guide*.

  While defining assets and asset types for the Home page, you must take into consideration the following limitations:
  - The assets should be added at the catalog level.
  - The assets should be valid HTML files. HTML is the only file format that is supported for the assets.

- The content location of the asset should be a valid URL. The URL can be an absolute HTTP URL if the location is external to the deployment such as `http://<host>:<port>/...`, a relative URL if the location is internal, or a valid file URL such as `file://<host>/<path>`.

  Notes:

  Using the backslash (\) file separator to access files stored locally is dependent on the file system and may not work all the time.

  If the assets are stored in locations external to the deployment, it may be necessary to set up the proxy appropriately for the Web container to get by the firewall and other security measures.

Reference Implementation

None.

Authenticating Requests in Sterling Web

Every incoming request to the Sterling Web™ application must be authenticated for security reasons. Also every user accessing the Sterling Web application must be authenticated. Even after a user is authenticated, every request made by that user must be verified to ensure that the user is authorized to perform the requested task.

Sterling Web enables authentication of a user’s credentials every time a user accesses the application or tries to log in to the application. If the request does not require an explicit login, the application will automatically authenticate the request with an implicit guest user login. After a user is successfully authenticated, Sterling Web enables authorization of every HTTP request made by the user. Whenever the HTTP session expires, a new session is created and authenticated. However, if the request for a new session is generated from a logged-in user, the user is redirected to the Login page. The information in the URL identifies the theme of the storefront the user was browsing before the session expired. The authentication of the requests coming in to the Sterling Web application is performed using the WCAuthenticationProvider, which implements the ISCUIAuthenticaionProvider interface. For more information about authentication, refer to the *Selling and Fulfillment Foundation: Customizing the Web UI Framework Guide.*

For authentication and authorization of user, the user must set a password to log in to the Sterling Web application. If a user forgets the password, the user can reset it. For more information about enabling a user to reset the password, refer to the "Forgot Password" topic. A user must set the secret question and answer unique to the user’s profile. The password policy rules are configured in the Applications Manager. For more information about these rules, refer to the *Selling and Order Management Password Policy Management Guide*.

Authorizing Requests in Sterling Web

Authorization is performed to verify whether a user has permissions to perform tasks based on the permissions assigned to the user. Every JSP in the user interface is invoked through a struts action. Every action is associated with a resource ID which in turn, is associated with a specific group. This group has resource permissions assigned to it. When struts is invoked, authorization is performed to verify whether user has permission for the resource ID in the corresponding struts.

After a user is authenticated, every request from that session is authorized using a Struts interceptor called WCAuthorizationInterceptor. The WCAuthorizationInterceptor performs authorization verification for that session by using the ResourceID associated with the action definition specified in the struts.xml file. If the authorization is successful, the Struts action class is invoked, and the request is processed accordingly. If the authorization fails, and the user is a guest user, the user is redirected to the Login page. If the user is logged in to the application, an error message is displayed indicating that the user does not have the required permission to perform the action. For example, when a user clicks Home to navigate to the Home page, the WCAuthorizationInterceptor verifies the mapping of the Resource ID to the action Home in the home-struts.xml file. Based on the value of the Resource ID passed as the parameter, the Home page is displayed.

New user groups are created and appropriate resource permissions are assigned to them through Selling and Fulfillment Foundation. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide.*

Viewing Account Activity

A buyer user may sometimes want to view the activities related to the buyer user’s organization. For example, a buyer user may want to view the list of orders created for the buyer user’s organization, and that are on hold pending the buyer user’s approval.

Solution

Sterling Web enables a buyer user to view the Account Activity page either by clicking the **View Account Activity** hyperlink in the application header bar or as the post login landing page, based on the user’s preferences. For more information about setting a user’s preferences, refer to the topic "Managing Buyer User Details".

In the Account Activity page, a buyer user can view the account activity pertaining to the buyer user’s organization. The user can view the following information:

- Account Information: This includes the address information of either the buyer user or the buyer user’s organization, and the name of the buyer user’s organization. The user can also view the hyperlinks that enable the user to manage the details of the users of the buyer organization, or the details of the buyer organization, based on the permissions assigned to the user.

  The following logic is used to display the address information:

  a. The default Sold To address of either the user or the buyer organization is displayed if that exists.

  b. If the default Sold To address does not exist, the first Sold To address of the user is displayed.

  c. If the Sold To address of the user does not exist, the first Sold To address of the buyer organization is displayed. This address may be either the first Sold To address defined for the buyer organization or the first Sold To address inherited by the buyer organization from its parent organization. For example, let us consider two organizations, A and B, with A being a child organization of B. If A does not have a Sold To address defined for itself and inherits all the Sold To addresses from B, the address that is displayed is the first Sold To address that A has inherited from B.

  d. If the Sold To address of the buyer organization does not exist, the default Bill To address of either the user or the buyer organization is displayed.

  e. If the default Bill To address of either the user or the buyer organization does not exist, the first Bill To address of the user is displayed.

  f. If the first Bill To address of the user does not exist, the first Bill To address of the buyer organization is displayed. This address may be either the first Bill To address defined for the buyer organization or the first Bill To address inherited by the buyer organization from another organization.

  The getCustomerDetails API is called to retrieve the details displayed in the Account Information area.

- Orders: A user can view one or more of the following based on the data access policies configured for the user:
  - The list of orders owned by the user.
  - The list of orders owned by all the users of the buyer organization.

  - The list of orders owned by all the users of the buyer organization, and the users of the child organizations of this buyer organization.

  For more information about defining data access policies for buyer users, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

  The getOrderList API is called to retrieve the list of orders for the user.

- Orders Pending Approval: A user can view the list of orders that are kept on hold, pending approval from the user if the user has the necessary resource permissions. By default, the resource permissions are given to the BUYER APPROVER group. The getOrderList API is called to retrieve the orders pending approval from the user.
- Cart Management: Based on the permissions assigned to a user, the user can view the **Cart Management** hyperlink that enables the user to manage carts.

End-User Impact

This section explains the end-user impact of this functionality:

- The Account Activity page is displayed as a post login landing page based on the preferences that have been set for the corresponding user.
- A user can view the Account Activity page only if the user has the necessary permissions.
- A user can view the following hyperlinks in the Account Activity page only if the user has the necessary permissions:
  - View/Update your user profile
  - View/Update your organization profile
  - View/Update your user list
  - Cart Management
- In the **Orders** panel, a user will be able to view only those orders that the user has access to based on the data access policy configurations.
- In the **Orders for Approval** panel, a user will be able to view only those orders that have been kept on hold pending the user’s approval. This is based on the resource permissions assigned to the user. By default, the resource permissions are given to the BUYER APPROVER group. Additionally, the user will be able to view only those orders that the user has access to based on the data access policy configurations.

Implementation

This section explains the configurations for this functionality:

- You must set the user preferences to enable a user to view the Account Activity page as the post login landing page. For more information about setting a user’s preferences, refer to the topic "Managing Buyer User Details".
- You must assign permissions to the corresponding user groups to enable them to view the Account Activity page. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

- You must assign permissions to the corresponding user groups to enable them to view the hyperlinks that enable them to manage the details of users, the details of the corresponding buyer organization, and carts. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must assign a user to the BUYER USER group to enable the user to view the list of orders that are kept on hold and are pending approval from the user. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must configure the data access policies to enable a buyer user to view the orders owned by the user, all the users belonging to the same buyer organization, or all the users of the buyer organization as well as the users of the child organizations. For more information about defining data access policies, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

Remembering a User ID

Sometimes users may want Sterling Web™ to remember their user ID so that they do not have to provide that information every time they log in to the application. A user accessing Sterling Web can either be a guest user or a buyer user.

Solution

A user, whose information the application stores, can view the catalog and prices of products, add or delete products from the user's cart, and view the **View Account Activity** and **View my Account** hyperlinks. The preferences chosen by a user when the user last logged in to the application are used when the user logs in subsequently. To modify these preferences, the user must log in to the Sterling Web application.

Sterling Web provides a user interface that allows a user to choose whether the Sterling Web application should remember the user's login information the next time the user logs in to the application. Sterling Web stores the information pertaining to a user in the form of a cookie. If a user selects the **Remember Me** check box when the user logs in, a cookie will be stored on the client machine. The name of the cookie is stored in the following format:

UserID_<Organization Code of the storefront>

For example, if the user's user name is ABC and the Organization Code of the storefront is 100, the name of the cookie will be UserID_100 and value of the cookie will be ABC.

When a user identified as a guest user browses Sterling Web, the Sterling Web application verifies if a cookie exists for the guest user's cart. The name of the cookie is stored in the following format:

CartID_<Organization Code of the storefront>

The getCompleteOrderDetails API is called to get the details for the order header key stored in the cookie for the guest user's cart in order to authenticate whether the cart is a valid guest user's cart.

When the user logs in to Sterling Web the next time, the cookie from the browser is read and validated against the user in the database. The getUserHierarchy API is called to validate if the user name stored in the cookie is a valid user name and belongs to the current storefront. If the validation is successful, the user's user name is displayed in the Login page. If the validation is not successful, the cookie stored on the client machine is deleted.

When the remembered user accesses the Login page, a welcome message is displayed along with a **Click here if it isn't you** hyperlink. The getCustomerContactList API is called to retrieve the first name and last name of the user and display it in the welcome message. If the user clicks the **Click here if it isn't you** hyperlink, the user switches from being a remembered user to a guest user and the previously remembered user's cart gets copied to a new cart pertaining to the guest user. The price of the products in the new cart pertaining to the guest user are based on the price list assigned to the guest user through Business Center. For more information about price lists, refer to the *Business Center Pricing Administration Guide*.

The copyOrder API is called to copy the remembered user's cart. The changeOrder API is called to remove any user related information from the order header of the new cart so the address or payment details are not copied to the new cart. The payment and address information in the order header level detail and the order lines is not copied to the new cart for the guest user.

End-User Impact

This section explains the end-user impact of this functionality:

- Based on the configuration of the **Allows Users To Select The “Remember Me” Option During Login** rule, a user will be able to see or not see the **Remember Me** check box in the Login page. By default the **Remember Me** check box is selected. For more information about configuring this rule, refer to the *Channel Applications Manager: Configuration Guide*.
- If a user chooses to remember the user’s login information and leaves the system, a different user can access Sterling Web on the same system and view the data of the remembered user. In such a scenario, the user must not select the **Remember Me** check box while logging in. A user must clear the cookie cache before leaving the system.

Implementation

This section explains the configurations for this functionality:

- You must configure the **Allows Users To Select The “Remember Me” Option During Login** rule in the Channel Applications Manager to allow users to choose whether they want to be remembered by the application. For more information about configuring this rule, refer to the *Channel Applications Manager: Configuration Guide*.
- This rule is storefront-specific. For example, if the **Allows Users To Select The “Remember Me” Option During Login** rule is enabled for a storefront ABC, all the users of the ABC storefront can choose whether they want to be remembered by the application. By default, this rule is enabled. For more information about configuring this rule, refer to the *Channel Applications Manager: Configuration Guide*.

Reference Implementation

None.

Forgot Password

Sometimes users accessing the Sterling Web™ application may forget the password they use to log in to the application. In such a scenario, users may want to reset their password.

Solution

Sterling Web enables users to reset their password by clicking the **Forgot Password** hyperlink in the Login page.

When a user clicks the **Forgot Password** hyperlink, the system launches the following procedure:

1. The user is asked to enter the User ID.
2. After the user provides the User ID, the getUserHierarchy API is called to validate whether the user belongs to the corresponding storefront or not.
3. If the system confirms that the user belongs to the storefront, the user is asked to answer the secret question the user provided at the time the corresponding user profile was created. (The system calls the getAuthQuestionList API to retrieve the secret question).
4. After the user enters the answer and clicks Submit, the validateAuthAnswers API is called to validate the answer.
5. If the answer is confirmed as being correct, the **Confirmation Is Required On Password Reset** rule is verified whether it is enabled or not in the Applications Manager.
6. If the **Confirmation Is Required On Password Reset** rule is enabled, the requestPasswordReset API is used to generate a random string (password request ID).
7. The requestPasswordReset API raises the RESET_PASSWORD.ON_REQUEST event. The random string generated by the requestPasswordReset API is passed to the RESET_PASSWORD.ON_REQUEST event, which in turn invokes the YCD_Send_Reset_Password_8.5 service to generate an e-mail containing a generated URL. This URL is sent to the user's e-mail address.
8. When the user clicks the URL provided in the e-mail, the user is directed to Sterling Web, where the user is asked to provide the User ID again.
9. This User ID is verified along with the password request ID in the URL.
10. The user is redirected to a new page to provide a new password and click the Submit button.
11. On clicking Submit, the changePassword API is called to verify whether the new password meets the password policy criteria, and save the new password. The validatePasswordResetRequest API is called to validate the request ID embedded in the URL.

If the **Confirmation Is Required On Password Reset** rule is not enabled in the Applications Manager, the system redirects the user to provide a new password. On clicking the Submit button, the changePassword API is then called to verify whether the new password meets the password policy criteria, and save the new password.

Either a buyer administrator or a customer service representatives can create a new user using Sterling Web. However, neither a buyer administrator nor the customer service representative is allowed to set the secret question and answer for the new user. The new buyer user must set the secret question and answer unique to that profile.

The password policy rules are configured in the Applications Manager. For more information about these rules, refer to the *Selling and Order Management Password Policy Management Guide*.

Notes:

- The new buyer user must set the secret question and answer unique to the buyer user's profile. This task is a prerequisite to initiating a forgot password request at a future date. Whenever a user provides the secret answer, the comparison between the saved secret answer and the answer provided by user is not case sensitive.
- If a user forgets the secret question and the corresponding answer, the administrator can request password-reset for that user by clicking the **Reset Password** button in the user's profile.

Sterling Web enables you to allow users to skip the e-mail confirmation step and directly reset their password in the user interface by configuring the **Confirmation Is Required On Password Reset** rule in the Applications Manager. For more information about configuring a rule to send confirmation to users after password reset, refer to the *Selling and Order Management Password Policy Management Guide*.

End-User Impact

This section explains the end-user impact of this functionality:

Users can reset their password only after a confirmation e-mail is sent to them if the **Confirmation is Required On Password Reset** rule is enabled in the Applications Manager, and the `ResetType` attribute is set to Email in the for the active Password Policy in the Applications Manager. For more information about resetting passwords through e-mail, refer to the *Selling and Order Management Password Policy Management Guide*.

Implementation

This section explains the configurations for this functionality:

- You must configure the **Confirmation is Required On Password Reset** rule appropriately to enable users to reset their password either directly in the user interface or after a confirmation e-mail is sent to them. By default, this rule is not enabled. For more information about configuring a rule to send confirmation to users after the password is reset, refer to the *Selling and Order Management Password Policy Management Guide*.
- You must configure the list of secret questions for the organization. For more information about configuring secret questions, refer to the *Selling and Order Management Password Policy Management Guide*.
- Ensure that the YCD_Send_Reset_Password_8.5 service is configured for the RESET_PASSWORD.ON_REQUEST event to send the e-mail to the user with a generated URL, which in turn is sent to the user's e-mail address. For more information about configuring this service, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

Managing Buyer Organization Details

A buyer organization is an organization that purchases products from a storefront. After a buyer organization is created, buyer administrators belonging to that organization may want to view the details of that buyer organization, update the details of that organization, create child organizations, or update the details of the child organizations.

Solution

This section explains the solution provided by Sterling Web™ to enable buyer users to manage the details of a buyer organization. This includes viewing and modifying buyer organization details, creating child organizations, and viewing and updating the details pertaining to the child organizations. For example, if the address of a child organization has been modified, a buyer administrator, who is a buyer user with administrator privileges, can update the details of the child organization accordingly.

Note: Only a buyer administrator can modify the details of the buyer organization.

View Buyer Organization Details

Sterling Web enables buyer users to view information pertaining to their organization based on the permissions assigned to them. The getCustomerDetails API is called to retrieve the buyer organization details.

A buyer administrator can view the following information:

- **General Information**: This includes information pertaining to a buyer organization, such as the name of the organization, the organization's Web site address, default currency used by the buyer organization, the buyer organization’s status, the buyer organization’s effective status, and so on. A buyer administrator can also view the address details of the buyer organization.

  The effective status of a buyer organization is the aggregate status of the buyer organization calculated after taking into consideration the status of the organizations in the hierarchy. However, the status of the child organizations of the buyer organization will not be taken into consideration for calculating the effective status.

  The status of a buyer organization can take one of the following values:

  - 10 (Active)
  - 20 (On Hold)
  - 30 (Inactive)

  To calculate the effective status, the highest status value takes priority. For example, if a buyer organization’s status is Active (10) and the status of the buyer organization’s parent organization is Inactive (30), the buyer organization’s effective status will be displayed as Inactive. If the buyer organization’s status is On Hold (20) and the status of the parent organization is Active (10), the buyer organization’s effective status will be displayed as On Hold.

- **Corporate Information**: This includes information pertaining to shipping preferences. A buyer administrator can view the list of child organizations under their organization’s hierarchy. The getCustomerList API is called to display the list of child organizations.

- **Payment Information**: This includes payment-related information such as payment methods, whether orders should be exempt from tax, and tax exemption certificate number. Sterling Web supports the Credit Card and Account payment methods for a buyer organization.
- **Notes**: This includes the notes entered about a buyer organization by buyer administrators, enterprise users, customer service representatives, or store representatives.

Modify Buyer Organization Details

Sterling Web enables buyer administrators to modify information pertaining to their organization and its child organizations based on the permissions assigned to them.

A buyer administrator can modify the following information:

- **General Information**: This includes modifying the name of the buyer organization, the organization's Web site address, D&B (Dun&Bradstreet) ID, and default currency. A buyer administrator can also edit, delete or duplicate an existing address, or add a new address for either their organization or its child organizations, as also the Bill To, Ship to, and Sold to addresses of their organization. A buyer administrator can modify the customer status of its child organization. If a default address has not been specified for a child organization, the default address defined for the corresponding parent organization is inherited and displayed for that child organization. If the corresponding parent organization too does not have a default address, the default address of the parent organization’s parent organization is displayed as the child organization’s default address. This hierarchical inheritance of default address continues till the last required level. The getCurrencyList API is used to display the corresponding currency in the **Currency** drop-down list.The getRuleDetails API is called to retrieve the value of the **Region Schema to Use for State Selection** rule configured in the Channel Applications Manager. For more information about configuring this rule, refer to the *Channel Applications Manager: Configuration Guide*. Based on the value of this rule, the getRegionList API is called to display the list of states in the **State** drop-down list. The getCommonCodeList API is called to display the corresponding value in the **Customer Status drop-down** list. The value displayed in the **Relationship Type**, **Membership Level**, **Effective Status**, and **Vertical** fields is retrieved from the output of the getCustomerDetails API**.** The getCommonCodeList API is called to populate the **Country** and **Title** drop-down lists with the applicable values.

  When a new address is added, or an existing address is modified, the address is validated by calling the verifyAddress API, which in turn calls the YCDVerifyAddressWithAVSUE user exit. For more information about the YCDVerifyAddressWithAVSUE user exit, refer to the *Selling and Fulfillment Foundation: Javadocs.*
- **Corporate Information**: This includes modification of shipping preferences. A buyer administrator can also add new child organizations to their organization and modify information pertaining to these child organizations.
- **Payment Information**: This includes modification of the payment methods of their organization and its child organizations. Additionally, a buyer administrator can add a new payment method, delete an existing payment method, and set a payment method as the default payment method of their organization or its child organizations. When a payment method with payment type as Credit Card is added, a billing address can be associated with it. A buyer administrator can exempt the child organizations from tax. The getRuleDetails API called to retrieve the value of ENCRYPT_ADDNL_ATTRIBUTES_CREDIT_CARD_PAYMENT_TYPE_GROUP attribute. This attribute enables encrypting and masking of the credit card holder’s name and the expiry date of the credit card.
- When a new payment method is added, the getPaymentTypeList API is called to display the corresponding payment types in the **Payment Type** drop-down list. The getCommonCodeList API is

called to retrieve the value for YCD_CREDIT_CARD_TYPE to list the credit card types. The payment options specified in this rule are displayed in the user interface. For more information about setting credit card options, refer to the *Channel Applications Manager: Configuration Guide*. The getCurrencyList API is used to display the corresponding currency in the **Currency** drop-down list. The getRuleDetails API is called to retrieve the value of the **Region Schema to Use for State Selection** rule configured in the Channel Applications Manager. For more information about configuring this rule, refer to the *Channel Applications Manager: Configuration Guide*. Based on the value of this rule, the getRegionList API is called to display the bill to address on the credit cart used for making payments.

- **Notes**: A buyer administrator can add new notes about their organization and its child organizations. The manageCustomer API is called to save these notes.

Create a Child Organization

Sterling Web enables buyer administrators to create child organizations for their organization based on the permissions assigned to them. Buyer administrators can create a child organization for their organization and the organizations that fall under its hierarchy.

After the buyer administrator provides the mandatory information required to create a child organization, such as the name of the organization and the customer ID, the manageCustomer API is called to create the child organization with the information provided.

The following attributes are inherited by the newly created child organization from the parent organization:

- Relationship value
- Membership level
- Vertical

End-User Impact

This section describes the end-user impact of this functionality:

- Buyer users can manage the details of their organization and its child organizations only if they have the necessary permissions.
- The **State** drop-down list is displayed only if you configure the **Region Schema to Use for State Selection** rule in the Channel Applications Manager. The **State** text box is displayed if the **Region Schema to Use for State Selection** rule is not configured in the Channel Applications Manager. For more information about configuring the **State** drop-down menu, refer to the *Channel Applications Manager: Configuration Guide*.
- A buyer user cannot modify a child organization's relationship level, membership level, and vertical because these are inherited from the parent organization.
- If a buyer administrator changes the customer status of the buyer organization, the status of all the child organizations in the hierarchy will also change accordingly.
- If the customer status of a buyer organization is On Hold, the customer status of all the corresponding child organizations in the hierarchy will also be On Hold. When the customer status of an organization is On Hold, orders created for that organization or any of the child organizations in the hierarchy will not be processed based on the configuration of the **Hold Type to be applied to an Order when Customer Contact is on Hold** rule. To modify this attribute, configure the **Hold Type to be applied to an Order when Customer Contact is on Hold** rule in the Applications Manager. For more

information about configuring the **Hold Type to be applied to an Order when Customer Contact is on Hold** rule, refer to the *Sterling Distributed Order Management: Configuration Guide*.

Implementation

This section explains the configurations for this functionality:

- You must configure a region schema to be used for displaying the list of states in the **State** drop-down list in the Channel Applications Manager. For more information about configuring a region schema, refer to the *Channel Applications Manager: Configuration Guide*.
- Based on the values specified in YCD_CREDIT_CARD_TYPE, payment options will be displayed in the user interface. for more information about setting credit card options, refer to the *Channel Applications Manager: Configuration Guide*.
- You must assign the necessary permissions to the corresponding user groups to enable them to manage a buyer organization’s details, create child organizations, and manage the details of child organizations. By default, permissions for managing the details of a buyer organization and its child organizations are assigned to the BUYER-ADMIN user group. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must implement the YCDVerifyAddressWithAVSUE user exit to verify the address with the Address Verification System (AVS) whenever a new address is added for a buyer organization or its child organizations.
- You must configure the **Hold Type to be applied to an Order when Customer Contact is on Hold** rule to enable orders to be put on hold when the customer status of an organization is On Hold. For more information about configuring the **Hold Type to be applied to an Order when Customer Contact is on Hold** rule, refer to the *Sterling Distributed Order Management: Configuration Guide*.

Reference Implementation

None.

Managing Buyer User Details

A buyer user is a user who can purchase products from a storefront on behalf of the buyer organization to which the user belongs. Buyer users must be able to view or update their own user details and view or update the details of other buyer users who belong to the same buyer organization.

Solution

This section explains the solution provided by Sterling Web™ to enable buyer users to manage their own user details, and buyer administrators to manage the details of other buyer users belonging to their organization. Based on the permissions assigned to them, buyer users can view and modify their own user details. Additionally, buyer administrators can search for users belonging to their organization, and view and modify the details of these users.

Search for a Buyer User

Sterling Web enables buyer administrators to search for buyer users belonging to their organization based on the permissions assigned to them. A buyer administrator can search for other buyer users by providing the relevant search criteria in the User List page and clicking **Search**. The search is case-sensitive. A buyer administrator can perform a search by using First Name, Last Name, E-mail ID, or Business Phone as the search criteria.

A buyer administrator can also use the wild card characters * to search for buyer users. The wild card character can be inserted either at the beginning for a search term as a prefix or at the end of a search term as a suffix. For example, if the buyer administrator is searching for a buyer user with XYZ as the First Name, the buyer administrator can perform a quick search by providing XY* as a search term. Based on how the search term is entered, the query type for the search is passed as shown in the following table:

Search Term	Query Type
XYZ	LIKE
XYZ*	FLIKE
XYZ	EQ

The buyer administrator can also click the **Show All** hyperlink to view all the buyer users belonging to their organization. If any existing search settings are available, those settings will not be considered when the results are displayed. The getCustomerContactList API is called to retrieve the list of buyer users matching the search criteria entered.

Pagination

If the number of buyer users matching the search criteria is more than ten, the User List page displays the results across multiple pages. A buyer administrator can navigate to any of the pages by clicking the corresponding page number hyperlink at the bottom of the page. The getPage API is called to retrieve the pages to be displayed. For more information about the getPage API, refer to the *Selling and Fulfillment Foundation: Javadocs*.

Create Buyer User

Sterling Web enables buyer administrators to create new users for a buyer organization and its child organizations based on the permissions assigned to them. After the mandatory information, such as the user name, preferred locale, first name, last name, and e-mail ID, is provided, the manageCustomer API is called to create the new buyer user. The e-mail ID that is provided for the buyer user is used to receive the system-generated password. Using this password, the new buyer user can log in to Sterling Web and reset the password or set a secret question and answer to be used when the user forgets the password.

View Buyer User Details

Sterling Web enables buyer users to view their own user details based on the permissions assigned to them. Additionally, buyer administrators can view the details of other buyer users belonging to their organization and its child organizations. The getCustomerContactList API is called to display the details of a buyer user.

Note: Buyer administrators cannot view other buyer users' password and the secret question set for other buyer users.

A buyer user can view the following information:

- **General Information**: This includes information such as the user name, preferred locale, job title, address information, phone numbers, secret question, addresses specified for the user's organization, status, effective status, and so on. Buyer users can also view the addresses specified for their organization.

  The effective status of a buyer user is the aggregate status of the buyer user calculated after taking into consideration the status of the organizations in the hierarchy. However, the status of the child organizations of the buyer user's organization will not be taken into consideration for calculating the effective status.

  The status of a buyer user or an organization can take one of the following values:

  - 10 (Active)
  - 20 (On Hold)
  - 30 (Inactive)

  To calculate the effective status, the highest status value takes priority. For example, if a buyer user's status is Active (10) and the status of the buyer user's organization is Inactive (30), the buyer user's effective status will be displayed as Inactive. If the buyer user's status is On Hold (20) and the status of the buyer user's organization is Active (10), the buyer user's effective status will be displayed as On Hold.
- **Preferences**: This includes information such as the checkout type and the post-login landing page for a buyer user. If the user preferences have been specified for the buyer user, the getUserUiStateList API is called to retrieve the user's preferences. If the user preferences have not been specified, the getRuleDetails API is called to retrieve the default preferences.
- **Payment Information**: This includes payment-related information such as the payment methods for the buyer user the organization of the buyer user. A buyer administrator can also view the information about spending limit, approver, and proxy for the approver of the buyer user. The getCustomerContactList API is called to retrieve the payment-related information of the buyer user.

An approver and proxy approver are buyer users who have been assigned permissions to approve an order that is on hold. An order is put on hold if the order total has exceeded the buyer user's spending limit.

Note: Sterling Web does not allow users to view sensitive payment information, such as credit card number and the customer account numbers. Therefore, based on the implementation, these attributes are encrypted and hidden. For example, for the Credit Card Number, a buyer user can view the `DisplayCreditCardNo` attribute value generated by the manageCustomer API.

- **Notes**: This includes viewing notes, if any, entered for a buyer user. Additionally, a buyer administrator can view the notes entered for other buyer users belonging to their organization. The getCustomerContactList API is called to retrieve the notes.

Modify Buyer User Details

Sterling Web enables buyer users to modify their own user details based on the permissions assigned to them. Additionally, buyer administrators can modify the details of other buyer users belonging to their organization and its child organizations.

A buyer user can modify the following information:

- **General Information**: This includes general information such as the password, preferred locale, phone numbers, address information, secret question, and so on. The getLocaleList API is called to display the list of locales in the **Preferred Locale** drop-down list. The getCommonCodeList API is called to retrieve the applicable values for the **Status** drop-down list and the **Title** drop-down list. The getAuthQuestionList API is called to retrieve the questions for the **Secret Question** drop-down list. A buyer administrator can also modify the details of other buyer users such as the preferred locale, phone numbers, address information, and so on. However, a buyer administrator cannot modify the password and secret question of other buyer users. Instead, a buyer administrator can place a request for resetting a buyer user's the password. When the buyer administrator clicks the **Reset Password** button, an e-mail containing the URL of the page in which the password can be reset is sent to the corresponding buyer user's e-mail address. When a new address is added or an existing address is modified, the address is validated by calling the verifyAddress API, which in turn calls the YCDVerifyAddressWithAVSUE user exit. For more information about the YCDVerifyAddressWithAVSUE user exit, refer to the *Selling and Fulfillment Foundation: Javadocs*.

  A buyer administrator can deactivate a buyer user by selecting the **Inactive** option from the **Status** drop-down list. A user can be in the **Active**, **On hold**, or **Inactive** status. If the user is in the **Active** status, the user can perform all the functions, including adding products to carts or placing an order. A user who is in the **On Hold** status can perform all the functions, but all the orders placed will be kept on hold. A user who is in the **Inactive** status cannot log in to the Sterling Web application.
- **Address Information**: Buyer users can modify the addresses set for them, add a new address, delete or duplicate an address, and classify an address as their default Bill To, Ship To, or Sold To address. However, they cannot modify any of the addresses that have been inherited from the buyer organization but can only set them as their default address.
- **Preferences**: This includes the user's preferences such as the checkout type and the post-login landing page for the user. Buyer users can modify their own user preferences. Additionally, buyer administrators can modify the user preferences of other buyer users belonging to their organization.
- **Payment Information**: This includes payment-related information such as the payment method, spending limit, approver, proxy for approver, and so on. The getUserList API is called to retrieve the values for the **Approver name** and **Proxy for approver** drop-down lists. The getCurrencyList API is called to retrieve the values for the **Currency type** drop-down list. Buyer administrators can also

modify the spending limit, approver, and the proxy for the approver for other buyer users belonging to their organization. A buyer user can also add a new payment method of the CREDIT_CARD payment type group or delete an existing payment method. However, a buyer user cannot delete a payment method that has been inherited from their organization. Spending limit is the maximum amount a buyer user is allowed to spend.

- **Notes**: Buyer users can add notes to their own profile. Additionally, a buyer administrator can add notes for other buyer users belonging to their organization.

After the modifications have been made, the manageCustomer API is called to save the updated details of the buyer user.

End-User Impact

This section explains the end-user impact of this functionality:

- Buyer users can manage their own user details and the details of other users, but only if they have the necessary permissions.
- If a buyer user does not have any addresses set, the **Delete** and **Duplicate** buttons in the **My address book** area under the **General Information** tab are disabled.
- If a buyer user without the login credentials is created using the Sterling Call Center™ application, a buyer administrator can view and modify the details of the buyer user in the Sterling Web application. Additionally, a buyer administrator can create the login credentials for the buyer user by providing a user name.
- A user can view or modify the user preferences only if the **Allow Users To Change Their Preferences** rule is enabled in the Channel Applications Manager.

Implementation

This section explains the configurations for this functionality:

- You must set the list of secret questions for an organization in the Applications Manager. For more information about creating a question, refer to the *Selling and Order Management Password Policy Management*.
- You must configure **Allow Users To Change Their Preferences** rule to enable users to view or modify the user preference. For more information about configuring this rule, refer to the *Channel Applications Manager: Configuration Guide*.
- You must assign the necessary permissions to the corresponding user groups to enable them to manage the details of a buyer user. By default, permissions are assigned to the BUYER-USER group to enable them to manage their own user details. Additionally, by default, permissions are assigned to the BUYER-ADMIN group to enable them to manage the user details of other buyer users belonging to their organization. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You can specify whether the storefront users accessing the Sterling Web application are globally unique or unique only within the corresponding storefront by configuring the **Allow the display User ID to be used across all Enterprises** rule appropriately in the Applications Manager. By default, this rule is enabled, and the users are configured as globally unique. For more information about configuring users, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

- You must implement the YCDVerifyAddressWithAVSUE user exit to verify the addresses with the Address Verification System (AVS).

Reference Implementation

None.

Browsing Catalog

Users may want to browse the catalog to view the list of products under a particular category or search the catalog to look for a particular product they are interested in purchasing. For example, a user who is browsing the catalog to purchase a laptop may not be interested in viewing the entire list of products in the catalog, and may want to view only the products available under the category Computers.

Solution

Sterling Web™ enables a user to browse the catalog from the Home page and search for products. The product categories are displayed in the catalog navigation bar based on the entitlements that are specified at the storefront level for the user. For more information about entitlements, refer to the *Business Center Item Administration Guide*.

In the catalog navigation bar, a user can view only those categories that have at least one product the user is entitled to within these categories or their subcategories. For example, if there is a category called Computers with Desktops and Notebooks as its subcategories, and a user is not entitled to any of the products in the Desktops and Notebooks subcategories, the user will not be able to view the Computers category in the catalog navigation bar.

The number of categories displayed in the catalog navigation bar is based on the value of the **SWC_NUM_OF_CATEGORIES_DISPLAY** rule, configured in the Channels Application Manager. The searchCatalogIndex API is called to retrieve the categories and the products to be displayed under a selected category in the Category Product List page.

The images and assets displayed in the Product List page and the product detail page respectively are associated with corresponding products in the Business Center application. You can associate images with the products displayed in the Product List page. You can also associate assets with the products displayed in the product detail page. For more information about defining a product's main image and adding an asset to an item, refer to the *Business Center Item Administration Guide*. The image of the product displayed in the Product List page is of 125 X 125 pixels. The Category Product List page also displays promotional offers. The assets that are used to display these promotional offers are associated with the CategoryTopSellers and CategoryNewArrivals asset types. For more information about how to add an asset to a category, refer to the *Business Center Item Administration Guide*.

The category IDs corresponding to the categories displayed in the catalog navigation bar of Sterling Web are created through the Business Center application. By default, Sterling Web uses Apache Lucene, a third-party software, for generating search indexes. This software does not support IDs containing spaces. Therefore, a user must not include spaces in the IDs if the user is using the Lucene parser. For example, if the user creates a category ID, Wireless Products, searching for products under the Wireless Products category and subcategories will display incorrect products. However, if the user creates the category ID WirelessProducts or Wireless_Products, the correct products will be displayed.

Sterling Web enables a user to filter the products they want to view based on the categories and subcategories. The catalog navigation bar displays the categories and subcategories available for viewing by a user. Users can further refine the list of products they want to view by performing a search for the products in the Browse Categories and Narrow By panel.

When user clicks on a particular category the corresponding sub category is listed in the Browse Categories panel. The number of products present in a subcategory are displayed adjacent to the sub category. A user can expand the sub category and view the products. In the Narrow By panel a user can view the number of products adjacent to the attribute values that are associated with the sub category. A user can navigate through attributes and the corresponding values. The user can click an attribute value and view the search results.When a user expands a particular category in the Narrow By panel, the products based on attribute and attribute values are displayed. The searchCatalogIndex API is called to retrieve the search results.

When a user browses a catalog, a bread crumb is generated at every step. This enables the user to return to any of the earlier steps. The user can also delete a bread crumb in order to delete the associated search condition from subsequent search queries. If a user navigates to a primary page, for example, Home > Variants, the user can click the **Home** hyperlink to directly navigate back to the catalog landing page. However, if a user navigates to a secondary page, for example, Home > Variants > MPLXMODEL7550, and clicks the **Back** hyperlink that is displayed as a bread crumb (Back > MPLXMODEL7550), in the product details page, the product list page is displayed instead of the catalog landing page.

The normal, condensed, and mini view options are available to the user in the product list and search result page.

Sterling Web enables a user to search for a product while browsing the catalog. A user can search for a product using the following options:

- Quick Search: A user can perform a quick search for a product by providing a search term as the search criteria. You can configure the search terms that can be used by a user to search for a product. The search terms are configured in the `CatalogSearchConfigProperties.xml` file that is located in the `resources.jar/template/resource` folder. For example, to enable a user to search for an item by the Item ID, you must set the **XMLName** attribute to ItemID, and **Searchable** attribute to Y. If a user navigates to a category having sub categories, the name of the category will be displayed in the **Search In** drop-down list. The search result comprises a restricted set of products based on the selected category.
- Advanced Search: A user can perform an advanced search for a product by providing more restrictive criteria.

Quick Search

The user can use wild card characters to search for a product. For example, if the user is looking for a product with XYZ123 as Product ID, the user can perform a quick search for the product by providing XY* as a search criteria. If the user provides XY "Z" as the search criteria, the search will be conducted for either of the terms, XY or Z. However, if the user provides "XY Z" as the search criteria, the search will be conducted for both the terms, XY and Z, in the same order. The searchCatalogIndex API is called to retrieve the list of products matching the search criteria.

Advanced Search

A user can perform an advanced search for a product by providing more restrictive criteria to down the search results. A user can navigate to the Advanced Search page by clicking the **Advanced Search** hyperlink in the application header bar or the **Narrow By** panel of the **Product List** page. The user can select the **Include superseded products** check box to include the superseded products while searching for a product. The user can also clear all the search terms by clicking the **Reset Page** hyperlink. A superseded product is a product that has been configured to replace another product whose effective end date has reached. For example, you can configure a product A to be superseded by another product B when the effective end date of A expires. For more information about associating a supersession product, refer to the *Business Center Item Administration Guide*.

A user can perform an advanced search for a product using the following criteria:

- **Narrow By**: A user can narrow down the search category to one category and within this category, perform an advanced search for a product. From the **Search this category** drop-down list, a user can select the category to be included in the search. The search criteria will be restricted to the selected category and its subcategories. The searchCatalogIndex API is called to display the values in the **Search this category** drop-down list. If a user navigates to the Advanced Search page from the **Narrow By** panel in the Product List page, the category path will be displayed and the **Search this category** drop-down list will display all the subcategories for that category. If there are no subcategories for the selected category, the **Search this category** drop-down list will not be displayed. The user can also enter the price range to be included in the search.
- **Match By**: A user can select key words and perform an advanced search for a product. In the **One or more of these keywords**, **All of these keywords**, and **None of these keywords** areas, a user can select the Product ID, Product Name, Product Description, and so on from the **Field** drop-down list. The getSearchableIndexFieldList API is called to display the values in the **Field** drop-down list. The user can also enter the key word to be included in the search.

A user can select the normal, condensed, or mini view of the search result by clicking the corresponding icon in the user interface. The getCompleteItemList API is called to display the details of a product in the Product Comparison page. A user can also select the products for comparison in the Product List page by dragging and dropping it in the Drag to Compare basket. This basket displays the number of products added for comparison. If the user has configured the alternative view for the Product List page, the user can compare the products by selecting them and clicking the **Compare** hyperlink. However, only the first four products will be compared. A user can select and remove the products in the Product Comparison page. If a user has previously included, say, four products for comparison, and adds four more products now, the products added previously will be automatically moved to a list above the Product Info panel as hyperlinks. This is because only four of the latest products can be compared. For more information about configuring the alternative page, refer to the *Sterling Web Customization Guide*.

Sterling Web enables a user to search for a product within the search results. This enables a user to further narrow down the search results. The searchCatalogIndex API is called to retrieve the product or the list of products matching the search criteria.

In the Category Product List page, a user can select the number of search results to be displayed on each page. A user can also sort the search results by the product's name or product's price in either ascending or descending order.

descending order.

Configuring an Alternative View of the Product List page

A user can configure an alternative view of the Product List page by performing appropriate modifications.

To configure the alternative view of the Product List page, perform the following steps:

1. Determine the name of the storefront for which you want to associate an alternative view.
2. Navigate to the `<INSTALL_DIR>/repository/eardata/swc/war/WEB-INF/classes/` folder and copy the `alternate_catalog_flow_struts.xml.sample.`

3. Navigate to the customization folder created by the end-user and paste the `alternate_catalog_flow_struts.xml.sample` file.
4. Rename the `alternate_catalog_flow_struts.xml.sample` to `alternate_catalog_flow_struts.xml`.

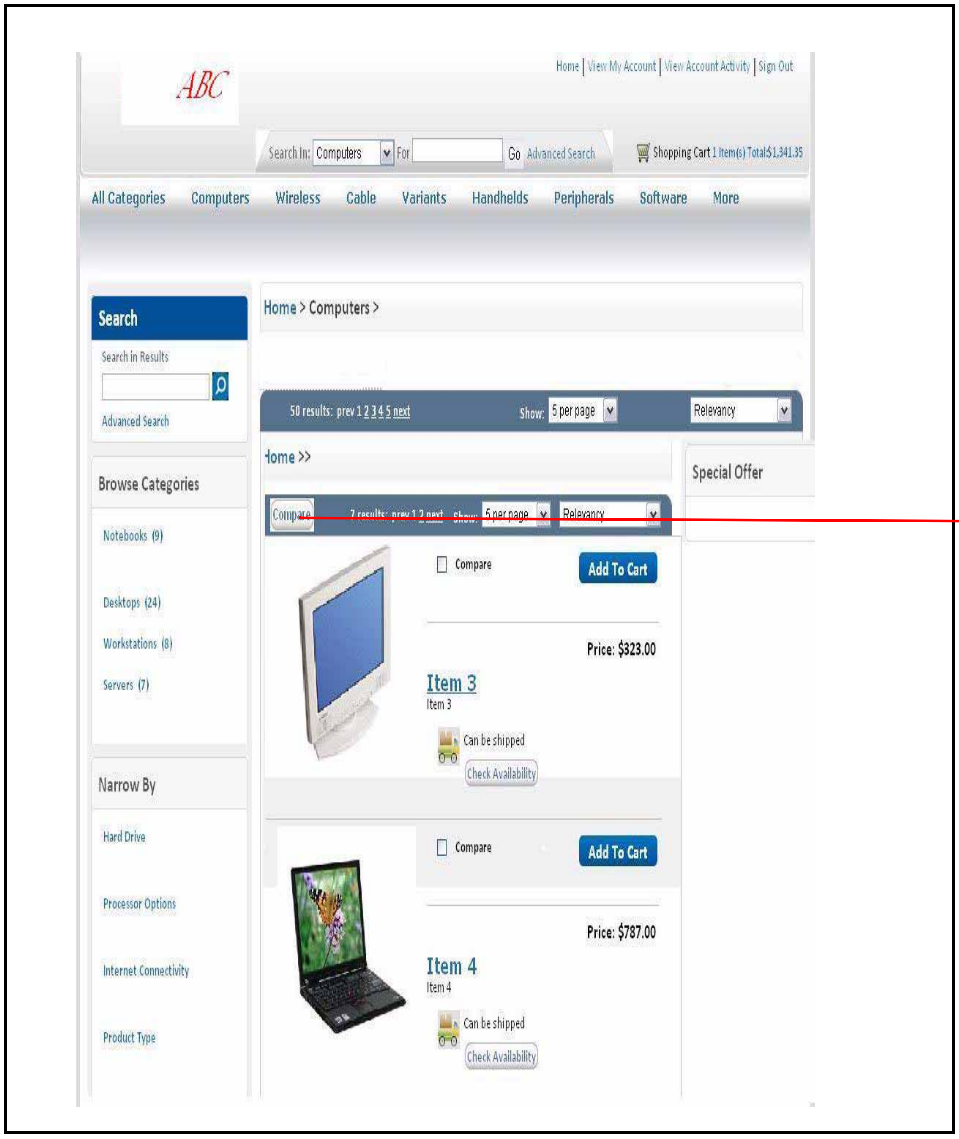
   Note: If this file is not present in the folder, run the `buildear` script to generate this file.
5. In `alternate_catalog_flow_struts.xml`, replace the `_replace_with_storefront_name_` attribute with the name of the storefront for which you want to configure an alternative view.
6. Navigate to the `<INSTALL_DIR>/repository/eardata/swc/extn/` folder and copy the `struts.xml.sample` file.
7. Navigate to the customization folder created by the end-user and paste the `struts.xml.sample` file.
8. Rename `struts.xml.sample` file as `struts.xml`

   Note: If this file is not present in the folder, run the `buildear` script to generate this file.
9. Edit `struts.xml` to include the `alternate_catalog_flow_struts.xml` file as follows:

   ```
   <include file="alternate_catalog_flow_struts.xml"/>
   ```
10. Perform the configurations required to set up the storefront. For more information about setting up a storefront, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
11. Deploy the customized JAR by performing the steps mentioned in the *Sterling Web: Customization Guide*.
12. After the server is started and the request is made for the configured storefront, the alternative page is displayed.

Note: If multiple storefronts have to be associated with the alternative page, the `_replace_with_storefront_name_` attribute and the subelements of this attribute must be replicated in the `struts.xml` file. This process must be repeated for each of the storefronts a user wants to create.

The following figure displays the alternative view of the Product List page:



Pagination

While browsing the catalog, if the number of products matching the search criteria is more than five, the Category Product List page displays the products across multiple pages. A user can navigate to any of the pages by clicking the corresponding page number hyperlink displayed below the list of products. The searchCatalogIndex API is called to retrieve the page of records to be displayed. The value of the PageSize attribute passed in the searchCatalogIndex API changes dynamically based on the value selected in the **Show** drop-down list. For more information about the searchCatalogIndextPage API, refer to the *Selling and Fulfillment Foundation: Javadocs.*

End-User Impact

This section explains the end-user impact of this functionality:

- In the catalog navigation bar, a user can view only those categories that have at least one product the user is entitled to within these categories or their subcategories.
- The **More** hyperlink in the catalog navigation bar is displayed only if the number of categories the user is entitled to view is more than seven. By default, seven categories are displayed. The remaining categories will be displayed on clicking **More**. The number of categories configured in the **SWC_NUM_OF_CATEGORIES_DISPLAY** rule, are displayed in the catalog navigation bar.
- The Category Product List page displays promotional offers only if promotions are assigned to the selected product category.
- The **Can be shipped** hyperlink is displayed for a product only if shipping is allowed for the product. For more information about how to define the fulfillment details for sourcing and transporting a product, refer to the *Business Center Item Administration Guide*.
- The **Check availability** hyperlink is not displayed for a variant product and for product that is configurable.
- The **Add to Cart** button is not displayed for a product with variations.
- The **Select** button is displayed only if the product has a variant product associated with it.
- The **Build** button is displayed for a product only if the product is configurable.
- The normal, condensed, or mini view of the result page are available only if the views are configured for the storefront.
- If the user selects the **Include superseded products** check box in the Advanced Search page, the search results will display the item and the item’s supersession product only if the original item is expired.
- A user can compare the products by dragging and dropping the products in the Drag to Compare basket and clicking the **Compare** button.
- The number of products listed in the search result page can be modified based on the options listed in the **Show** drop-down list.

Implementation

This section explains the configurations for this functionality:

- You must associate the image to be displayed for the product in the Category Product List page. This image is assigned through Business Center. For more information about defining the product main image, refer to the *Business Center Item Administration Guide*.
- You can configure the search terms that can be used to search for an item in the `CatalogSearchConfigProperties.xml` file.
- You must configure the **SWC_NUM_OF_CATEGORIES_DISPLAY** rule in the Channel Applications Manager to set the number of catalogs displayed in the user interface. By default the value of this rule is set as seven.
- You must associate entitlements to users to either grant or restrict the users’ access to the categories in the catalog navigation bar. For more information about entitlements, refer to the *Business Center Item Administration Guide*.

- You must associate a supersession product with a product to enable users to search for superseded products while performing an advanced search for a product. For more information about associating a supersession product, refer to the *Business Center Item Administration Guide*.

Reference Implementation

None.

Viewing Product Details

Users accessing the Sterling Web™ application may want to view the details of a product before adding it to their cart. For example, a user may want to view the details of a notebook before adding it to the cart.

Solution

In the Product Detail page, a user can view the details of a product. The Product Detail page comprises the following components:

- Header area: This area provides header-level information about a product such as the short description of the product, the product's image, a brief description of the product, availability information, characteristics of the product that indicate whether the product can be shipped or picked up from a store, actions that can be performed, for example, adding the product to a cart or configuring the product, and so on. The alert-level information is obtained from the getCompleteOrderDetails API. If the alert-level value is Low, a real-time call is made to check the inventory of the corresponding product. (The alert-level value can be Low, Medium or High as set in the Applications Manager.) For more information about configuring the alert level values, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*. The getCompleteItemList API verifies the cache for the availability of the product. If the product information is available, the cache returns In Stock. If the cache does not have information about the availability of the product and the **Check Real-time Inventory when Cached Inventory is Low** rule is enabled in the Channel Applications Manager, the findInventory API verifies the real-time inventory for the availability of the product. If the **Check Real-time Inventory when Cached Inventory is Low** rule is disabled, the availability of the product is displayed as N/A in the user interface. The getCompleteItemList API is called to display the details of the product in the Product Detail page.

  A user can view details and more images pertaining to a product in the Image Viewer by clicking the **More Images** hyperlink in the Product Detail page. The image of the product displayed in the Product Detail page is associated with the ItemImage1 asset type and is of 200 X 180 pixels. The ItemImageLarge1asset type is associated with the images displayed in the Image Viewer and is of 500 X 399 pixels.

  A user can also view the details of a product by clicking the **Data Sheet** hyperlink in the Product Detail page. The ItemDataSheet asset type is associated with the Data Sheet. The Data Sheet is stored in the `/swc` directory. A primary image is displayed for each product, and has Attribute asset type associated with it. For more information about assets, refer to the *Business Center Item Administration Guide*.

- Promotions column: This area displays the cross-sell and up-sell products associated with a product. The priority with which the cross-sell and up-sell products are displayed in the user interface is configured through the Business Center™ application. The products are displayed based on the priority set by the user. For more information about setting up the priority of the products to be displayed, refer to the *Business Center Item Administration Guide*. The products displayed under the Popular Accessories area in the Promotions column pertain to the cross-sell products. The products displayed under You might also consider area pertain to the up-sell products.

- Tabs: The following tabs are available in the Product Detail page. The tabs are displayed based on the type of product that is being viewed:

  - **Overview**: This tab displays a detailed description of a product.

- **Specifications**: This tab displays the attributes and values associated with a product.
- **Configuration**: This tab displays the details of configured and preconfigured products. The ExtractPicksFromBOM API is called to get the details of items picked for reconfigured or preconfigured products.
- **Kit Specification**: The tab enables a user to see the details of a physical kit product. This tab will only be visible for physical kit products.
- **Variation**: This displays the details of the variations assigned to a product. The classification of variant products is based on distinct attribute classifications. For example, a shirt may be available in different size variations such as Large, Regular, or Small, and color variations such as Black, White, and Red. The color variations denote the distinct attribute classifications. The image that is displayed when a variation is selected is associated with the AttributeValueImage asset type, and is of 16 X 16 pixels. For more information about assets, refer to the *Business Center Item Administration* Guide.
- **E-mail Page**: Sterling Web™ enables a user to send information pertaining to a product through an e-mail to a single recepient. The `YCD_ItemDetailsEmail.xsl` template is used for e-mailing product details. For more information about the `YCD_ItemDetailsEmail.xsl` template, refer to the "E-Mailing Cart and Order Details" topic. The address of the sender is based on the value of the **fromAddress** attribute specified in the `catalog-struts.xml` file. This file is located in the `com/sterlingcommerce/webchannel/catalog/catalog` folder.

End-User Impact

This section explains the end-user impact of this functionality:

- If a product is available, the availability information is displayed as In Stock. If none of the ship nodes have this product, the availability information is displayed as Unavailable.
- If the **Check Real-time Inventory when Cached Inventory is Low** rule is disabled, the availability information is displayed as N/A in the user interface.
- If a product is available after the current date, the availability information is displayed as Out of stock: Expected to be available on: <date>.
- The **Variations** tab is displayed only if a product has variations assigned to it. For example, if a product is available in three colors and three sizes, this tab will be displayed, showing the variants available for that product.
- The **Price** and **Build** buttons are not displayed for products with variant products.
- The **Pickup** and **Shippable** options are not displayed for a product with variant category.
- The **Build** button is displayed only if the value of IsConfigurable is set to Y in Business Center application. For more information about setting values, refer to the *Business Center Item Administration Guide*.
- The **Configuration** tab is displayed only if the value of IsPreConfigured is set to Y in Business Center application. For more information about setting values, refer to the *Business Center Item Administration Guide*.

Implementation

This section explains the configurations for this functionality:

- Ensure that the **Check Real-time Inventory when Cached Inventory is Low** rule is enabled in the Channel Applications Manager to enable the findInventory API to perform a real-time check on the inventory when the cache does not have the product availability information.
- You must set up appropriate entitlements to enable users to view the details of products. For more information about entitlements, refer to the *Business Center Item Administration Guide*.
- A user can view a product in the Sterling Web user interface if all of the following conditions are met:
  - The product is in **Published** status.
  - The product is associated with a category and the category is in **Published** status.
  - This product is included in the index that is generated once the product is in **Published** status.

  For more information about statuses associated with orders, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- If the value of IsConfigurable is set to Y in Business Center application, the **Build** button is displayed in the user interface. For more information about setting values, refer to the *Business Center Item Administration Guide*.
- If the value of IsPreConfigured is set yo Y in Business Center application the **Configuration** tab is displayed in the user interface. For more information about setting values, refer to the *Business Center Item Administration Guide*.

Reference Implementation

None.

Comparing Products

Sometimes users may want to compare a product with other products before taking a decision about which one to purchase. For example, a user may compare notebooks with similar specifications and decide to buy the one with lesser price.

Solution

Sterling Web™ enables a user to compare a set of products based on their attributes. A user can select and compare the products and with specific attributes and specification only if the **Specification** and **For Comparison** usages are assigned to the attributes in the Business Center Application. The usages can be assigned to the attribute at a category level. For more information about how to assign a usage to an attribute, refer to the *Business Center Item Administration Guide*. In the Product List page, a user can compare the products by dragging and dropping it in the Drag to Compare basket and clicking the **Drag to Compare** hyperlink. The Drag to Compare basket displays the number of products added for comparison. A user can expand and collapse the attribute panels in the Product Comparison page.

If the user has configured the alternative view for the Product List page, the user can compare the products by selecting them and clicking the **Compare** hyperlink. For more information about configuring the alternative view of the Product List page, refer to the *Sterling Web Customization Guide*. However, only the first four products will be compared. A user can select and remove the products in the Product Comparison page. If a product that was added for comparison is removed, the product in queue will replace the product that was previously added for comparison. If a user has previously included, say, four products for comparison, and adds four more products now, the items added recently will be listed above the products that are selected for comparison.

A user can select a product only once for comparison. For example, a user may select Product A and Product B for comparison. The user can include another product, Product C to this list, but cannot include Product A again. The getCompleteItemList API is used to display the key product attributes that represent the comparable attributes, and the differences between the attributes of the products selected for comparison.

A user can click the **Show Differences** button to view only the differences between the selected products. By default, all the attributes set for comparison for the selected products are displayed on the Product Comparison page. By default the **Show Differences** button is displayed on the Product Comparison page. From the Product Comparison page, a user can also add the selected products to the cart and build the configurable products. The `ShowAttributesWithDifferentValuesOnly` attribute in the getCompleteItemList API is used for displaying the differences in the attributes across these products chosen for comparison.

The primary image associated with a product, which is 100 X 80 pixels, is displayed for a product in the Product Comparison page. This image is assigned through Business Center. For more information about assigning an image to a product, refer to the *Business Center Item Administration Guide*.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can compare products pertaining to a category only if the **Specification** and **For Comparison** usage are assigned to the attributes in the Business Center Application. For more information about how to assign a usage to an attribute, refer to the *Business Center Item Administration Guide*.
- A user can remove a product from the comparison list by clicking the **Remove** hyperlink.
- The **Add to Cart** button is not displayed for a product with variations.
- The **Select** button is displayed only if the product has variant products associated with it.
- The **Price** and **Build** buttons are not displayed for products with variant category.
- By default, the **Show Differences** button is displayed on the Product Comparison page.
- The **Show All** button is displayed when the user is viewing the differences between the selected products.
- A user can compare the products by dragging and dropping the products in the Drag to Compare basket and clicking the **Drag to Compare** hyperlink. Based on the configured page, a user can compare the products by selecting them and clicking the **Compare** button.

Implementation

This section explains the configurations for this functionality:

The sequence of the attribute groups displayed in the Product Comparison page is based on the configurations set up in the Business Center application. For more information about setting attributes, refer to the *Business Center Item Administration* Guide.

Reference Implementation

None.

Creating a Cart

A user may want to create a new shopping cart to use it as a starting point for placing an order. For example, a user, who already has a shopping cart, may want to create a new cart to place an order for some products the user wants to be delivered to a different address.

Solution

Sterling Web™ creates a cart implicitly under the following scenarios:

- A user adds products to a cart and no cart exists in the context of the user.
- A user clicks on the mini cart hyperlink to navigate to the Cart Detail page and no cart exists in the context of the user.
- A guest user accesses the Sterling Web application.
- When a buyer user logs in to the application, and there are no existing carts for that user.

Additionally, a buyer user can create a new cart from the Cart List page by clicking the **Create New Cart** button. The createOrder API is called to create the new cart. The newly created cart is saved to the database with the **Last Modified Date** and **Creation Date** as the current date and time. The default attributes associated with a cart are based on the template defined for the cart in the Applications Manager. For more information about defining templates, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

End-User Impact

This section explains the end-user impact of this functionality:

- The **Create New Cart** button is displayed only if the user has the necessary permissions.
- A user can create a cart and checkout a cart only if the appropriate order modification rules are configured in the Applications Manager. For more information about configuring order modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.

Implementation

This section explains the configurations for this functionality:

- You must assign the necessary permissions to the corresponding user groups to enable them to create a new cart explicitly by using the **Create New Cart** button. By default, permissions are assigned to all the user groups except the GUEST-USER group. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must configure the appropriate order modification rules to enable a user to create and checkout a cart successfully. For more information about configuring order modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.

Reference Implementation

None.

Searching for a Cart

A user may want to search for a particular cart by providing appropriate search criteria. For example, a user may want to search for a cart that contains a particular product.

Solution

Sterling Web™ enables a user to search for a cart from the Shopping Cart List page. A user can search for a cart using the following options:

- Quick search: A user can provide the Order Number, Cart Name, or the Product ID of a product in a cart, and search for the cart.
- Advanced Search: A user can provide more restrictive criteria and search for a corresponding cart.
- Show All: A user can use this option to view all the available carts. If any existing search settings are available, those settings will not be considered when the results are displayed.

If a user does not enter any search parameter and conducts a search, all the available carts will be displayed. A user can also use wild card characters to search for a cart. For example, if the user is looking for a cart that contains a product with XYZ123 as Product ID, the user can perform a quick search for the cart by providing XY* as a search term.

The getOrderList API is called to retrieve the list of carts matching the search criteria provided by the user. The carts are displayed, sorted by the Last Modified value, in descending order. A user can sort the list in either ascending or descending order based on the following options:

- **Order No**
- **Cart Name**
- **Last Modified**
- **Creation Date**

When a wild card character is used to search for a cart, the value of the `QryType` attribute in the input XML file of the getOrderList API is set to FLIKE. If a wild card character is not used, the value of the `QryType` attribute is set to EQ. For more information about defining complex queries, refer to the *Selling and Fulfillment Foundation: Customizing APIs Guide.*

Pagination

If the number of carts matching the search criteria is more than five, the Shopping Cart List page displays the results across multiple pages. A user can navigate to any of the pages by clicking the page number hyperlink at the bottom of the page. The getPage API is called to retrieve the records to be displayed. For more information about the getPage API, refer to the *Selling and Fulfillment Foundation: Javadocs*.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can search for carts only if the user has the necessary permissions.

- A user can search for the user’s cart and other user’s cart only if the appropriate data access policy rules have been configured for the user group to which the user belongs. For more information about data access policy rules, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Implementation

This section explains the configurations for this functionality:

- You must configure appropriate data access policy rules for the appropriate user groups to enable users to search for their cart and carts pertaining to other users. For more information about data access policy rules, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must assign the necessary permissions to the corresponding user groups to enable them to search for carts. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

Viewing Cart Details

A user may sometimes want to view the details pertaining to a cart. For example, the user may want to see the products that have been added to the cart, or the availability information pertaining to the products in the cart.

Solution

Sterling Web™ enables a user to view the details pertaining to a cart. A guest user can view it's cart in the Sterling Web user interface till the cookies in the browser are valid. By default, the cookies are valid for one year starting from the current date. When the cookies expire, the cart is no longer available to the guest user. However, this cart exists in the database.

A user can view line-level information such as the quantity of the products in the cart, products that have not been configured, products that can be reconfigured, complementary products associated with the line products, UOM, availability, price, adjustments, line total, and so on. A user can also view header-level information such as the total price of the order, order total adjustments, if any, made to the cart, tax applied to the cart, and so on.

Note: A user cannot select the value of UOM from the Sterling Web user interface. This value is configured through Business Center Application. For more information about how to create a regular item, refer to the *Business Center: Item Administration Guide*.

The getCompleteItemList API is called to retrieve the cart details and the complementary products associated with a particular product. The getPossibleSchedules API is called to perform a real-time availability verification for a product if the availability information is not returned by the getCompleteItemList API. The YFSRecalculateHeaderTaxUE user exit is called to calculate header level taxes applied on a cart and the YFSRecalculateLineTaxUE user exit is called to calculate line level taxes applied on a cart. The YPMCalculateShippingChargeUE is called for calculating the shipping cost. For more information about these user exits, refer to the *Selling and Fulfillment Foundation: Javadocs*.

There may be differences in the price that is displayed on the Cart Details and the Sterling Configurator page because pricing rules will be applied on the Cart Details page. This implies that if there is any item level adjustment or line level adjustment applicable on a configurable product, the adjustment is not displayed in the Sterling Configurator page. After the product is configured in Sterling Web and added to the cart, the item level adjustment or line level adjustment, if applicable, is displayed in Cart Details page.

Additionally, a user can perform the following tasks from the Cart Detail page:

- Add or view notes: Users can add a note about a cart or view the existing notes. The getcompleteOrderdetails API is called to retrieve and display the details in the Notes page. If a user adds notes, the changeOrder API updates the information in the **Notes** page. A note entered from Sterling Web can be viewed by users if they have the necessary permissions. A Sterling Web user cannot view an internal note entered using the Sterling Call Center and Sterling Store application. For more information about adding or viewing notes, refer to the "Adding or Viewing a Note" topic.
- Print Cart Details page: Users can print the Cart Detail page. The getCompleteOrderDetails API is called to print the information in the Cart Detail page.
- E-mail cart details: A logged in user can e-mail the details pertaining to a cart. For more information about e-mailing cart details, refer to the "E-Mailing Cart and Order Details" topic.

- Add products to cart: Users can add products to a cart from the Quick Add area. The getCompleteItemList API is called to validate the Product ID of the products. If any of the products has been superseded by another product, the getCompleteItemList API is called to retrieve the new product. The changeOrder API is called to add the products to the cart. A user can select to substitute a product with either the preferred substitute or the regular substitute through the Business Center application. A superseded product is a product that has been configured to replace another product whose effective end date has reached. For example, you can configure a product A to be superseded by another product B when the effective end date of A expires. For more information about associating a supersession product, refer to the *Business Center Item Administration Guide*. If a user does not provide an Item ID and clicks the **Add to Cart** button, client side validation is carried out and an error is displayed. The **Validate Item** rule enables Sterling Web to validate whether an item is valid for a cart when the item is added to a cart. For more information about setting the **Validate Item** rule, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- Remove products from cart: Users can remove products from a cart. The changeOrder API is called to remove the products from the cart. If a user selects the **Remove** button on the Cart Details page without selecting any product in the cart, the corresponding jsp is validated and an error is displayed.
- Enter zip code for in-store pickup: Users can enter the postal zip code of the location from where they want to pick up a product. If a user tries to checkout without entering a zip code, JavaScript validation is performed and an error is displayed.
- View product details: Users can view the details page of a product if the product exists in the product catalog.
- Configure or reconfigure product in cart: Users can configure or reconfigure a product in a cart if the product is preconfigured or is configurable. For more information about configuring and reconfiguring a product, refer to the “Product Configuration” topic. The validateItemForOrdering API performs item validation on an order. Whenever a product is added to the cart, this API validates whether the product has passed the expiry date and product is a configurable product.
- View and add alternative products: A user can view the alternative products for a product in a cart and these products to cart, if there are any alternative products associated with the product and product is out of stock. The getAssociationRelationshipList API returns the relationship between products defined at the enterprise level and the enterprise-specific association relationships that match the input criteria. The getCompleteItemList API is used to get the products that are configured as alternative products.
- Modify cart details: A user can modify a cart’s name and the cart’s currency. The getCurrencyList API is called to retrieve the available currencies for the user. The changeOrder API is called to save the updated cart details.
- View line adjustments: In the Adjustments column, a user can view the adjustments made to a line if any discounts or coupons applied to a line. For example, if a discount of $50 is provided on a line item, this value will be displayed in the Adjustments column. The getCompleteOrderDetails API is called to display the Line Adjustments dialog box.
- View order total adjustments: In the Order Total Adjustments a user can view the adjustments made to an order if any discounts or coupons are applied to an order. For example, if a discount of $100 provided on an order, this value will be displayed in the Order Total Adjustments column. The getCompleteOrderDetails API is called to display the Order Total Adjustments dialog box.
- Updating the changes: The **Update** button enables a user to update the changes to the cart. The changeOrder API is called to save the updated cart information.
- Continue shopping: A user can continue to shop after viewing the details of the cart. The searchCatalogIndex API is called to display the Shopping Cart List page.

Note: Inventory is not reserved for the products that are added to the cart.

End-User Impact

This section explains the end-user impact of this functionality:

- The **Notes** hyperlink is not displayed for a guest user.
- The **Checkout** button is active only if there is at least one line item in the cart.
- The **More** hyperlink is displayed only if more than three complementary products have been specified for a product.
- The **Adjustments** hyperlink for a line items is displayed only if one or more line items discounts are applied to the corresponding line item. For more information about adjustments, refer to the *Business Center Pricing Administration Guide*.
- If the IsShippable and IsPickable flags are set to Y at the order line level, the Pickup and Ship options are displayed in the drop-down list. For more information about pickup tasks, refer to the “In-Store Pickup” topic. If the product can neither be picked nor shipped, the drop-down list is not displayed in the user interface. For more information about configuring Pickup and Ship options, refer to the *Business Center Item Administration Guide*.
- The **View alternatives** hyperlink is displayed if a product is not in stock and the corresponding product association rule, and relationship type is enabled in the Applications Manager. For more information about configuring relationships, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

  Note: Sterling Web does not support item associations by category.
- The **Build** hyperlink is displayed only for bundle the products that have not been configured.
- The **Reconfigure** hyperlink is displayed only for the products that have been configured.
- The **Order Total Adjustments** hyperlink is displayed only if more than one discount is applied to an order.
- The **Shipping Costs** hyperlink is displayed if more than one shipping discounts is applied to the order. The YPMCalculateShippingChargeUE is called for calculating the shipping cost.
- The Availability column display In Stock if either of the following conditions are met:
  - If the availability cache is in use, and the availability value in the cache for the product is not low.
  - If the availability cache is not in use, or the availability value in the cache for the product is low, a real-time availability verification is performed by calling the getPossibleSchedules API. Each of the elements in the API reference a ShipNode and the corresponding ProductAvailDate, ShipDate, and Quantity. If more than one node has the product, and the total quantity in stock is more than the required quantity.
- The Availability column displays the value Unavailable if none of the nodes have stock and will not have stock in the future too.
- The Availability column displays the value Backordered, Expected to be available on: <date>, if none of the nodes have the products in stock, more than one node will have the products in stock in future, and the total quantity that is expected to be in stock is greater than the required quantity.
- A user can view the details of a cart only if the user has the necessary permissions.
- The **Email Page** hyperlink is not displayed if the user is a guest user.

- A user can enter ten digits followed by a decimal, and four digits after the decimal in the **Qty** text box. A user can enter double quantity, if the yfs.install.displaydoublequantity property is set to Y in the `yfs.properties` file. If the value of this property is set to N, the user will not be able to enter any digit after the decimal.
- A user can click the **Edit cart details** hyperlink and modify the name of the cart only if the appropriate order modification rule is configured.
- The **Update** button is active only if there is at least one line item in the cart.
- If a dynamic physical kit, bundle item, delivery service, or provided service is associated with a cart, the cart is not displayed in the Sterling Web user interface.
- Users cannot view internal notes entered using the Sterling Call Center™ and Sterling Store™ applications.
- A user can modify the currency only if the appropriate order modification rules are configured in the Applications Manager. For more information about configuring order modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- In the case of a physical kit, the **Min** and **Max** rules are not applied for component items. These rules will be applied only when the component items of a physical kit are individually added to a cart.
- In the case of a physical kit, the status of component items are not taken into consideration. For example, if one of the component items in a physical kit is either in **Held** or **Published** status, an order can be placed on the entire physical kit. However, when you place individual orders for each component item of a physical kit, the status of component items are taken into consideration.
- To enables Sterling Web to validate whether a product is valid for a cart when the product is added to a cart, ensure that the Validate Item rule is set up in the Applications Manager. For more information about setting the **Validate Item** rule, refer to the *Sterling Distributed Order Management: Configuration Guide*.

Implementation

This section explains the configurations for this functionality:

- To display the complementary products available for a product, the product must be out of stock and the corresponding item association rules and relationship types must be enabled in the Channel Applications Manager. For more information about these rules, refer to the *Channel Applications Manager: Configuration Guide*.
- To display the alternative products available for a product, ensure that the corresponding product association rules and relationship types are enabled in the Channel Applications Manager. For more information about these rules, refer to the *Channel Applications Manager: Configuration Guide*.
- To enable validation whether a product is valid for a cart when the product is added to a cart, configure the **Validate Item** rule in the Applications Manager. For more information about setting the **Validate Item** rule, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- To validate configurable items when they are added to a cart, ensure that the **Validate Configurable Items When Viewing Cart** rule is enabled in the Channel Applications Manager. For more information about these rules, refer to the *Channel Applications Manager: Configuration Guide*.
- To enable in-store pickup of products and display the **Zip Code for Store Pickup** text box, ensure that the **Enable Store Pickup** rule is enabled in the Channel Applications Manager. For more information about this rule, refer to the *Channel Applications Manager: Configuration Guide*.

- To enable a user to enter ten digits followed by a decimal, and four digits after the decimal in the **Qty** text box, ensure that the yfs.install.displaydoublequantity property is set to Y in the `yfs.properties` file. By default, a user can enter a maximum of four digits after the decimal.
- To specify whether inventory checks should be performed for products while a user creates and views carts ensure that the **Check Inventory On Cart** rule is configured in the Channel Applications Manager. For more information about setting the **Check Inventory On Cart** rule, refer to the *Channel Applications Manager: Configuration Guide.*
- Sterling Web enables you to configure the number of alternative items and complimentary items that must be displayed in the Alternative Items and Complimentary Items dialog box, respectively. For more information, refer to the *Channel Applications Manager: Configuration Guide*.
- Sterling Web enables you to configure the order statuses for which modification of the order is allowed. For more information about configuring order modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- You must assign permissions to user groups to enable them to view cart details. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must implement the YFSRecalculateHeaderTaxUE user exit to calculate header level taxes applied on a cart and the YFSRecalculateLineTaxUE user exit to calculate line level taxes applied on a cart. For more information about these user exits, refer to the *Selling and Fulfillment Foundation: Javadocs*.
- To configure the charge name to which the value of the shipping cost is returned, ensure that the **Charge Name For Shipping** rule is configured in the Applications Manager. For more information about configuring the **Charge Name For Shipping** rule, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- Sterling Web enables you to modify the currency of a product only if the CHANGE_CURRENCY modification rule is set in the Applications Manager. For more information about configuring order modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide.*
- The **Build** or **Reconfigure** hyperlink is displayed based on the following algorithm:

IF the IsConfigurable flag is set to **N**

THEN the item is not configurable - do not show the **Configuration** hyperlink

ELSE

IF the IsPreConfigured flag is **N**

THEN

IF the item has a ConfigurationKey

THEN the item has been configured by the user - show **Reconfigure** hyperlink

ELSE it has not - show **Build** hyperlink

ELSE (the IsPreConfigured flag is set to **Y**) - show **Reconfigure** hyperlink.

For more information about the flags mentioned in the algorithm, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Note: Apart from modifying currency of a product, modifying name of the cart, and configuring the order statuses, order modification rules do not govern any action displayed on the jsp for the orders in the Draft status.

Reference Implementation

None.

Activating a Cart

A user may have multiple carts. However, a user can have only one active cart at a time, which can be accessed through a link in the header in the user interface. The active cart is the cart a user owns. The user can use this cart to add products, edit products, or place an order for. However, the user can edit or place an order for carts owned by other users too. For example, a user may have two carts, Cart1 and Cart2, of which Cart1 is the user's active cart. The user, may, however want to activate Cart2 because the user wants to navigate quickly to Cart2 and add products to that cart, the next time the user logs in to Sterling Web™.

Solution

By default, if a user has multiple carts, the most recently modified cart is set as the user's active cart, and this is the only cart that the user can add products to and place an order for. However, a user can place an order by adding products to carts pertaining to other users through Quick Add. An active cart is displayed as a mini cart in the application navigation bar. This enables a user to navigate quickly to the cart that was last modified.

Sterling Web enables a user to activate a cart from the Shopping Cart List page. When a user selects a cart and clicks the **Make Active Cart** hyperlink in the Shopping Cart List page, the changeOrder API is called to activate that cart. When this API is called, the MODIFYTS column in the YFS_ORDER_HEADER table is updated with the current timestamp.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can activate a cart only if the user has the necessary permissions.
- A user can view the **Make Active Cart** hyperlink for the carts that belong to the user. A user can view other user's cart only if appropriate data access policy rules have been configured. For more information about defining data access policies, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Implementation

This section explains the configurations for this functionality:

- You must configure the data access policy rules to enable a user to view a cart belonging to the user and the carts belonging to other users belonging to the same organization or its child organizations. By default, data access policy rules have been configured to enable a user to activate only those carts that belong to the user. For more information about defining data access polices, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must assign permissions to the corresponding user groups to enable them to activate a cart. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

Viewing a Mini Cart

Users may sometimes want to navigate quickly to their active cart or get a quick view of the products in their active cart before placing an order. For example, a user may want to know the number of major line items in the user's active cart.

Solution

Sterling Web™ provides the mini cart functionality that enables users to get a quick view of the contents of their shopping cart without navigating to the Cart Detail page. Users can view the contents of their default cart from any page. For example, when a user is performing an advanced search for a product in a catalog, the user can also view the contents of the default cart, add products to the cart, or delete products from the cart in the Catalog Advanced Search page. A user can move the pointer over the mini cart hyperlink and view the expanded mini cart. If a user has multiple carts, the user's active cart is displayed as the mini cart. As part of this functionality, the application navigation bar displays the order subtotal and the number of major line products in the cart as hyperlinks to enable a user to navigate quickly to the Cart Detail page. The number of line products displayed in the mini cart is based on the value of `miniCartListMaxElements` attribute specified in the `header.jsp` file. This attribute is passed to the header action of the page on which the mini cart is displayed. If the `miniCartListMaxElements` attribute is not passed, a maximum of 5 line items are displayed in the mini cart. To view the remaining products, click the **More** hyperlink in the expanded mini cart. The getCompleteOrderDetails API is called to display all the items on the cart details page. A user who has logged in can also click the **Cart Management** hyperlink and view the Shopping Cart List page.

Sterling Web also enables users to expand their mini cart and view line item information such as the item IDs of the major line items in the cart, line quantity, and line total. The getCompleteOrderDetails API is called to retrieve the information about a cart and its line products.

From the expanded mini cart, a user can perform the following tasks:

- Navigate to the Product Detail page of a line item
- Navigate to the Shopping Cart List page
- Delete line item
- Modify line item quantities

The **Update** button enables a user to update the changes to the cart. The changeOrder API is called to save the updated cart information.

End-User Impact

This section explains the end-user impact of this functionality:

- The **Update** button is displayed in the expanded mini cart only if there is at least one item in the cart.
- The **More** hyperlink is displayed in the expanded mini cart only if there are more than 5 line items.
- The **Cart Management** hyperlink is not displayed if the user is a guest user.
- The contents of the expanded mini cart are not displayed on the cart detail page of the expanded cart.

- The number of line items displayed in the mini cart is based on the value of `miniCartListMaxElements` attribute specified in the `header.jsp` file. If the `miniCartListMaxElements` attribute is not passed, a maximum of 5 line items are displayed in the mini cart.
- A user can enter ten digits followed by a decimal, and four digits after the decimal in the **Qty** text box. A user can enter double quantity, if the yfs.install.displaydoublequantity property is set to Y in the `yfs.properties` file. If the value of this property is set to N, the user will not be able to enter any digit after the decimal.
- A user can view the mini cart only if the user has the necessary permissions.

Implementation

This section explains the configurations for this functionality:

- You must assign the necessary permissions to the corresponding user groups to enable them to view the mini cart. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide.*
- To enable a user to enter ten digits followed by a decimal, and four digits after the decimal in the **Qty** text box, ensure that the yfs.install.displaydoublequantity property is set to Y in the `yfs.properties` file. By default, a user can enter a maximum of four digits after the decimal.

Reference Implementation

None.

Deleting a Cart

After creating a cart, a user may want to delete the cart. For example, a user may either have a change of mind and decide against purchasing any of the products in the cart, or may have found that the products are available for a lesser price elsewhere and may want to delete the cart.

Solution

Sterling Web™ enables a user to delete a cart from the Shopping Cart List page. If a user has more than one cart, the user can delete only one cart at a given instance. The deleteOrder API is called when a user selects a cart, clicks the **Delete** button, and clicks the **Ok** button in the confirmation window to delete the cart. If a user does not select any cart in the user interface and clicks the **Delete** button, a JavaScript validation is performed, and an error is thrown.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can delete a cart based on the order status modification rules that have been configured.
- A user can delete a cart only if the user has the necessary permissions.

Implementation

This section explains the configurations for this functionality:

- You must configure the order status modification rules appropriately to enable a user to delete a cart. For more information about defining modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- You must assign permissions to user groups to enable them to delete a cart. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

In-Store Pickup

While placing an order, a user may decide to pick up the products from a store instead of having them shipped. In such a situation, the user may want to know the stores from which the products can be picked up. For example, a user may want to know the closest store from the user's location from which the products can be picked up.

Solution

Sterling Web™ enables a user to see whether a product can be picked up from a store while placing an order. If a product can be picked up from a store, the user can view the stores from where the product can be picked up, verify the availability of the product at each store, and select the store from where the product will be picked up. A store is searched based on the value of latitude and longitude specified in the address, configured for the store in Applications Manager. For more information about configuring address, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*. The country code specified as the part of the address is obtained from the shipping address specified for the order. However, if the shipping address does not have country code, it is obtained from the locale configured for that organization. A user must have a region schema defined for the region where in-store pickup is allowed. For more information about creating a region schema, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

The getCompleteOrderdetails API is called to display the product description and quantity of the corresponding product on the **Select stores to pickup items** page. In the **Select store(s) for pickup item(s)** area, the getCommonCodes API is called to display the values in the **Find stores within** drop-down list and the **Country** drop-down list. The products and the quantity of products that a user has selected for pickup are displayed in the **Items to pick up** area. For every item line that a user wants to pick up from a store, the list of stores from where the product can be picked up is displayed in a drop-down list along with the availability status of the product. The changeOrder API is called when the user selects a store from the **Select a store** drop-down list and clicks the **Next** button. The products and the quantity of products that a user wants to be shipped are displayed in the **Other Items** area. The changeOrder API is called when the user clicks the **Next** button. If a user does not select a store on the **Select stores to pickup items** page, the corresponding jsp is validated and the user is not permitted to proceed further.

The list of stores is displayed based on the postal code provided by the user, and the distance the user is willing to travel. The getSurroundingNodeList API is called to retrieve the list of stores based on the zip code and distance specified by the user. The findInventory API is called to verify the availability of the product at each store, and the date on which the product can be picked up. The number of stores that will be displayed is based on the value of the `maxPickupStoreNumber` parameter. For example, if the getSurroundingNodeList API returns 12 stores, and the value of the `maxPickupStoreNumber` parameter is set to ten, the findInventory API will be called to verify the availability of the product in the first ten stores. The value of YCD_STORE_SEARCH is used as the sourcing rule to call the findInventory API.

The following table lists the rules that you must configure in order to be able to use this functionality:

Rule Name	Purpose
Enable Store Pickup	If this rule is enabled, a user can choose to pick up line products from a store. By default, this rule is disabled.

Rule Name	Purpose
Distance Radius To Consider When User Searches For Stores	This rule is used to define the distance to be considered when searching for a store location. By default the value of this rule is set to 25. The YFS COMMON CODES are defined in the database of Selling and Fulfillment Foundation.
Distance UOM To Consider When User Searches For Stores	This rule is used to define the unit of measure (UOM) to be used when defining the distance to consider while finding a store location. By default, the value of this rule is set to Mile. The YFS COMMON CODES are defined in the database of Selling and Fulfillment Foundation.
Maximum Number Of Stores To Display	This rule is used to define the maximum number of stores that should be displayed for every product that can be picked up. By default, the value of this rule is set to 10.
Enable Store Pickup For Future Date	This rule is used to allow future pickup of products. For example, if a products not available at a store currently, but will be available in the future, this rule can be used to allow a user to view when the product will be available in the future. By default, this rule is enabled.
Standard Search Fulfillment Type	This rule is used to define the fulfillment type to be used in the sourcing rule that determines the correct set of nodes to be considered while searching for stores.

The getRuledetails API is called to retrieve the values that are set for these rules. For more information about configuring these rules, refer to the *Channel Applications Manager: Configuration Guide*.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can pick up a product from a store only if the product has been configured for being picked up.
- If a user modifies the delivery method of a product from shipping to pick up, the product is displayed dynamically from the **Other Items** area to the **Items to pick up area**. Similarly, if a user modifies the delivery method of a product from pick up to shipping, the product is displayed dynamically from the **Items to pick up area** to **Other Items**.
- To enable a user to use this functionality, you must configure the store pickup rules. For more information about configuring in-store pickup rules, refer to the *Channel Applications Manager: Configuration Guide*.
- The availability information pertaining to a product will be displayed as In stock if the product is available in a particular store, and Out of stock if the product is currently unavailable but the product will be available in the future. In any other scenario, the store will not be displayed.
- The verification of the availability of the product in the future is done only if the **Enable Store Pickup For Future Date** rule is enabled in the Channel Applications Manager. For more information about configuring this rule, refer to the *Channel Applications Manager: Configuration Guide*.
- A user can view the regions configured to allow in-store pick up only if the region schemas are defined. For more information about creating a region schema, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Implementation

This section explains the configurations for this functionality:

- You must assign the necessary permissions to the corresponding user groups to enable users to pickup products. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- To enable a product to be picked up from a store, you must set the **Pickup Allowed** rule to Yes for that product. For more information about how to define the fulfillment details for sourcing and transporting a product, refer to the *Business Center Item Administration Guide*.
- You must create a sourcing rule for the YCD_STORE_SEARCH fulfillment type. For more information about sourcing and scheduling, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- To display the regions where in-store pickup is allowed, a user must have a region schema defined. For more information about creating a region schema, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

The Checkout Functionality

Before placing an order, a user may want to reconfirm that the important information provided by the user such as the address and payment information, is correct. For example, a user may want to ensure that the correct Ship-to address has been provided for the order to be shipped to the correct address.

Solution

Sterling Web™ provides the checkout functionality that enables users to reconfirm that the correct address and payment information has been provided before they place an order. Sterling Web supports both the single-step and multi-step checkout processes. The single-step checkout process must be selected if the user does not want to modify the address and payment information specified in the cart. From the Order Summary page, the user can view the address and payment information, and complete the checkout process in a single step. In the multi-step checkout process, a user can use the Order Summary page to view and modify the address and payment information, and place the order. If the user selects the single step checkout process, clicking the **Next** button on the Select stores to pickup items page takes the user to the Order Summary page. If the user has select the multi-step checkout process, clicking the **Next** button on the Select stores to pickup items page takes the user to the Sold To page. When a cart is created, the address specified by the corresponding user is considered during the checkout process. If required, the user can modify this address or select a different address.

The checkout process involves the following steps:

1. Provide Sold To Information
2. Provide Ship-to Information
3. Provide Bill To Information
4. Provide Payment Information
5. Confirm Order

Provide Sold To Information

The first step in the checkout process is providing the Sold To address information for the corresponding order. The Sold To page displays the current Sold To address and the user's addresses. The getCompleteOrderDetails API is called to display the Sold To address information. The address book displays addresses specified for the user and the user's organization. By default, the default address defined for the buyer user is considered. However, if no default address is specified for the user, the default address provided for the user's organization is considered.

It is not mandatory to provide Sold To address. The getCustomerDetails API is called to display the addresses specified for the user in the address book.

The user can perform the following tasks as part of this step:

- Use another address as the Sold To address for the order: The user can click the **Use this address** hyperlink below the corresponding address to make that address the default Sold To address for the order. When the user clicks this hyperlink, the new address is displayed in the **Sold-to address for this**

order area. When the user clicks the **Continue** button, this Sold to address is saved. The changeOrder API is called to save the Sold To address.

- Modify the Sold To address of the order: The user can click the **Edit address** hyperlink to modify the details of the default Sold To address. The getCompleteOrderDetails API is called to retrieve the Sold To address information. The getRegionList API is called to retrieve the information to be displayed in the **State/Province** drop-down list.

  When the user clicks the **Continue** button, the modifications are saved. The verifyAddress API is called to validate the address information provided. After the address information is validated, the changeOrder API is called to save the updated Sold To address information. After modifying the Sold To address information, the user can choose to save the updated address information in the address book. The modified address is saved as a new address in the address book.

Provide Ship-to Information

After providing the Sold To address information, the next step in the checkout process is providing the Ship-to address information and shipping preferences. It is mandatory to provide the Ship-to address. If Ship -to address is not specified, the Bill-to address is used. The Ship-to page displays the default Ship-to address and the addresses specified for the user. The getCompleteOrderDetails API is called to display the Ship-to address. The getCustomerDetails API is called to display the addresses specified for the user.

The user can perform the following tasks as part of this step:

- Use another address as the Sold To address for the order: The user can click the **Use this address** hyperlink below the corresponding address to make that address the Ship-to address for the order. When the user clicks this hyperlink, the new address is displayed in the **Main shipping address** text area. When the user clicks the **Continue** button, this Ship-to address gets updated. The changeOrder API is called to save the updated Ship To address in the address book.
- Modify the Ship-to address information for the order: The user can click the **Edit address** hyperlink to modify the details of the Ship-to address. The getCompleteOrderDetails API is called to retrieve the Ship-to address information. The getRegionList API is called to retrieve the information to be displayed in the **State/Province** drop-down list. The verifyAddress API is calls the YCDVerifyAddressWithAVSUE user exit to validate the address information provided by the user. For more information about the YCDVerifyAddressWithAVSUE user exit, refer to the *Selling and Fulfillment Foundation: Javadocs*. After the address information is validated, the changeOrder API is called to save the updated Ship-to address information. After modifying the Ship-to address information, the user can choose to save the updated address information in the address book. The modified address is saved as a new address in the address book. When the user clicks the **Continue** button, this Ship-to address is saved.
- Provide line-level Ship-to address information: The user can indicate that the line items should be shipped to multiple addresses by selecting the **Ship-to multiple addresses** check box. The getCompleteOrderDetails API is called to retrieve the line-level Ship-to address information. To provide line-level Ship-to address information, the user can perform one or more of the following tasks for every line item:
  - Use main Shipping address: If no Ship-to address is specified for a line item, the user can click the **Use Main Shipping Address** hyperlink to indicate that the line item should be shipped to the default Ship-to address of the order.
  - Provide a new Ship-to address: If no Ship-to address specified for a line item, the user can click the **New Address** hyperlink to provide a new Ship-to address. The getRegionList API is called to retrieve the information to be displayed in the **State/Province** drop-down list in the Edit ship-to

    address dialog box. The verifyAddress API is called to validate the address information provided by the user. The user can choose to save the new address in the address book. The manageCustomer API is called to save the Ship-to address in the address book.

  - Select a Ship-to address from the address book: If no Ship-to address is specified for a line item, the user can click the **Select from Address Book** hyperlink to select a Ship-to address from the address book. The getCustomerDetails API is called to retrieve the list of addresses from the address book.
  - Modify the details of the Ship-to address: If a Ship-to address is specified for a line item, the user can modify the details of the Ship-to address by clicking the **Edit Address** hyperlink. The getRegionList API is called to retrieve the information to be displayed in the **State/Province** drop-down list in the Edit ship-to address dialog box. The verifyAddress API is calls the YCDVerifyAddressWithAVSUE user exit to validate the address information provided by the user. For more information about the YCDVerifyAddressWithAVSUE user exit, refer to the *Selling and Fulfillment Foundation: Javadocs*. The verifyAddress API is called to validate the address information provided by the user. After modifying the Ship-to address information, the user can choose to save the updated address information in the address book. The modified address is saved as a new address in the address book.The manageCustomer API is called to save the updated Ship-to address in the address book.
  - Delete the Ship-to address: If a Ship-to address is specified for a line item, the user can delete that address by clicking the **Remove Address** hyperlink.

  The changeOrder API is called to save the updated line-level Ship-to address information.

- Provide the shipping preferences: Sterling Web enables users to indicate their shipping preferences such as the optimization type, primary shipping method, and primary requested delivery date. The getOrderFulfillmentDetails API is called to retrieve the shipping preferences for the order and order lines.

  The Sterling Web application provides the following shipping preferences:

  - Optimization Type: By default, the optimization type is selected from the preferences set at the organization level. If no preference is set at the organization level, the preferences set for SYSTEM scheduling rule is applied. For more information about SYSTEM scheduling rule, refer to the *Sterling Distributed Order Management: Configuration Guide*. The user can select the optimization type to be used to fulfill the order. The optimization type that the user selects will determine how the order is fulfilled. For example, the user can indicate that the products in the order should be shipped over as few shipments as possible in order to reduce shipping costs.
  - Primary shipping method: The user can provide the primary shipping method to be used for the order and order lines. If an item does not have inventory, it is not mandatory to provide the primary shipping method. However, if the item is available, the user must provide the primary shipping method. The listCarrierService API is called to retrieve the shipping methods available for the order. The getOrderFulfillmentDetails API is called to retrieve the shipping methods available for the order lines.
  - Primary Requested Delivery Date: The user can select the primary requested delivery date for the order.

    The user can update the changes made to the shipping preferences by clicking the **Update** button.

- Edit Line Shipping Options: The user can click this hyperlink to edit the shipping options for the line items. The user can choose to ship the entire order line. The user can also modify the requested delivery date and the shipping method.

If the shipping options and shipping methods defined at order line level are different than the ones defined at order header level, the shipping options and shipping methods defined at order line level will take precedence.

The changeOrder API is called when the user clicks the **Save** button, to save the updated shipping preferences for the order. For more information about shipping methods, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Provide Bill To Information

After providing the Sold To and Ship-to address information, it is mandatory for the user to provide the Bill To address information for the order. The Bill To page displays the current Bill To address and the addresses specified for the user. The getCompleteOrderDetails API is called to display the Bill To address information for the user. The getCustomerDetails API is called to display the addresses specified for the user.

The user can perform the following tasks as part of this step:

- Use another address as the Bill To address of the order: The user can click the **Use this address** hyperlink below the corresponding address to make that address the Bill To address for the order. When the user clicks this hyperlink, the new address is displayed in the **Bill-to address for this order** area. When the user clicks the **Continue** button, this Bill To address is saved. The ChangeOrder API is called to save this address.
- Modify the Bill To address information of the order: The user can click the **Edit address** hyperlink to modify the details of the default Bill To address. The getCompleteOrderDetails API is called to retrieve the Bill To address information. The getRegionList API is called to retrieve the information to be displayed in the **State/Province** drop-down list. The verifyAddress API is called to validate the address information provided by the user. After the address information is validated, the changeOrder API is called to save the updated Bill To address information. After modifying the Bill To address information, the user can choose to save the updated address information in the address book. The modified address is saved as a new address in the address book. When the user clicks the **Continue** button, this Bill To address is saved. The ChangeOrder API is called to save this address.
- Provide Purchase Order and Tax information: The user can provide additional information, such as the Purchase Order number, and indicate whether the order should be exempt from tax. The default tax exemption certification number and the flag associated with the tax is considered as mentioned in the user’s organization profile.

Provide Payment Information

Before placing an order, the user must specify the payment type to be used for the order. A user can use payments for an order using the following payment types:

- Credit Card
- Customer Account
- Stored Value Card

The getCompleteOrderDetails API is called to retrieve the payment methods available for the order. The getPaymentTypeList API is called to retrieve the payment types the storefront accepts. The YFSGetFundsAvailableUE user exit is used to retrieve information about the amount of funds available in the store value card that is used to pay for the order. The changeOrder API is called to save the changes made to the order. The various payment methods displayed in the user interface have priority assigned to them at the organization level in the Applications Manager. For more information about payment methods, refer to the *Sterling Distributed Order Management: Configuration Guide*.

Sterling Web provides different views of the page pertaining to payment entry based on the following conditions:

- Order has no payment method: If no payment method is saved on the order, users can view the order total, the amount to be charged to the payment methods, and the details of the payment methods such as the credit card type, display card number (in the case of a credit card), display account number (in the case of a customer account), and so on. In addition, the user can perform the following tasks:
    - Select a payment method: The user can select a payment method from the payment methods available to the user. The payment available for selection are defined at the buyer user level and buyer organization level. The getCustomerDetails API is called to retrieve the payment methods. The user can provide a new payment method for the order. The getPaymentTypeList API is called to retrieve the list of payment types to be displayed in the **Enter New Payment** drop-down list. Payment types pertaining to only CREDIT_CARD payment type group, are displayed in the **New Payment** drop down list. While providing a new payment method of the CREDIT_CARD payment type group, the user can modify the billing address for the credit card. The getCustomerDetails API is called to retrieve the billing address information displayed for the credit card. The user can add a new payment method of the CREDIT_CARD payment type group either by selecting a payment method from the existing payment methods or providing a new payment method. The getCustomerDetails API is called to retrieve the existing payment methods available for the user. If no payment method of the CREDIT_CARD payment type group has been saved on the order, the getcompleteOrder details API is called to add a new credit card for the order. The user can select a customer account from the various payment methods set for the user and the user's organization.
    - Use additional payment method: The user can add an additional payment method from the payment methods available to the user. The user can provide the CVV number for payment methods of the CREDIT_CARD payment type group. The user can add a new payment method of the CREDIT_CARD payment type group by selecting a payment method either from the existing payment methods or providing a new payment method. The getCustomerDetails API is called to retrieve the existing payment methods available for the user. If no payment method of the CREDIT_CARD payment type group is saved on the order, the getCompleteOrderDetails API is called to retrieve the information to be displayed when the user clicks the **Add Credit Card** button. The getCustomerDetails API is called to retrieve the billing address information to be displayed for the credit card. The user can remove any of the payment methods saved on the order by clicking the **Remove** button. The user can add a new payment method of the CUSTOMER_ACCOUNT payment type group from existing payment methods. The getCustomerDetails API is called to display the account numbers available to the user. The user can update the changes made to the payment method by clicking the **Update** button.
- Order has one payment method: If only one payment method is saved on the order, a user can view the order total, the amount to be charged to the payment method, details of the payment method, such as the credit card type, display card number (in the case of a credit card), display account number (in the case of a customer account), and so on. In addition, the user can perform the following tasks:
    - Modify CVV number: The user can provide the CVV number if the payment method for the order is of the CREDIT_CARD payment type group.

    - Modify credit card details: The user can modify the details of the credit card if the payment method saved on the order is of the CREDIT_CARD payment type group.
    - Remove payment method: The user can remove the payment method saved on the order.
- Order has multiple payment methods: If multiple payment methods are saved on the order, the user can view the order total, the amount to be charged to each of the payment methods, details of payment method, such as the credit card type, display card number (in the case of a credit card), display account number (in the case of a customer account), and so on. In addition, the user can perform the following tasks:
    - Provide CVV number: The user can provide the CVV number for payment methods of the CREDIT_CARD payment type group.
    - Modify credit card details: The user can modify the details of the credit card for payment methods of the CREDIT_CARD payment type group.
    - Remove payment method: The user can remove any of the payment methods saved on the order.
    - Change chargeable amount: The user can change the amount to be charged to each of the payment methods except the last payment method. The last payment method gets updated if a gift card or store value card is applied on an order. For example, if an order has three payment methods namely, CC1, CC2 and CC3 with charge amount of $300, $200 and $100 respectively. If a user adds a gift card of $150, the payment methods, CC1, CC2, and CC3 will be updated as $300, $200, and $0 respectively.
    - Add new credit card payment method: The user can add a new payment method of the CREDIT_CARD payment type group by selecting a payment method either from the existing payment methods or providing a new payment method. The getCustomerDetails API is called to retrieve the existing payment methods available for the user. If no payment method of the CREDIT_CARD payment type group is saved on the order, the getCustomerDetails API is called to retrieve the information to be displayed when the user clicks the **Add Credit Card** button. The getCustomerDetails API is called to retrieve the billing address information to be displayed for the credit card.
    - Add a new account: The user can add a new payment method of the CUSTOMER_ACCOUNT payment type group from existing payment methods. The getCustomerDetails API is called to display the account numbers available to the user.

The user can can provide a store value card number and add a store value card to the order. In addition, the user can remove an existing store value card from the order. For more information about payments, refer to the *Sterling Distributed Order Management: Configuration Guide.*

Confirm Order

The final step in the checkout process is confirming the order. After providing the address and payment information, the user can view the header-level and line-level information of the order to verify that the information provided is correct and place the order in the Order Summary page. The validateItemForOrdering API is called to verify the order details when an order is confirmed. After an order is confirmed, the getRuledetails API is called to get the value of SWC_VALIDATE_CART_ITEMS_ON_CHECKOUT rule to validate the products in the cart before checkout. For more information about setting this rule, refer to the *Channel Applications Manager: Configuration Guide*. The changeOrder API is called to save the Federal Trade Commission (FTC) rules on a confirmed order. For more information about FTC, refer to the *Channel Applications Manager:*

Configuration Guide. The getCompleteOrderDetails API is called to retrieve the header-level and line-level information to be displayed in the Order Summary page. In the single-step checkout process, the getOrderFulfillmentDetails API is called to retrieve the estimated delivery dates, and the changeOrder API is called to save the estimated delivery dates for the order. In the multi-step checkout process, the getCompleteOrderDetails API is called to retrieve the information to be displayed in the Order Summary page. The processOrderPayments API is called for payment processing when an order is confirmed. Whenever, an order is confirmed, an e-mail is sent to the buyer user comprising the details of the order. To confirm an order appropriate payment processing rules should be set. For more information about payment processing rules, refer to the *Sterling Distributed Order Management: Configuration Guide.*

The user can perform the following tasks as part of this step:

- Modify Sold To address: The user can modify the Sold To address information by clicking the **Edit** hyperlink in the Sold To address area. For more information, refer to the topic, "Provide Sold To Information".
- Modify Ship-to address: The user can modify the Ship-to address information by clicking the **Edit** hyperlink in the Ship-to address area. For more information, refer to the topic, "Provide Ship-to Information".
- Modify Bill To address: The user can modify the Bill To address information by clicking the **Edit** hyperlink in the Bill To address area. For more information, refer to the topic, "Provide Bill To Information".
- Modify the shipping preferences: The user can modify the shipping preferences by clicking the **Edit** hyperlink in the Shipping Option area. For more information, refer to the topic, "Provide Ship-to Information".
- Navigate to the Product Detail page of line items: The user can navigate to the Product Detail page by clicking the **Item ID** hyperlink. For more information, refer to the topic, "Viewing Product Details".
- Modify the line-level Ship-to address information: The user can modify the Ship-to address information for a line item by clicking the **Edit Shipping** hyperlink. For more information, refer to the topic, "Provide Ship-to Information".
- Navigate to the Adjustment Details dialog box: The user can navigate to the Adjustment Details dialog box by clicking the **Adjustments** value hyperlink to view the details of adjustments for a line item. The getCompleteOrderDetails API is called to retrieve the adjustment details for the line item.
- Navigate to the Shipping Adjustment Details dialog box: The user can navigate to the Shipping Adjustment Details dialog box by clicking the **Shipping Costs** hyperlink to view the details of the shipping discounts for the order. The getCompleteOrderDetails API is called to retrieve the details of the shipping discounts for the order.
- Navigate to the Adjustment Details dialog box: The user can navigate to the Adjustment Details dialog box to view the details of all the header-level adjustments for the order by clicking the **Order Total Adjustments** hyperlink. The getCompleteOrderDetails API is called to retrieve the details of the order level adjustments for the order. The getChargeNameList API is called to display the charge details based on the value of the OrganizationCode attribute passed in this API. For more information about getChargeNameList API, refer to the *Selling and Fulfillment Foundation: Javadocs*.
- Navigate to the View All Adjustments dialog box: The user can navigate to the View All Adjustments dialog box to view the details of the order-level and line-level adjustments for the order by clicking the **Total Price** hyperlink. The getCompleteOrderDetails API is called to retrieve the details of the order-level and line-level adjustments for the order. The getChargeNameList API is called to display the charge details based on the value of the OrganizationCode attribute passed in this API. For more information about getChargeNameList API, refer to the *Selling and Fulfillment Foundation: Javadocs*.

- Add a coupon: The user can provide a coupon code and add a coupon to the order only if the **Enable Coupon Entry** rule has been set in the Channels Application Manager. For more information about this rule, refer to the *Channel Applications Manager: Configuration Guide*. The validateCoupon API is called to validate the coupon.
- Modify payment information: The user can modify the payment information for the order by clicking the **Edit** hyperlink in the Payment Summary area. For more information, refer to the topic, "Provide Payment Information".
- Return to cart detail page: The user can return to the cart detail page by clicking the **Return to cart detail page** button. For more information, refer to the topic, Viewing Cart Details.
- Submitting order: The user can confirm and submit the order after reviewing it by clicking the **Submit Order** button. The confirmDrfaftOrder API is called submit the order.

If the user makes any changes to the order in the Order Summary page, the changeOrder API is called to save the updated information.

Note: Sterling Web assumes that all the order status modification rules, except the Change Currency modification rule, are set to allow modifications. If the order status modification rules do not allow modifications to be made to a cart, the user will not be able to check out a cart and confirm the order.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can checkout a cart in either a single step or multiple steps based on the user preference that has been configured. For more information about configuring checkout type preferences, refer to the "Managing Buyer User Details" topic. If the user has selected the single step checkout process, clicking the **Next** button on the Select stores to pickup items page takes the user to the Order Summary page. If the user has selected the multi step checkout process, clicking the **Next** button on the Select stores to pickup items page takes the user to the Sold To page.
- The **Show more addresses** hyperlink is displayed only if more than four addresses exist in the address book.
- The getRegionList API is called to get the value of YCD_STATE_DROP_DOWN_REGION_SCHEMA rule. If this API does not return any value, the **State/Province** text box is displayed as a text box and not a drop-down list.
- The **New Address** hyperlink is displayed in the Line Level Shipping Information page only if the line-level Ship-to address is not defined for the line item.
- The **Edit Address** hyperlink and **Remove Address** hyperlink are displayed in the Line Level Shipping Information page only if the line-level Ship-to address has been defined for the line item.
- The **Product Description** column can be expanded only if the item has minor line items associated with it, and they are not displayed.
- The **Product Description** column can be collapsed only if the item has minor line items associated with it, and they are displayed.
- The store value card information pertaining to an order will be displayed only if at least one payment method of the STORED_VALUE_CARD payment type group exists for the order.
- A user can enter a store value card number and add the store value card to an order only if the storefront is configured to accept at least one payment method of the STORED_VALUE_CARD payment type group.

- In the Order Summary page, the **Edit Shipping** hyperlink is displayed only for order lines that are to be shipped.
- In the Order Summary page, the **Adjustments** hyperlink is displayed for a line item only if one or more discounts are applied to the line item.
- In the Order Summary page, the **Order Total Adjustments** is displayed as a hyperlink if one or more discounts are applied to the order. The
- In the Order Summary page, the **Shipping Costs** is displayed as hyperlink if one or more shipping discounts are applied to the order.
- In the Order Summary page, the **Total Price** is displayed as a hyperlink if discounts of any type are applied to the order.
- A user can make modifications to a cart and check out a cart only if the appropriate order status modification rules are set up.
- When a new address is added, or an existing address is modified, the address is validated by calling the verifyAddress API, which in turn calls the YCDVerifyAddressWithAVSUE user exit. For more information about the YCDVerifyAddressWithAVSUE user exit, refer to the *Selling and Fulfillment Foundation: Javadocs*
- The YFSGetFundsAvailableUE user exit is used to retrieve information about the amount of funds available in the store value card that is used to pay for the order. For more information about the YFSGetFundsAvailableUE user exit, refer to the *Selling and Fulfillment Foundation: Javadocs.*
- The order of Payment Methods displayed in the user interface depends upon the charge sequence configured in the Applications Manager. For more information about setting up the charge sequence, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- A user can add coupons to an order only if **Enable Coupon Entry** rule is configured in the Channels Application Manager. For more information about setting this rule, refer to the *Channel Applications Manager: Configuration Guide*.
- When a user places an order using credit card, the YFSCollectionCreditCardUE user exit is used to authorize the payments. For more information about the YFSCollectionCreditCardUE user exit, *Selling and Fulfillment Foundation: Javadocs.*
- When a user places and order using a customer account and a third party authorizes the payments, the YFSCollectionCustomerAccountUE user exit is used to authorize the payments. For more information about the YFSCollectionCustomerAccountUE user exit, *Selling and Fulfillment Foundation: Javadocs.*
- A guest user must log in to the Sterling Web application to be able to check out.
- A buyer user receives an e-mail whenever the user confirms an order.
- A user can perform checkout functions only if the user has the necessary permissions.

Implementation

This section explains the configurations for this functionality:

- To enable a user to pay for an order using a store value card, you must configure a store value card payment type of the STORED_VALUE_CARD payment type group for the storefront. For more information about creating a payment type, refer to the *Sterling Distributed Order Management: Configuration Guide*.

- To enable a user to add coupons to the order, you must enable the **Enable Coupon Entry** rule. For more information about this rule, refer to the *Channel Applications Manager: Configuration Guide*.
- You must configure the default service carrier for new draft orders by configuring the **Default Carrier Service For New Draft Orders** rule in the Channel Application Manager. For more information about setting this rule, refer to the *Channel Applications Manager: Configuration Guide*.
- To display the availability information, you must enable the **Check Inventory On Cart** business rule. For more information about configuring this rule, refer to the *Channel Applications Manager: Configuration Guide*.
- To validate the products in the cart before checking out, you must enable the SWC_VALIDATE_CART_ITEMS_ON_CHECKOUT rule in the Channel Application Manager. For more information about setting this rule, refer to the *Channel Applications Manager: Configuration Guide*.
- To enable synchronization between dates, enable the **Synchronize Dates Between Master Order Dates And Dates On Order Line And Schedules** business rule in the Applications Manager. This rule synchronizes the requested dates and the expected ship dates for the order, order header, order line, and order line schedules. The requested dates are synchronized with the requested ship, requested delivery, and cancel dates on the order line or header. The expected dates are synchronized with the order schedules.
- You must implement the YCDVerifyAddressWithAVSUE user exit to verify the addresses and the YFSGetFundsAvailableUE user exit to determine whether funds are available on a payment method for an order. This user exit is also implemented to verify the addresses with the Address Verification System (AVS). For more information about these user exits, refer to the *Selling and Fulfillment Foundation: Javadocs*.
- Sterling Web assumes that all the order status modification rules, except the Change Currency modification rule, are set to allow modifications. If the order status modification rules do not allow modifications to be made to a cart, the user will not be able to check out a cart and confirm the order. For more information about configuring order modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- You must configure appropriate payment processing rules to be able to confirm an order. For more information about payment processing rules, refer to the *Sterling Distributed Order Management: Configuration Guide.*
- You must assign the necessary permissions to the corresponding user groups to enable them to check out an order. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Note: Apart from modifying currency of an item, modifying the cart, and configuring the order statuses, order modification rules do not govern any action displayed on the jsp for the orders in the Draft status.

Reference Implementation

None.

Searching for an Order

A user may want to search for a particular order by providing appropriate search criteria. For example, a user may want to search for all the orders that contain a particular item.

Solution

Sterling Web™ enables a logged in user to search for an order from the Order List page. A user can search for an order using one of the following options:

- Quick Search: A user can provide the Sales Order Number, Order Name, or Product ID and search for an order. However, the Order Name will not be displayed on the Order List page and the Order Detail page. If a physical kit item with multiple line items is added to an order and the user conducts a search with the product ID of the line item, the order comprising this line item is not displayed. For example, let us suppose that Order 1 contains ProductA and ProductB as line items, and Order2 contains a physical kit item comprising product A and product C as minor line items. When a user searches for an order by providing Product ID as ProductA in the search criteria, only Order1 will be retrieved in the search results.
- Advanced Search: Show All: A user can use this option to view all the available orders. If any existing search settings available, those settings will not be considered when displaying the results. A user can select Open, Shipped or, Backordered statuses from the **Status** drop-down list. When a user selects Open status from the **Status** drop-down list, all the orders are displayed. When a user selects, Backordered from the **Status** drop-down list, orders in the Backordered status are displayed. When a user selects Shipped from the **Status** drop-down list, orders that are in Partially Shipped, Beyond Shipped, Shipped or delivered statuses are displayed. A user can search for orders pending approval by selecting Approval from the **Hold** drop-down list.

A user can use wild card characters to search for Order No, Order Name, or Product ID. For example, if a user is searching for an order with 12345 as Order No, the user can perform a quick search by providing 123* as a search term.

The getOrderList API is called to retrieve the list of orders matching the search criteria provided by the user. The orders are displayed, sorted by the Last Modified value in descending order.

This list can be sorted in either an ascending order or a descending order based on the following options:

- **Sales Order No**
- **Purchase Order No**
- **Last Modified**
- **Date Submitted**
- **Payment Status**

When a user performs a search for an order using Order Status as the search criteria, the value of the `ModifytsQryType` attribute in the input XML file of the getOrderList API is set to DATERANGE.

When a wild card character is used to search for an order, the value of the `QryType` attribute in the input XML file of the getOrderList API is set to FLIKE. If a wild card character is not used, the value of the `QryType` attribute is set to EQ. For more information about defining complex queries, refer to the *Selling and Fulfillment Foundation Customizing APIs Guide.*

Pagination

If the number of orders matching the search criteria is more than five, the Order List page displays the list of orders across multiple pages. A user can navigate to any of the pages by clicking the corresponding page number hyperlink at the bottom of the page. The getPage API calls the getOrderList API to retrieve the corresponding page. For more information about the getPage API, refer to the *Selling and Fulfillment Foundation: Javadocs*.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can search for particular orders only if the user has the necessary permissions.
- A user cannot search for either purged or archived orders.
- A guest user cannot search for orders.
- A user can reorder an order with the `SupportLevel` attribute set as medium, only if the appropriate user permissions are configured for the user.

Implementation

This section explains the configurations for this functionality:

- You must assign permissions to the corresponding user groups to enable the corresponding users to search for orders. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must set the appropriate resource permissions for the users to enable them reorder the orders with the `SupportLevel` attribute set as medium.

Reference Implementation

None.

Viewing Order Details

After an order is placed, a user may want to view the details of the order. For example, a user may want to view the notes added to the order if the user wants to track the changes, if any, made to the order.

Solution

Sterling Web™ enables a user to view the details pertaining to an order. A user can view header-level information such as the Sold To, Ship-to, and Bill To addresses, and the shipping option for the order. A user can also view line-level information such as the ID of the line item, short description of the line item, the order line status, alternate Ship-to address, requested quantity of the line item, price adjustments, order line total, and so on. A user can also view information about the coupons or promotions, if any, applied to the order and the mode of payment for an order. The getCompleteOrderDetails API is called to retrieve the details of the order.

Additionally, a user can perform the following actions from the Order Detail page based on user permissions and order status modification rules:

- Navigate to the Item Detail page of a line item: A user can navigate to the Item Detail page by clicking the item description hyperlink. For more information about viewing the details of a line item, refer to the topic, "Viewing Product Details".
- Add or view notes: A user can add a note about an order or view existing notes, if any, pertaining to the order. For more information about viewing or adding notes to an order, refer to the topic "Adding or Viewing a Note".
- Print the Order Detail page: A user can print the Order Detail page. The getCompleteOrderDetails API is called to retrieve the read-only details of the order.
- E-mail order details: A user can e-mail the details pertaining to an order. For more information about e-mailing order details, refer to the topic "E-Mailing Cart and Order Details".
- Reuse an Order: A user can create a new cart from an existing order. For more information about creating a new cart from an existing order, refer to the topic "Copy a Cart or an Order".
- Edit an order: A user can make changes to an order based on the order status modification rules. For more information about changing an order, refer to the topic "Changing an Order".
- Cancel an order: A user can cancel an order based on the order status modification rules. For more information about cancelling an order, refer to the topic "Cancelling an Order".
- Navigate to the Order Total Adjustments dialog box: The user can navigate to the Order Total Adjustments dialog box by clicking the Order Total Adjustments value hyperlink to view the details of adjustments made on an order.
- View shipping cost: The user can navigate to the Shipping Costs dialog box to view the shipping cost applied on the order.
- View the tracking number of an order line: A user can view the tracking number of an order line. By clicking the Tracking # value hyperlink, a user can view the shipment status of the order line, which is displayed in the Web site of the carrier shipping the order line. The invokeUE API is called to determine, from the system cache, if the URL of the external Web site is supported. If the URL is supported, the invokeUE API is called again to retrieve the URL.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can view the details of an order only if the user has the necessary permissions.
- A user can view the holds applied on an order by clicking the appropriate hyperlink on the Order Detail page.
- The **Edit Order** button is displayed in the Order Detail page if all of the following conditions are met:
  - The user has the appropriate permissions to modify the order. By default, the permission is given to a user belonging to the BUYER-USER group.
  - The value of the `SupportLevel` attribute is Full.
  - The `ContainsPendingChanges` attribute is set to Y in the getOrderDetails API, and the user is authorized to make changes to the order.
  - The order is not in Cancel status.
  - The order status modification rules allow modification of the order.

  For more information about setting the Support Level attribute, refer to the S*elling and Fulfillment Foundation: Application Platform Configuration Guide*.
- The **Refresh** button is displayed in the Order Detail page if all of the following conditions are met:
  - The user has the appropriate permissions to change the order. By default, the permission is given to a user belonging to the BUYER-USER group.
  - The `ContainsPendingChanges` attribute in the getOrderDetails API is set to Y, and the user is authorized to make changes to the order.
- The **Approve** and **Reject** buttons are displayed only if the user has the permission to approve or reject the order.
- If an order has already been approved, a user can view the name of the person who has approved the order, as also the name of the person previously approved the order, and who is going to be the next approver, by clicking the **Approval History** hyperlink. This link is displayed if an approval hold exists on an order irrespective of the order status.
- The **Order Total Adjustments** hyperlink is displayed only if more than one discount is applied to an order.
- The **Shipping Costs** hyperlink is displayed only if shipping cost is applied on the order.
- The **Order Again** button is displayed only if all of the following conditions are met:
  - The order can be ordered again and if the value of `SupportLevel` attribute is Full.
  - The user has the appropriate permissions to change the order. By default, the permission is given to a user belonging to the BUYER-USER group.
- A user can reorder an order with the `SupportLevel` attribute set as medium, only if the appropriate user permissions are configured for the user. For more information about setting the Support Level attribute, refer to the S*elling and Fulfillment Foundation: Application Platform Configuration Guide*.
- The **Tracking #** value will be displayed as a hyperlink only if the shipment tracking URL is available.

Implementation

This section explains the configurations for this functionality:

- You must assign the necessary permissions to the corresponding user groups to enable users to view the details of an order. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must set the appropriate resource permissions for the users to enable them reorder the orders with the `SupportLevel` attribute set as medium. For more information about setting the Support Level attribute, refer to the S*elling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

Changing an Order

Sometimes a user may want to modify an existing order. For example, a user may want to add more order lines to an order.

Solution

The change order functionality enables a user to change an order and details of an order, based on the modification rules configured in the system. The change order functionality enables a user to perform the following functions:

- Add a new order line to an order
- Delete the order lines
- Reconfigure order lines
- Increase or decrease the quantity of order lines
- In-store pickup selection
- Change the Ship to address at the order header level and the order line level
- Change the Requested Delivery Date
- Change the shipping options
- Change shipping preferences
- Change the Bill To address
- Change the Sold To address
- Change the payment methods
- Change the Ship complete option
- Change Delivery method
- Change shipping methods
- Change the order name
- Change the order fulfillment method
- Add coupons
- Change tax exemption
- Change the PO Number
- Change the line shipping options

The details of an order returned by the getCompleteOrderDetails API contain the modification rules list. The modification rules list comprises information about what changes can be performed on an order. Changes made using the Sterling Web™ user interface are sent to an order service by calling the changeOrder API. The changes made to an order are saved in the database based on the time period configured in **Pending Order Changes Will Expire in** transaction rule in the Applications Manager. For more information about configuring this rule, refer to the *Sterling Distributed Order Management: Configuration Guide.* There is a possibility that submission of the request for modification of an order to the order service may fail because of other modification rules configured in the system, that are not set as a part of Sterling Web. For example, if a user wants to add a new order line, the corresponding modification rule enables a user to do so. However, adding a new order line changes the price of the order and the tax applied on the order. An error is thrown if the user does not have permission to modify the Price and Tax modification rules.

For information about the standard modification rules and creating custom modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.

The Order modification rules are specified at two levels.

- Order Header Level: The modification rules specified for the order header level affect modification of order header level information. Some of the order header level changes include changes to Ship to address, and Change Payment Type, and adding multiple payment types.
- Order Line Level: The order line change is similar to the order header change. The order lines changes include cancelling an order line, changing the line item quantity, changing the line item configuration, and changing the line item's Ship to address.

The order service enables modifications to specific attributes of an order, for example, adding an order line to an order. Modification level indicates the level at which a particular modification is carried out. The modification levels include header level and line level. Modifications are applied to a particular level and a particular processing status. For example, if modifications are requested for an order either at the header level or at the line level, the corresponding order lines are picked up for validation irrespective of the permissions set for allowing modifications to the order statuses. All modification rules operate at system-defined ranges. For example, the Change Bill To modification to the order is always defined between the statuses 1000.00 and 3350.00. The system never allows a Change Bill To modification to occur at a status of 3700.00, whereas modifications between the statuses 1000.00 and 3350.00 are allowed.

The output of the getCompleteOrderDetails API includes an additional `SupportLevel` attribute that indicates the functions that can be performed on an order in the user interface. The `SupportLevel` attribute can take the following values:

- Minimum: If the `SupportLevel` attribute is set to Minimum, the order cannot be viewed in the Sterling Web user interface. An error message s displayed in the user interface to inform the user that the order cannot be viewed in the user interface.
- Medium: If the `SupportLevel` attribute is set to Medium, the order can be viewed, but only specific tasks can be performed in the Sterling Web user interface. A user can cancel, approve, or reject the order.

  Note: The user can cancel the order by configuring the appropriate modification rule in the Applications Manager at the order header level.
- Full: If the `SupportLevel` attribute is set to Full, a user can perform operations on the order based on the appropriate modification rules set in the Applications Manager.

For more information about modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.

Sterling Web also enables a user to make changes to an order created using other applications. For example, a user can modify an order that was created using Sterling Call Center™. However, orders that have a delivery service, price lock, or provided service associated with them cannot be modified through Sterling Web. A user can perform specific functions on an order based on the tagging mechanism. The getCompleteOrderDetails API returns a `SupportLevel` attribute, which decides the support level of the order based on the order tags. For more information about defining order tags, refer to the *Sterling Distributed Order Management: Configuration Guide*.

The changes pending on an order expire after a specific time period configured in the **Pending Order Changes Will Expire in** transaction-specific rule in the Applications Manager. For more information about configuring this rule, refer to the *Sterling Distributed Order Management: Configuration Guide*.

Note: If a user places an order through Sterling Call Center™ without specifying the shipping method, but the user wants to modify through Sterling Web, the user will have to specify the shipping method in Sterling Web.

End-User Impact

This section explains the end-user impact of this functionality:

- The **Edit Order** button will be displayed in the Order Detail page if all of the following conditions are met:
  - If the `SupportLevel` attribute returned by the getCompleteOrderDetails API is set to Full.
  - The value of the `ContainsPendingChanges` attribute is set to Y in the getOrderDetails API.
  - The order is not in Cancel state.
  - A user can change an order only if the user has the necessary permissions.
- The **Refresh** button will be displayed on the Order Detail page if all of the following conditions are met:
  - The value of the `ContainsPendingChanges` attribute is set to Y in the getOrderDetails API and the logged in user is the owner of the orders that were modified.
  - A user has the necessary permissions.
- A user can click the **Edit cart details** hyperlink and modify the name of the cart only if the appropriate order modification rule is configured.
- The **Notes** hyperlink is not displayed when the user is modifying the order.
- A user can cancel, approve, or reject an order only if the value of the `SupportLevel` attribute returned by the getCompleteOrderDetails API is set to Medium.
- When a user modifies an order, the pricing rules that were applicable to the original order may not be applicable to the modified order. Other pricing rules that exist in the system may be applicable to the modified order.
- When a user modifies an order, if the total price of the order changes, the increase in the order amount will be charged using the last payment method in the order. If the user has modified the payment method, the order amount will be charged using the new payment method.

Implementation

This section explains the configurations for this functionality:

- You must assign the necessary permissions to the corresponding user groups to enable them to change an order. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must configure the appropriate Order Modification Rules to enable users to perform modifications to an order based on the status of the order and modify the name of the order. For more information

about order modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.

- You must configure the **Pending Order Changes Will Expire in** transaction specific rule to set up expiry time period on the changes made to an order in the Applications Manager. For more information about pending order changes, refer to the *Sterling Distributed Order Management: Configuration Guide*.

Reference Implementation

None.

Approving Orders

Sometimes, orders may have to be placed on hold if the order's total amount exceeds the user's spending limit. In such a scenario, another user must be able to approve the order for the order to be processed.

Solution

Sterling Web™ provides the capability to place an order on hold when the order total exceeds the user's spending limit. When an order is on hold, the order cannot be scheduled or released. The hold type that is applied to the order is determined by the **Select hold type that needs to be applied when order needs approval** rule that is configured using the Applications Manager. For more information about defining transaction rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.

The default values associated with the e-mail template of an order are based on the template defined for the orders in the Applications Manager. For more information about defining templates, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

By default, Sterling Web provides the following e-mail templates:

- `YCD_OrderApprovalEmail.xsl.sample`
- `YCD_OrderApprovalRejectedEmail.xsl.sample`
- `YCD_OrderApprovalAcceptedEmail.xsl.sample`

This section describes the following tasks:

- Viewing the List of Orders Pending Approval
- Viewing Order Approval History
- Approving Orders That are on Hold

Viewing the List of Orders Pending Approval

Approvers or proxy approvers can view the orders that have been placed on hold, in their Account Activity page. Approvers can view the orders owned by them, and the orders for which they are proxy approvers. Approvers cannot view the orders that are pending approval by another approver.

After an order is placed on hold, the order is displayed under the **Order Approvals** panel in the Account Activity page of the approver and proxy approver of the user to whom the order belongs. If more than five orders are pending approval, the approver can click the **More** hyperlink under the **Order Approvals** panel and navigate to the Order Approvals List page.

If the user accessing the Account Activity page does not belong to the BUYER_APPROVER user group, the **Order Approvals** panel is not displayed in the Account Activity page. If the user belongs to the BUYER_APPROVER user group, the getCustomerContactList API is called to retrieve the list of users for whom the current user is the approver or proxy approver.

For example, let us suppose that User1 has User2 as approver and User3 as proxy approver, and User4 has User5 as approver and User2 as proxy approver. When User2 accesses the Account Activity page, the getCustomerContactList API is called. This API returns the values of the `CustomerContactID` attribute as User1 and User4, the values of the `ApproverUserID` attribute as User2 and User5, and the values of the `ProxyApproverUserID` attributes as User3 and User2. The values of the `CustomerContactID`, `ApproverUserID`, and `ProxyApproverUserID` attributes returned by the getCustomerContactList API are passed as input to the complex query for the `CustomerContactID` and `ResolverUserID` attributes in the input XML of the getOrderList API.

The getPage API is called, which in turn calls the getOrderList API to retrieve the list of orders that are pending approval by the approver or proxy approver, which is User2 in this example. For more information about defining complex queries, refer to the *Selling and Fulfillment Foundation Customizing APIs Guide*.

Pagination

If the number of orders that are pending approval is more than ten, the Order Approvals List page displays the orders list across multiple pages. A user can navigate to any of the pages other than the current page by clicking the page number hyperlink at the bottom of the corresponding page. The getPage API is called to retrieve the page of records to be displayed. For more information about the getPage API, refer to the *Selling and Fulfillment Foundation: Javadocs*.

Viewing Order Approval History

A user can view who the name of the person who has approved an order previously, and the name of approver who is scheduled to approve the order, by clicking the **Approval History** hyperlink in the Order Detail page. If the order has been rejected by an approver, the user can view the name of the person who has rejected the order. Additionally, the user can view the notes, if any, entered against the order when the order was either approved or rejected.

Approving Orders That are on Hold

Sterling Web enables an approver to approve or reject an order that is pending approval, from the Order Detail page. When an approver navigates to the Order Detail page of the order that is pending approval, the getCompleteOrderDetails API is called to retrieve the details of the order to be displayed. The Order Detail page also displays an **Approve** and a **Reject** button that enable the approver to either approve or reject the order. If the approver approves the order, the system determines if the order total exceeds the approver's spending limit. If the order total is less than the approver’s spending limit, the hold applied to the order is resolved automatically. However, if the order total exceeds the approver’s spending limit, an e-mail notification is sent to the approver of the approver, stating that the order is still pending approval. When the person who is authorized to give final approval to the order approves it, the hold on the order is automatically resolved, and an e-mail notification is sent to the user to whom the order belongs and the approver of the order, stating that the order has been approved.

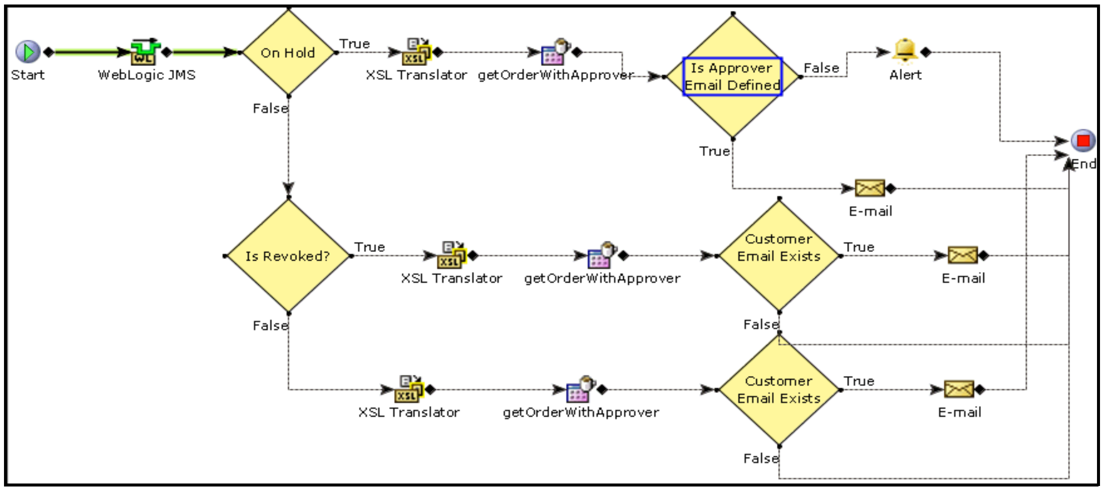
If the approver rejects the order, an e-mail is sent to the user to whom the order belongs, indicating the reasons for the rejection. If an order is rejected, the order will no longer be displayed in the list of orders pending approval for both the approver and the proxy approver.

Sterling Web provides the CHANGE_ORDER.ON_HOLD_TYPE_STATUS_CHANGE event as part of the factory setup that generates e-mails to be sent to the approver of the order and the user to whom the order belongs. A user must associate the Order Approval Email action with the CHANGE_ORDER.ON_HOLD_TYPE_STATUS_CHANGE event. The CHANGE_ORDER.ON_HOLD_TYPE_STATUS_CHANGE event is raised when a hold is applied to an order through the CHANGE_ORDER transaction. Whenever the Order Approval Email action is invoked, the CHANGE_ORDER.ON_HOLD_TYPE_STATUS_CHANGE event invokes the

YCD_Order_Approval_Email_8.5 service that generates an e-mail, which is sent to the approver of the order and the user who placed the order, to indicate that the order is on hold, pending approval.

For more information about configuring a service, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide.*

A sample implementation of the YCD_Order_Approval_Email_8.5 service, used to generate an e-mail when an order is on hold, pending approval, is illustrated in the following figure.



In this illustration, the custom API verifies whether a proxy approver exists for the approver of an order. If a proxy approver exists for the approver of the order, the getCustomerContactList API is called to fetch the details of the user who is set as the approver of the order. The YCD_Order_Approval_Email_8.5 service passes through a condition that checks whether the approver of the order has an e-mail address. If the order was created using either the Sterling Web application or the Sterling Call Center and Sterling Store™ application, it is assumed that the approver has an e-mail address.

If the approver of the order has an e-mail address, an e-mail is sent to both the approver of the order and the user to whom the order belongs.

If the approver of the order has an e-mail address, e-mails will be sent in the following situations:

- After a user places an order, if the order is put on hold, pending approval, an e-mail will be sent to the approver of the order or the proxy approver and the user to whom the order belongs stating that the order has been put on hold, pending approval.
- When the approver approves or rejects the order, an e-mail will be sent to the approver of the order and the user to whom the order belongs stating that the order has been either approved or rejected.
- If, after the approver of the order approves the order, the order is put on hold, pending approval from a second approver, an e-mail will be sent to the new approver and the user to whom the order belongs, stating that the order is on hold, pending approval.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can approve or reject an order that is pending approval only if the user has the necessary permissions.
- A user can view the **Approve** and a **Reject** button on the Order Detail page, only if the user has the necessary permissions. By default, the necessary permissions are assigned to a user belonging to the BUYER_APPROVER user group.
- A user can view the Order Approvals panel on the Account activity page only if the user has necessary permissions. By default, the necessary permissions are assigned to a user belonging to the BUYER_APPROVER user group.

Implementation

This section explains the configurations for this functionality:

- To view the Order Approvals panel on the Account activity page, the user must have the necessary permissions. By default, the necessary permissions are assigned to a user belonging to the BUYER_APPROVER user group.
- To use this functionality, you must configure the **Select hold type that needs to be applied when order needs approval** rule in the Applications Manager. For more information about defining transaction rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- You must configure the **Select hold type that needs to be applied when order needs approval** rule in the Applications Manager appropriately, so that the user belonging to the BUYER_APPROVER user group can automatically resolve the hold type. For more information about defining hold types, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- You must assign the necessary permissions to the corresponding user groups to enable them to approve or reject orders pending approval. By default, permissions for this feature are assigned to the BUYER_APPROVER user group. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

Searching for Orders Pending Approval

A user may want to search for an order that is on hold, pending approval, and view the details of that order.

Solution

Sterling Web™ enables a user to search for orders that are on hold, pending approval, from the Order Approvals List page. A buyer user can search for orders on hold only on the Advanced Search page by selecting Approval from the **Hold** drop-down list. Approvers or proxy approvers can view the orders that have been placed on hold, in their Account Activity page. Approvers can view the orders owned by them, and the orders for which they are proxy approvers. Approvers cannot view the orders that are pending approval by another approver.

A buyer approver can search for an order that is on hold using the following options:

- Quick Search: A user can provide the order number, name of the order owner or the order approver, and search for the order.
- Show All: A user can use this option to view all the orders that are on hold, and for which the current user is the approver or proxy approver. If any existing search settings are available, those settings will not be considered to display the results.

A user can use wild card characters to search for an order that is on hold. However, Sterling Web supports only the asterisk (*). For example, if a user is looking for an order with order number 12345, the user can perform a quick search for the order by providing 123* as a search term.

The get Page API is called, which in turn calls the getOrderList API to retrieve the list of orders matching the search criteria provided by the user. The orders are displayed, sorted by the Date Received value in descending order. The user can sort the list in either ascending or descending order based on the following options:

- **Order No.**
- **Date Received**
- **Order Owner**

When a wild card character is used to search for an order, the value of the `QryType` attribute in the input XML file of the getOrderList API is set to LIKE or FLIKE. If a wild card character is not used, the value of the `QryType` attribute is set to EQ. For more information about defining complex queries, refer to the *Selling and Fulfillment Foundation Customizing APIs Guide.*

Pagination

If the number of orders matching the search criteria is more than ten, the Order Approvals List page displays the orders list across multiple pages. The user can navigate to any of the pages other than the current page by clicking the page number hyperlink at the bottom of the page. The getPage API is called to retrieve the corresponding page to be displayed. For more information about the getPage API, refer to the *Selling and Fulfillment Foundation: Javadocs*.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can view the Order Approvals panel on the Account activity page only if the user has necessary permissions. By default, the necessary permissions are assigned to a user belonging to the BUYER_APPROVER user group.
- A user can search for orders that are on hold only if the user has the necessary permissions. By default, the necessary permissions are assigned to a user belonging to the BUYER_APPROVER user group.
- If an approver is a proxy of another approver, both the approver and the proxy approver can view the orders, which are pending the approver's approval, and either of them can approve or reject those orders. For example, if A is a proxy of B, both A and B can view the orders that are pending approval from B, and either A or B can approve or reject those orders.
- A buyer user can search for orders on hold only on the Advanced Search page by selecting Approval from the **Hold** drop-down list.

Implementation

You must assign permissions to the corresponding user groups to enable them to search for orders that are on hold. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation Configuration Guide*. By default, the necessary permissions are assigned to a user belonging to the BUYER_APPROVER user group.

Reference Implementation

None.

Adding or Viewing a Note

A user may sometimes want to view the notes added to a cart or order, or add new notes for future reference. For example, a user may add notes to an order that went through several modifications.

Solution

Sterling Web™ provides a **Notes** hyperlink in the Cart Detail page and the Order Detail page that enables a user to either view the existing notes in a cart or order, and add new notes. A user can view the existing notes to track the changes that have occurred in a cart or order, or add new notes describing the products that have been added to the cart or order. The getCompleteOrderDetails API is called to retrieve the existing notes, and the changeOrder API is called to add a new note to a cart or order. A note entered from Sterling Web can be viewed by users if they have the necessary permissions. A Sterling Web user cannot view an internal note entered using the Sterling Call Center and Sterling Store application.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can view or add notes only if the user has the necessary permissions. Guest users cannot view or add notes to a cart.
- A user cannot delete a note after adding it.
- Users cannot view internal notes entered using the Sterling Call Center and Sterling Store application.

Implementation

This section explains the configurations for this functionality:

- To enable a user to add a new note to a cart or order, ensure that the corresponding order modification rule is enabled in the Channels Applications Manager.
- If you are not using the note reasons provided as part of the reference implementation, ensure that the value of the NOTES_REASON common code is hard-coded as Web_Added_Visible_Note to enable a user to add or view notes.
- You must assign the necessary permissions to the corresponding user groups to enable them to add or view notes. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide.*

Reference Implementation

None.

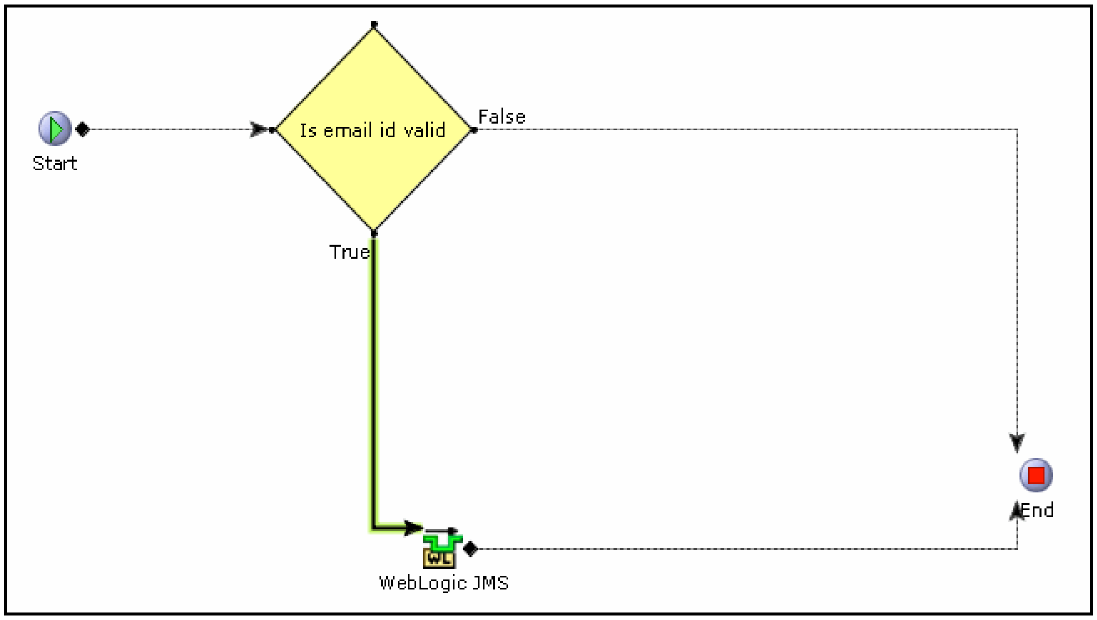
E-Mailing Cart and Order Details

A user may sometimes have to send the details pertaining to a cart or order through e-mail.

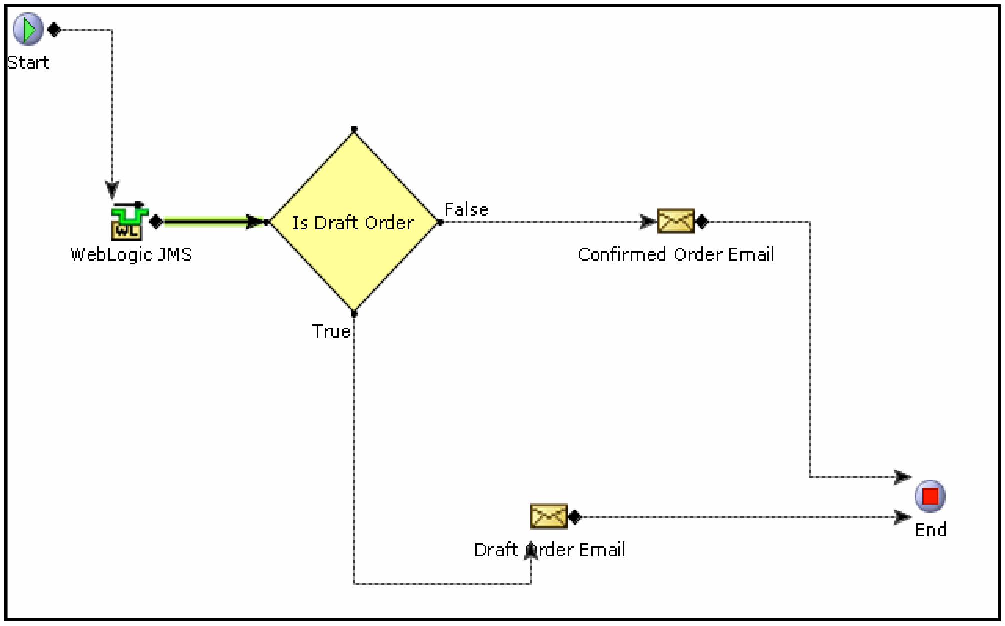
Solution

Sterling Web™ enables a user to send information pertaining to a cart or order through e-mail. A logged in user can send the information to either a single e-mail address or multiple e-mail addresses. The default values associated with the e-mail template of a cart are based on the templates defined for the cart in the Applications Manager. For more information about defining values for templates, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

The YCDEmailOrderUE user exit is called to send the details pertaining to a cart or order to the corresponding users through e-mail. The YCDEmailOrderUE user exit uses the YCD_Populate_Order_Email_8.5 service and the YCD_Send_Order_Email_8.5 service illustrated in the following figure to send the order information through e-mail.



The YCDEmailOrderUE user exit calls the getCompleteOrderDetails API to retrieve the relevant information required for sending an e-mail to the recipient’s e-mail address. The information is passed to the YCD_Populate_Order_Email_8.5 service, and the e-mail is sent to the recipient using the YCD_Send_Order_Email_8.5 service, as illustrated in following figure:



Ensure that an agent server is associated with YCD_Send_Order_Email_8.5 service. A user must run the agent server to be able to send the e-mail. For more information about configuring agent servers, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide.*

By default Sterling Web provides the following templates:

- YCD_ItemDetailsEmail.xsl.sample
- YCD_Draft_Email_Order.xsl.sample
- YCD_Confirmed_Email_Order.xsl.sample

YCD_ItemDetailsEmail.xsl.sample

The `YCD_ItemDetailsEmail.xsl.sample` template is used for e-mailing product details. Ensure that you rename this template to `YCD_ItemDetailsEmail.xsl`. A user can customize this template. This template is passed as an input to the sendMail API.

The `YCD_ItemDetailsEmail.xsl.sample` is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:emailformatters="com.yantra.pca.email.formatters"
    xmlns:java="java"
    exclude-result-prefixes="java emailformatters">
```

```
<xsl:template match="/">
<HTML>
 <xsl:comment>RECIPIENTS=Undisclosed </xsl:comment>
<xsl:comment>FROM=sales@yourcompany.com</xsl:comment>
<xsl:comment>SUBJECT=Item Details</xsl:comment>
<xsl:comment>CONTENT_TYPE=text/html</xsl:comment>
<HEAD>
  <STYLE TYPE="text/css">
.table  {
  padding:0;
  font-size: 12;
  font-family: Tahoma;
  font-weight: normal;
   color: #000000;
   width: 100%;
  border: 1;
.tablecolumn{
  padding-left:2px;
  padding-right:2px;
  padding-top: 0px;
  padding-bottom: 0px;
  font-size: 12;
  vertical-align: top;
  text-align: left;

numerictablecolumn{
  padding-left:2px;
  padding-right:2px;
  padding-top: 0px;
  padding-bottom: 0px;
  vertical-align: top;
  text-align: right;
  font-size: 12;

.tablecolumnheader {
  border-left:1px solid buttonhighlight;
  border-right:1px solid buttonshadow;
  border-top:1px solid buttonhighlight;
  border-bottom:1px solid buttonshadow;
  PADDING-LEFT: 2px;
  PADDING-RIGHT: 2px;
  PADDING-top: 0px;
  PADDING-bottom: 0px;
  VERTICAL-ALIGN: middle;
  HORIZONTAL-ALIGN: center;
  BACKGROUND-COLOR: #e0e0e0;
  TEXT-ALIGN: left
</STYLE>

</HEAD>

<BODY topmargin="0" leftmargin="0">
<xsl:apply-templates select="ItemList"/>

</BODY>
</HTML>
</xsl:template>

<xsl:template match="ItemList">
<BR/><BR/><BR/><P><font><B>Item ID : <xsl:value-of
select="Item/@ItemID"/></B></font></P>
<BR/>
<TABLE BORDER="2" WIDTH="50%" CELLSPACING="0"  CELLPADDING="3"
borderColor="#e0e0e0">
<thead>
<tr>
<td class="tablecolumnheader">
Item Description
</td>
<td class="tablecolumnheader" style="text-align:right;">
Unit Price
</td>
<td class="tablecolumnheader" style="text-align:right;">
```

```
UnitOfMeasure
</td>

</tr>
</thead>
<tr class="Items">
<td class="tablecolumn" >
  <xsl:value-of select="Item/PrimaryInformation/@Description"/>
</td>
<td class="numerictablecolumn">
<xsl:value-of select="Item/ComputedPrice/@UnitPrice"/>   <xsl:value-of
select="@Currency"/>
</td>
<td class="tablecolumn" >
  <xsl:value-of select="Item/@UnitOfMeasure"/>
</td>
</tr>

</TABLE>
<P></P>
<BR/><BR/>Thank you for shopping with us and hope you visit our site
soon.<BR/><BR/>
Sincerely,<BR/>
Customer Service Manager.<BR/><BR/>
</xsl:template>
</xsl:stylesheet>
```

YCD_Draft_Email_Order.xsl.sample

The `YCD_Draft_Email_Order.xsl.sample` template is used for e-mailing draft order details. Ensure that you rename this template to `YCD_Draft_Email_Order.xsl`. A user can customize this template.

The YCD_Draft_Email_Order.xsl.sample xml is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:emailformatters="com.yantra.pca.email.formatters"
    xmlns:java="java"
    exclude-result-prefixes="java emailformatters">
```

```
<xsl:include href="/template/email/ycd/common/YCD_brand.xsl.sample"/>
<xsl:include href="/template/email/ycd/common/YCD_personalInfo.xsl.sample"/>
<xsl:include href="/template/email/ycd/common/YCD_miscUtils.xsl.sample"/>
<xsl:include href="/template/email/ycd/common/YCD_itemData.xsl.sample"/>
<xsl:include
href="/template/email/ycd/common/YCD_summaryOfCharges.xsl.sample"/>
<xsl:include href="/template/email/ycd/common/YCD_paymentInfo.xsl.sample"/>


<xsl:variable name="Brand">
<xsl:call-template name="BrandName">
<xsl:with-param name="BrandCode" select="/Order/@EnterpriseCode"/>
</xsl:call-template>
</xsl:variable>
<xsl:variable name="BrandPhoneNumber">
<xsl:call-template name="BrandPhoneNumber">
<xsl:with-param name="BrandCode" select="/Order/@EnterpriseCode"/>
</xsl:call-template>
</xsl:variable>
<xsl:variable name="Currency"><xsl:value-of
select="/Order/PriceInfo/@Currency"/></xsl:variable>


<xsl:template match="/">
<HTML>
<xsl:call-template name="applyStyle"/>
<BODY topmargin="0" leftmargin="0" STYLE="font:normal 10pt Tahoma">
<xsl:apply-templates select="Order"/>
</BODY>
</HTML>
</xsl:template>


<xsl:template name="break">
<xsl:param name="text" select="."/>
<xsl:choose>
  <xsl:when test="contains($text, '&#xa;')">
  <xsl:value-of select="substring-before($text, '&#xa;')"/>
  <br/>
  <xsl:call-template name="break">
```

```
<xsl:with-param name="text" select="substring-after($text,'&#xa;')"/>
  </xsl:call-template>
  </xsl:when>
  <xsl:otherwise>
  <xsl:value-of select="$text"/>
  </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<xsl:template match="Order">
<xsl:call-template name="break">
<xsl:with-param name="text" select="@Memo"/>
</xsl:call-template>
<xsl:if test="@Memo and not(@Memo = '')">
<hr/>
</xsl:if>
Dear Customer,
<p/>
Thank you for shopping <xsl:value-of select="$Brand"/>.com.
<br/><br/>
Your Current Cart:
<br/>
Cart Number: <xsl:value-of select="@OrderNo"/>
<br/>
Cart Name: <xsl:value-of select="@OrderName"/>
<br/>
Cart Date: <xsl:value-of
select="emailformatters:TextEmailFormat.formatDate(@OrderDate, 'MMM dd,
yyyy')"/>
<br/><br/>
<xsl:call-template name="OrderStatusMessage"><xsl:with-param name="Brand"
select="$Brand"/></xsl:call-template>
<br/>
Here&apos;s a summary of your order placed on <xsl:value-of
select="emailformatters:TextEmailFormat.formatDate(@OrderDate, 'MMM dd,
yyyy')"/>:
<p/>
<xsl:call-template name="ShippingDetails">
```

```
<xsl:with-param name="carrierServiceCode" select="@CarrierServiceCode"/>
</xsl:call-template>

<p/>
<b>Shipping Address:</b>
<xsl:apply-templates select="PersonInfoShipTo"/>
<p/>
<b>Items in your cart</b>
<table class="table" >
<xsl:call-template name="ItemDataHeader"/>
<xsl:apply-templates select="OrderLines//OrderLine"/>
</table>
<p/>

<b>Current Cart Total</b>
<table class="table" >
<xsl:apply-templates select="OverallTotals[1]"/>
</table>

You may receive status updates regarding your order, and we will send a
confirmation e-mail as soon as we ship your order.
<xsl:call-template name="UnsubscribeMessage"><xsl:with-param name="Brand"
select="$Brand"/><xsl:with-param name="CustomerEmailID"
select="@CustomerEMailID"/></xsl:call-template>

<p/>
<xsl:call-template name="StandardClosingMessage"><xsl:with-param name="Brand"
select="$Brand"/><xsl:with-param name="BrandPhoneNumber"
select="$BrandPhoneNumber"/></xsl:call-template>
</xsl:template>

<xsl:template match="OrderLine">
  <xsl:variable name="LineQuantity"><xsl:value-of
select="./@OrderedQty"/></xsl:variable>
  <xsl:variable name="_UnitPrice"><xsl:value-of
select="format-number(LinePriceInfo/@UnitPrice , '.00')"/></xsl:variable>
    <xsl:variable name="LineTotal"><xsl:value-of
select="format-number($_UnitPrice * $LineQuantity, '.00')"/></xsl:variable>
```

```
  <xsl:call-template name="ItemData">
<xsl:with-param name="ItemDesc" select="Item/@ItemDesc"/>
<xsl:with-param name="UnitPrice" select="$_UnitPrice"/>
<xsl:with-param name="Quantity" select="$LineQuantity"/>
<xsl:with-param name="TotalPrice" select="$LineTotal"/>
<xsl:with-param name="UnitOfMeasure" select="Item/@UnitOfMeasure"/>
<xsl:with-param name="Currency" select="$Currency"/>
    </xsl:call-template>
</xsl:template>

<xsl:template match="PersonInfoShipTo">
  <xsl:call-template name="FormatYantraAddress"/>
</xsl:template>

<xsl:template match="OverallTotals">
  <xsl:call-template name="DisplaySummaryOfCharges">
<xsl:with-param name="LineSubTotal" select="@LineSubTotal"/>
  <xsl:with-param name="TotalCharges" select="@GrandCharges"/>
  <xsl:with-param name="TotalTax" select="@GrandTax"/>
  <xsl:with-param name="TotalDiscount" select="@GrandDiscount"/>
  <xsl:with-param name="GrandTotal" select="@GrandTotal"/>
  <xsl:with-param name="Currency" select="$Currency"/>
    </xsl:call-template>
</xsl:template>
<xsl:template match="PaymentMethod">
<xsl:if test="./@PaymentType='CREDIT_CARD'">
<xsl:call-template name="PaymentTypeCreditCard">
<xsl:with-param name="CreditCardType" select="./@CreditCardType"/>
<xsl:with-param name="DisplayCreditCardNo" select="./@DisplayCreditCardNo"/>
</xsl:call-template>
</xsl:if>
<xsl:if test="./@PaymentType='CHECK'">
<xsl:call-template name="PaymentTypeCheck">
<xsl:with-param name="CheckNo" select="./@CheckNo"/>
</xsl:call-template>
```

```
</xsl:if>
<xsl:if test="./@PaymentType='SVC'">
<xsl:call-template name="PaymentTypeCheck">
<xsl:with-param name="PaymentAmount" select="./@PaymentAmount"/>
<xsl:with-param name="DisplaySvcNo" select="./@DisplaySvcNo"/>
</xsl:call-template>
</xsl:if>
<br/>
</xsl:template>
</xsl:stylesheet>
```

YCD_Confirmed_Email_Order.xsl.sample

The `YCD_Confirmed_Email_Order.xsl.sample` template is used for e-mailing the details of a confirmed order. Ensure that you rename the template to `YCD_Confirmed_Email_Order.xsl.` A user can customize this template.

The `YCD_Confirmed_Email_Order.xsl.sample xml is as follows:`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:emailformatters="com.yantra.pca.email.formatters"
    xmlns:java="java"
    exclude-result-prefixes="java emailformatters">
<xsl:include href="/template/email/ycd/common/YCD_brand.xsl.sample"/>
<xsl:include href="/template/email/ycd/common/YCD_personalInfo.xsl.sample"/>
<xsl:include href="/template/email/ycd/common/YCD_miscUtils.xsl.sample"/>
<xsl:include href="/template/email/ycd/common/YCD_itemData.xsl.sample"/>
<xsl:include
href="/template/email/ycd/common/YCD_summaryOfCharges.xsl.sample"/>
<xsl:include href="/template/email/ycd/common/YCD_paymentInfo.xsl.sample"/>

<xsl:variable name="Brand">
<xsl:call-template name="BrandName">
<xsl:with-param name="BrandCode" select="/Order/@EnterpriseCode"/>
</xsl:call-template>
</xsl:variable>
<xsl:variable name="BrandPhoneNumber">
<xsl:call-template name="BrandPhoneNumber">
```

```
<xsl:with-param name="BrandCode" select="/Order/@EnterpriseCode"/>
</xsl:call-template>
</xsl:variable>
<xsl:variable name="Currency"><xsl:value-of
select="/Order/PriceInfo/@Currency"/></xsl:variable>

<xsl:template match="/">
<HTML>
<xsl:call-template name="applyStyle"/>
<BODY topmargin="0" leftmargin="0" STYLE="font:normal 10pt Tahoma">
<xsl:apply-templates select="Order"/>
</BODY>
</HTML>
</xsl:template>

<xsl:template name="break">
  <xsl:param name="text" select="."/>
  <xsl:choose>
  <xsl:when test="contains($text, '&#xa;')">
   <xsl:value-of select="substring-before($text, '&#xa;')"/>
  <br/>
  <xsl:call-template name="break">
<xsl:with-param name="text" select="substring-after($text,'&#xa;')"/>
</xsl:call-template>
  </xsl:when>
  <xsl:otherwise>
<xsl:value-of select="$text"/>
  </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<xsl:template match="Order">
<xsl:call-template name="break">
<xsl:with-param name="text" select="@Memo"/>
</xsl:call-template>
<xsl:if test="@Memo and not(@Memo = '')">
<hr/>
```

```
</xsl:if>
Dear Customer,

<p/>
Thank you for shopping <xsl:value-of select="$Brand"/>.com. Your order number
is <xsl:value-of select="@OrderNo"/>.
<xsl:call-template name="OrderStatusMessage"><xsl:with-param name="Brand"
select="$Brand"/></xsl:call-template>
<br/>
Here&apos;s a summary of your order placed on <xsl:value-of
select="emailformatters:TextEmailFormat.formatDate(@OrderDate, 'MMM dd,
yyyy')"/>:
<p/>

<xsl:call-template name="ShippingDetails">
<xsl:with-param name="carrierServiceCode" select="@CarrierServiceCode"/>
</xsl:call-template>
<p/>
<b>Shipping Address:</b>
<xsl:apply-templates select="PersonInfoShipTo"/>

<p/>
<b>Items in this order</b>
<table class="table" >
<xsl:call-template name="ItemDataHeader"/>
<xsl:apply-templates select="OrderLines//OrderLine"/>
</table>

<p/>

<b>Summary of Charges</b>
<table class="table" >
<xsl:apply-templates select="OverallTotals[1]"/>
</table>

<p/>
<b>Payment Info</b>
<br/>
```

```
<xsl:apply-templates select="PaymentMethods//PaymentMethod"/>


<p/>
You may receive status updates regarding your order, and we will send a
confirmation e-mail as soon as we ship your order.
<xsl:call-template name="UnsubscribeMessage"><xsl:with-param name="Brand"
select="$Brand"/><xsl:with-param name="CustomerEmailID"
select="@CustomerEMailID"/></xsl:call-template>


<p/>
<xsl:call-template name="StandardClosingMessage"><xsl:with-param name="Brand"
select="$Brand"/><xsl:with-param name="BrandPhoneNumber"
select="$BrandPhoneNumber"/></xsl:call-template>
</xsl:template>


<xsl:template match="OrderLine">
  <xsl:variable name="LineQuantity"><xsl:value-of
select="./@OrderedQty"/></xsl:variable>
  <xsl:variable name="_UnitPrice"><xsl:value-of
select="format-number(LinePriceInfo/@UnitPrice , '.00')"/></xsl:variable>
  <xsl:variable name="LineTotal"><xsl:value-of
select="format-number($_UnitPrice * $LineQuantity, '.00')"/></xsl:variable>


  <xsl:call-template name="ItemData">
<xsl:with-param name="ItemDesc" select="Item/@ItemDesc"/>
<xsl:with-param name="UnitPrice" select="$_UnitPrice"/>
<xsl:with-param name="Quantity" select="$LineQuantity"/>
<xsl:with-param name="TotalPrice" select="$LineTotal"/>
<xsl:with-param name="UnitOfMeasure" select="Item/@UnitOfMeasure"/>
<xsl:with-param name="Currency" select="$Currency"/>
   </xsl:call-template>
</xsl:template>


<xsl:template match="PersonInfoShipTo">
  <xsl:call-template name="FormatYantraAddress"/>
</xsl:template>


<xsl:template match="OverallTotals">
<xsl:call-template name="DisplaySummaryOfCharges">
```

```
<xsl:with-param name="LineSubTotal" select="@LineSubTotal"/>
<xsl:with-param name="TotalCharges" select="@GrandCharges"/>
<xsl:with-param name="TotalTax" select="@GrandTax"/>
<xsl:with-param name="TotalDiscount" select="@GrandDiscount"/>
<xsl:with-param name="GrandTotal" select="@GrandTotal"/>
<xsl:with-param name="Currency" select="$Currency"/>
  </xsl:call-template>
</xsl:template>
<xsl:template match="PaymentMethod">
<xsl:if test="./@PaymentType='CREDIT_CARD'">
<xsl:call-template name="PaymentTypeCreditCard">
<xsl:with-param name="CreditCardType" select="./@CreditCardType"/>
<xsl:with-param name="DisplayCreditCardNo" select="./@DisplayCreditCardNo"/>
</xsl:call-template>
</xsl:if>

<xsl:if test="./@PaymentType='CHECK'">
<xsl:call-template name="PaymentTypeCheck">
<xsl:with-param name="CheckNo" select="./@CheckNo"/>
</xsl:call-template>
</xsl:if>

<xsl:if test="./@PaymentType='SVC'">
<xsl:call-template name="PaymentTypeCheck">
<xsl:with-param name="PaymentAmount" select="./@PaymentAmount"/>
<xsl:with-param name="DisplaySvcNo" select="./@DisplaySvcNo"/>
</xsl:call-template>
</xsl:if>
<br/>
</xsl:template>
</xsl:stylesheet>
```

Validation of E-Mail Addresses

Sterling Web validates the e-mail addresses to which a user has to send the cart or order details based on the following criteria:

- The first character must be an alphabet.
- The symbol @ must be present.

- There must be at least one character before @.
- There must be least one character between at and the period.
- There must be between 2–4 characters after the period.
- There must not be spaces in the e-mail address.
- There must be a semi-colon(;) between addressees.

If any of these criteria is not met, Sterling Web displays an error message to notify the user of the incorrect criteria.

End-User Impact

This section explains the end-user impact of this functionality:

- The **Email Page** hyperlink is not displayed if a user is a guest user.
- A user can send the details pertaining to a cart or order only if the user has the necessary permissions.

Implementation

This section explains the configurations for this functionality:

- To use this feature, implement the YCDEmailOrderUE user exit. For more information about defining user exit implementations, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- To use this feature, associate an agent server with the YCD_Send_Order_Email_8.5 service. For more information about configuring agent servers, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- You must assign the necessary permissions to user groups to enable them to send the details pertaining to a cart or order through e-mail. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

Copy a Cart or an Order

When creating a cart, a user may want to reuse a cart that already exists in an order. Similarly, a user may want to reuse an existing order. This is especially useful, for example, when a user places an order for a particular list of products every month, with the list varying only slightly from month to month.

Solution

Sterling Web™ enables a user to copy an existing cart or create a new cart from an existing order. The copyOrder API is called to copy a cart or create a cart from a confirmed order, including all the attributes defined in the Copy Order template in the Applications Manager. For more information about defining a process type’s templates, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*. Buyer administrators have the option to reuse a buyer user order for themselves. A buyer administrator can reuse an order placed by the user of a child organization. A user can also increase the quantity of the line items in the new cart when reusing an order.

However, a user cannot reuse the following type of orders in Sterling Web:

- A cancelled order
- An orders containing Delivery Service
- An order containing Provided Service

A user can edit the following entities copied or created as part of the reuse process:

- Order-level Sold-To address
- Order-level Ship-To address
- Line-level Ship-To address
- Order-level Shipping Method
- Line-level Shipping Method
- Order-level Bill-To address

The line-level SHIP COMPLETE flag entity and the configuration of a reconfigurable product also get copied during the reuse process.

The following entities are not copied during the reuse process:

- Tax information
- Store Pick Up selections
- Order-level Requested Delivery Date
- Line-level Requested Delivery Date
- Payment method
- Gift cards
- Coupons

End-User Impact

This section explains the end-user impact of this functionality:

- A user can copy a cart or create a new cart from an existing order only if the user has the necessary permissions.
- A user can reorder an order with the `SupportLevel` attribute set as medium, only if the appropriate user permissions are configured for the user.
- The address of the order owner gets copied when another user reuses the order. The new order date is set to the current date after reordering.
- The currency and line-level details (Delivery method and Quantity) are copied from the original order to the new cart page during reorder. However, line-level details (Delivery method and Quantity) are not copied when a user reorders a single item.
- For some particular orders, the users do not have the permission to order an item that is copied from an original order when the status of the item is not entitled to be ordered now.
- If a user does not have the permission to order currently, the user cannot reorder even a single item in the corresponding order.

Implementation

This section explains the configurations for this functionality:

- To use this functionality, define the order and order line attributes when copying a cart or creating a cart from an order in the Copy Order template defined at the Hub level in the Applications Manager.
- You must assign the necessary permissions to the corresponding user groups to enable them to copy a cart or create a cart from an existing order. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide.*
- You must set the appropriate resource permissions for the users to enable them reorder the orders with the `SupportLevel` attribute set as medium.
- After reusing an existing order, repricing is carried out for the items in a new cart with respect to pricing rules.

Reference Implementation

None.

Cancelling an Order

After placing an order, a user may want to cancel either the entire order or certain line items in the order because the user discovers, for example, that these products are available at a lesser price elsewhere.

Solution

Sterling Web™ enables a user to cancel some or all the order lines based on the user permissions and order status modification rules. The YCD_CancelOrderNotification_3.0 service must be configured if the user wants to receive an e-mail when an order is cancelled. The changeOrder API is called to cancel the order or order lines. The getCompleteOrderDetails API is called to update the order information after the cancellation of the order or order lines.

End-User Impact

This section explains the end-user impact of this functionality:

- A user can cancel an order or order lines only if the user has the necessary permissions.
- A user can cancel an order or order lines only if the order status modification rules allow cancellation of an order or order lines.
- If the order is in a state where the modifications made to it are yet to be saved, only the user who made those changes can cancel the order or order lines.
- A user can cancel an order with coupons only if the Change Price modification type is enabled or modification is allowed for the Cancelled status in the Order Modification rules.

Implementation

This section explains the configurations for this functionality:

- Sterling Web enables you to configure the order statuses for which cancellation of order or order lines is allowed. For more information about defining modification rules, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- You must assign permissions to user groups to enable them to cancel an order. For more information about administering user group permissions, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.
- Ensure that the YCD_CancelOrderNotification_3.0 service is configured to enable a user to receive e-mail whenever an order is cancelled. For more information about creating a service, refer to the *Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Reference Implementation

None.

Product Configuration

Users must be able to configure the products they are interested in buying so that they can build the products according to their requirements. For example, a user who is interested in buying a desktop computer may want to select the appropriate RAM, hard disk, or monitor for the computer based on the user's requirements.

While a configurable products is a product that can be customized by a user at the time of purchase, a preconfigured products is a product whose components have already been configured. Preconfigured products and configurable products are products that have been set up as model products, using the Visual Modeler. Each model consists of option classes, and option items within these option classes. Option items are the selections made by a user to configure the corresponding products. For example, the option classes are RAM, hard disk, and monitor, and the options that are displayed under these option classes are the option items.

Solution

The Sterling Web™ application enables a user to configure products through the Sterling Configurator™ before placing an order. You can offer users the complete range of available products options available, including preconfigured products containing predefined choices. Users can purchase the preconfigured products as is or use them as the starting point for performing additional configurations.

The `sic_properties.zip` provide the property files that are utilised by the Sterling Configurator. These propertiy files must be placed in the properties folder configured in the Applications Manager.

The following files are located in the properties folder.

- `configurator.properties` - This file comprises the basic properties that can be used to modify the functionality of Sterling Configurator. For example, the number of models that can be cached by Sterling Configurator can be modified by making the appropriate changes to this file. This file comprises the pricingType property that defines whether the price for an item is taken from the pricelist or from a model. The valid values that can be defined for the pricingType property are as follows:
  - STATIC_PRICING - To pick the the price of an item from the model.
  - DYNAMIC_PRICING - To pick the price of an item from the price list.
  - OVERRIDE_PRICING - To pick the price from the price list. If the price for the item is not found in the price list, the price defined for the model is used.
- `controls.properties` - This file defines the set of controls that are available for Sterling Configurator. A user can control the functionality of the option classes and option items displayed in the Visual Modeler.

Note: DynamicInstantiationControlHandler is a control handler class that dynamically adds child option items to a model when it is retrieved from the cache and removes the dynamic items when the model is returned to the model cache.

- `functionHandlers.properties` – The function handlers are declared in this file. Sterling Configurator provides a rule engine that is used to evaluate the rules defined for each model. The rule engine can invoke custom functions to handle scenarios where existing functions are unable to solve a configuration specification. . The `lookupValues.properties` property file is used by the sample

LookupFunctionHandler. The `webServiceLookup.properties` property file is used by the sample WebServiceLookup handler.

All the paths comprising the location where the models, properties files, and rules are stored, are configured in the Applications Manager. The following jars are located in the `<INSTALL_DIR>\jar\smcfs\9.0` folder and must be copied to the folder specified in the path configured for the rules.

- `cmgt-rulesEngine.jar`
- `cmgt-configuredItem.jar`
- `cmgt-configurator.jar`

For more information about configuring the Sterling Configurator rules, refer to the *Visual Modeler: Application Guide*.

A user can configure a product by clicking the **Build** button. The **Build** button is available in the Cart Details page when a configurable product is added to the corresponding cart. The **Build** button is also available in the Product List page of a configurable product. The price of an item shown in the Sterling Configurator page will be based on the unit price of the item in the catalog. Discounts will be considered in the Sterling Configurator page only if they are available for unit price. There may be differences in the price that is displayed on the Sterling Configurator page and the Cart Details page because pricing rules will be applied on the Cart Details page.

A user can configure products through the Business Center application by setting the value of IsConfigurable as Y. For more information about setting values, refer to the *Business Center Item Administration Guide*. For more information based on the way the user interface display property of the model item is configured, the Configuration page is displayed. For more information about working with display properties, refer to the *Sterling Multi-Channel Selling Solution Administration Guide*.

Notes:

- To configure a product using the Sterling Configurator, ensure that the product's Effective Start Date and Effective End Date are configured in the Business Center Application. For more information about primary information pertaining to an item, including the Effective Start Date and Effective End Date, refer to the *Business Center: Item Administration Guide*.
- The image of a selected item is displayed only when you select the radio button or the check box next to it in the step-wise configurator.

Tabbed and Normal Configuration

In a Tabbed configuration layout, the option classes for a model are displayed under the respective tabs, with the corresponding option items listed under each option class. In the Normal configuration layout, the option classes are displayed on a single page along with the corresponding option items. Whenever an action is performed on the Configuration page of the Sterling Configurator, a request is sent to the server from the Configuration page and the ConfiguratorController is called to handle that action.

Depending on the action that is called in the Configuration page, the system first performs one or many of the following operations, which are also referred to as initializeConfigurations, using the Configurator business object:

- Get and process existing XMLs
- Get prices
- Apply entitlements and pricing

- Apply picks
- Fire rules
- Compute prices
- Get list of controls
- Get list of existing tabs

Following are the actions that can be called in the Tabbed configuration and the Normal configuration page:

- configure — This is the main action handled by the Sterling Configurator. This action is called when the Sterling Configurator is launched, a user clicks on a tab, or a user selects an option item in the Configuration page. This action takes the results of the initializeConfiguration operation as the input, sets the appropriate attributes on request, and sets the value of the configURL parameter to display the appropriate page.
- addToCart — This action is called when a user adds a configurable product to either a cart or an order. This action takes the results of the initializeConfiguration operation and builds the bill of material (BOM) that is passed to the cart. The control is passed to the URL specified in the returnURL parameter. During the checkout process or when a user is submits an order, BOM validations are disabled if the `SkipBOMValidations` attribute is set to Y in either the changeOrder API or the createOrder API. BOM validations are not performed if an order is in Draft status. The BOM validations can be activated or deactivated during the submission of a request or when a user performs a checkout. To configure the BOM validations during checkout, ensure that the **Validate Configurable Items When Viewing a Cart** rule is configured in the Channel Applications Manager. For more information about configuring the **Validate Configurable Items When Viewing a Cart** rule, refer to the *Channel Applications Manager: Configuration Guide*. To configure the BOM validations during the submission of a request to the server, ensure that the **Validate Configurable Items During Checkout** rule is configured in the Channel Applications Manager. For more information about configuring the V**alidate Configurable Items During Checkout** rule, refer to the *Channel Applications Manager: Configuration Guide*.
- To configure the BOM validations during the submission of a request to the server, ensure that the Validate Item rule is configured in the Applications Manager. For more information about configuring the Validate Item rule, refer to the Sterling Distributed Order Management: Configuration Guide.
- summary — This action is called when a user clicks the **Summary** button. It builds the Sterling Configurator BOM based on the results of the initializeConfiguration operation and sets the request attributes required to display the Summary page.
- subModelConfig — This action is called when a user navigates from a parent model to a submodel. This action gets the submodel, sets the necessary input properties from the parent model and the attributes required to display the submodel, and displays the submodel for configuration.
- subModelReturn — This action handles the transition from a parent model to a sub model. It takes the set of output properties from the submodel, adds them to the parent model, and displays the parent model again. A parent model is configured with a sub model through the Visual Modeler application. A user can specify sub model validation by setting the CONFIG: VALIDATE SUBMODEL property.
- showRuleTrace —This action is called when a user clicks the **Debug** button when the Sterling Configurator is run in debug mode. It sets the property pool and trace messages based on the results of the initializeConfiguration operation, and sets the attributes necessary to display the rule trace.
- resolveConflict — This action is called when a user clicks the **Resolve** button to resolve a constraint violation. It collects information about the item that caused the constraint violation and builds the set of alternatives that can be used to resolve the constraint. These alternatives are displayed on the Resolver page.

- conflictResolution — This action is called from the Resolver page to resolve a constraint violation. The resolution selected by the user is applied to the model by removing the product that caused the constraint violation and selecting the product that the user specified in the Resolver page.
- testCart — This action is used to display the test cart. When a model is launched from the Catalog page, the returnURL specifies the standard Add to Cart action which adds the configuration to the cart. However, when the Configurator is launched through the Visual Modeler, a different returnURL is provided. This action transfers the control to a page that simulates the task of adding the configuration to the cart, with additional functionality available for debugging.

  The Test Cart page comprises the following tabs:

  - Test Cart— This tab displays the BOM to be displayed in the cart.
  - BOM Details—This tab displays the BOM comprising the hidden products. The editable BOM page displays the actual XML representation of the BOM, which can be used by the Visual Modeler for modifying and reconfiguring the punching of the product with the modified BOM. The GenerateConfigurationBOM API is called to build the XML representation of the BOM.
  - Launch New Model — This tab enables the Visual Modeler to launch a new configuration model by passing the state of the existing model.
- configstatus — This action displays the Configuration Status page. This page is available only when the Sterling Configurator is run in the debug mode. This action allows a user to see the state of the cached configuration models and clear the cache, if required.

Step-Wise Configuration

In the step-wise configuration layout, each step involved in configuring a products is displayed as a tab. Each tab comprises option classes pertaining to that step. You can associate images with models, option classes, and option items. For more information about these image properties, refer to the *Visual Modeler Application Guide*. Under each tab, the corresponding option classes are displayed as icons and have an image and description associated with them. If no image is provided, a default image is displayed. If there are multiple option classes, a user can scroll to view the additional option classes. The step-wise configuration can be launched by calling the getConfigurationModel API. The Sterling Configurator is launched by a user by invoking the /configurator/configure.action. This action passes the name of the model to be configured as the parameter and calls the ConfiguratorService.managePicks API to get the set of picks to be displayed in the Configurator.

The ConfiguratorService.managePicks API retrieves the following information:

- The tabs that are displayed for the model.

  After the Sterling Configurator is loaded, an AJAX call is made to retrieve and display the tabs. The Sterling Configurator controller processes the AJAX call and generates a list of tabs and details such as tab name, guiding text, and the status of the tab. Whenever the AJAX call is processed, the controller verifies from the cache whether the tab information is available for this model. If the information is available, the controller retrieves the cached information and builds the JSON response.

  The information used to build this response is based on the cached tabs and messages retrieved from the last call to the Configurator Service. After all the tabs are retrieved, the first tab on the user interface is activated. When a tab is activated, an AJAX call is made to retrieve the list of option classes associated with that tab. When the controller processes the AJAX call, a JSON object containing the list of option classes for a given tab is created. For each option class, the option class name, icon, and status are returned. When processing this call, the controller retrieves the requested information from the controller's cache. The user can view the selected tabs with the help of visual cues. Each tab comprises the guiding text at the top of the page that is displayed when the tab is selected. The guiding text for the tabs is stored as a list property defined in the model for a product. The Sterling Configurator reads this property and constructs the tab objects to set the guiding text for each tab. A tab within a tab configuration displays a collection of option classes and option items. When a new tab is selected, a request is sent to the server to render the contents of the new tab and return it back to the client. Each time a user navigates between tabs, the configuration page is reloaded to display the current tab. getTabs is the name of the action that retrieves the tab names and message indicators, if any, to be displayed. If any of the messages pertains to an item pertaining to an option class, a message indicator corresponding to the tab is set on that tab.

- The list of top-level option classes defined in the model.

  To retrieve the list of option classes, an asynchronous call is made to the server. After the list of option classes is retrieved, the Configurator saves the information and renders only those option classes that are associated with the selected tab. By default, the first option class on the page is active and highlighted with a visual cue. All the option items under pertaining to this option class are displayed under a tab. The Help me decide hyperlink displays detailed information for each of the option item available within the selected option class. The user must then click the **Submit** button to submit the selection. Whenever a user selects an option or enters a value in a text box, the selection is added to the current pick. The managePicks action manages the selection and removal of an option item by invoking the processConfigurationPicks API.

- The property pool that determines how certain elements of the model are displayed.

  The property pool is used while rendering the option items within an option class. The properties in the property pool decide whether a product is visible or not, whether an image is displayed or not, and so on.

- The list of messages displayed in the messages panel.

  The getMessage action retrieves the list of messages displayed in the messages panel. Each message comprises the type of message, the text of the message, and the information necessary to allow the Sterling Configurator to navigate to the source of the message

- The summary displayed in the summary panel.

  Whenever user selects an option item, the page makes an AJAX call to the server to retrieve the summary information. If this call was made in the past, the controller retrieves the summary information cached from the previous call. The controller processes this request and retrieves the summary information specified for the selected model. The getPricingSummary action returns pricing summary details such as base price, and order-level pricing discounts. The summary page displays the price of the selected items after applying pricing rules, if any. For more information about pricing rules, refer to the *Business Center Pricing Administration Guide*. The pricing information is calculated when a pick is added to a configuration and is cached on the server. The getPricingSummary action retrieves the cached information and sends it to the JSON object to be displayed on the user interface.

For each option class that has been selected by the user, the corresponding option items are displayed with their price. An image of the option item that is currently selected is also displayed. If the user moves the pointer over an option item, the image of this option item is displayed. However, the image will not be displayed if you have configured not to display the image in the Visual Modeler application. The displayChildren action invokes the getConfigurationModel API to get the option items of an option class.

On the configuration page, if a user selects a single selection control, the addPick action is invoked. This action removes the previous pick and adds the new pick. If the user selects a check box or a multiselect control, the previous selection is not removed and the addPick action is invoked to pick a new selection. When a selection is made in an option class, the Sterling Configurator calls the processConfigurationPicks API with the new pick information. This API comprises a new set of picks cached along with messages resulting from the call. This cached information is available for responding to any asynchronous calls made to the server.

The following information can be cached:

- Tabs

  By default, the first option class for a model is active, highlighted with a visual cue, and is displayed under a tab. Whenever a tab is selected, it displays the corresponding option items.

- Picks

  The selections made by a user on the option class are defined as picks.

- Messages

  The messages are displayed in the message panel, in the user interface, based on the rules that are fired on the model.

- Summary Rail Information

  The Sterling Configurator displays the summary of selections made by a user in the Summary Rail area. The summary is updated every time the user selects a new option item. Users can view their selections. Additionally, buyer users can save the selections and purchase the configured product. You can specify the duration of time for which saved configurations will be available to the users using a cron job. This enables you to clean up old and outdated configurations.

  The Summary Rail displays the summary of option item selections under different headers. The summary headers are displayed based on the way they are configured in the Visual Modeler. You can configure the summary headers by defining a list property and attaching it to the root node of the model. If no property is set on the root node of the model, the model's tabs will be displayed as summary headers.

A pick made by the user in the Sterling Configurator may result in a constraint. The constraint table is used in a model to specify the valid configurations. All the information required to resolve the constraint is stored as properties for a product. For example, a model can be designed to support a 150 GB hard disk with 512 MB RAM. If a user selects a 256 MB RAM and a 150 GB hard disk, a constraint will occur. When an error occurs because of a constraint violation, the user can resolve the constraint by clicking on the error message. A Constraint Resolution page is displayed which enables the user to select a different combination of items and resolve the constraint. The resolveConstraint action sends the control to the Constraint Resolution page. If the user does not want to resolve any constraint, the user can click the **Cancel** button.

A user can navigate between pages by clicking the **Next** and **Previous** buttons. The getNextPreviousButtons action retrieves the text that should be displayed on the **Next** and **Previous** buttons. The text to be displayed on the **Next** and **Previous** button is determined by traversing the list of option classes retrieved by the processPicks action. The option class names in the list can be searched and then the next and previous names in the list are returned.

End-User Impact

This section explains the end-user impact of this functionality:

- Based on the configuration, the Sterling Configurator displays the Tabbed, Normal, or the Step-wise configuration page.
- When an administrator launches the Sterling Configurator with the DEBUG parameter set to TRUE, the following buttons are visible:
  - Show Rule Trace
  - Test Cart
  - Debug mode
  - Clear cache
- A user cannot launch the Sterling Configurator from stepwise layout.
- BOM validations are not performed on an order when the order is in Draft status.
- Whenever a user performs a configuration in the step-wise layout, the Sterling Configurator updates the screen automatically. The submit to server functionality is not supported in the step-wise layout.
- A user has the option to return to either the Catalog page or the Cart page by clicking the **Cancel** button in the Sterling Configurator page.
- A user can view the consolidated list of items selected during configuration for tabbed and normal configurations by clicking the **Summary** button in the Sterling Configurator page. A user can view the consolidated list of items selected during configuration for step-wise configuration by clicking the **Review And Buy** button in the Sterling Configurator page.

Implementation

This section explains the configurations for this functionality:

- You can configure whether to display the Tabbed configuration page, Normal configuration page, or step-wise configuration page is displayed for configuring a product by defining the UI: JSP FILENAME property. For more information about working with display properties, refer to the *Visual Modeler Application Guide*.
- When an order is purged, the corresponding order lines and the saved configurations are also purged. For more information about order purge, refer to the *Sterling Distributed Order Management: Configuration Guide*.
- To configure the BOM validations during checkout, ensure that the **Validate Configurable Items When Viewing a Cart** rule is enabled in the Channel Applications Manager. For more information about configuring the **Validate Configurable Items When Viewing a Cart** rule, refer to the *Channel Applications Manager: Configuration Guide*.
- To configure the BOM validations during the submission of a request to the server, ensure that the **Validate Configurable Items During Checkout** rule is configured in the Channel Applications Manager. For more information about configuring the V**alidate Configurable Items During Checkout** rule, refer to the *Channel Applications Manager: Configuration Guide*.
- To enable sub model validation, ensure that the CONFIG: VALIDATE SUBMODEL property is set in the Visual Modeler application. For more information about setting the CONFIG: VALIDATE SUBMODEL property in Visual Modeler, refer to the *Visual Modeler Application Guide*.

- To disable BOM validations during the checkout process or order submission, set the `SkipBOMValidations` attribute is set to Y in either the changeOrder API or the createOrder API. For more information about the changeOrder API and the createOrder API, refer to the *Selling and Fulfillment Foundation: Javadocs*.
- To display the images of option items available under an option class, you must set the UI: SHOW ITEM IMAGES property to "true" at the option class level in the Visual Modeler, and specify the image for the option items by setting the UI: ITEM IMAGE NAME property, in the image URL field at the option item level, appropriately. For more information about working with display properties, refer to the *Visual Modeler Application Guide*.
- A user can set UI controls, create constraints, set model groups, models, option classes, option items, and rules. For more information about setting them, refer to the *Visual Modeler Application Guide*.

Reference Implementation

None.

Sterling Configurator Best Practices

Introduction

Mapping eBusiness requirements to the features provided by Sterling Configurator can be daunting. Understanding how Sterling Configurator works in a variety of circumstances and studying examples demonstrating common usage patterns can help.

This chapter covers:

- Planning Considerations describes planning considerations for designing and implementing a Sterling Web application model.

The following are among the trade-offs to keep in mind.

- Cost factors: creation, maintenance, and performance.
- You must balance the cost of creating the model, compared with. the cost of maintaining it, compared with its expected performance. The cost of creating the model represents the one-time effort expended to develop it; the cost of maintaining the model represents the effort expended over time to maintain and enhance it, while the performance represents the execution speed of the model on a particular hardware platform. You can optimize for any one of these factors, but be mindful that trying to optimize for more than one of them means, in most cases, that you have competing goals. For example, a complex model might run fast, but would be hard to maintain.For an overview of models and best practice design principles, refer to the "Models" topic.
- "Properties" provides tips and procedures for working with properties.
- Rules describes how to define and use rules to promote efficiency in your model.
- Performance describes considerations for writing models that provide the best performance.

About Absolute and Relative Paths

This chapter refers to the use of absolute or relative paths to specify entities such as properties and rules. Paths have the following form:

<model group root node>.<path to the option item that has the property or rule>.<property name or rule name>

For example, consider the following absolute path to the property memoryProvided:

```
MXDS-7500.memory.sim256.memoryProvided
```

The model group's root node is MXDS-7500; the path to the option item that has the property memoryProvided is memory.sim256; memoryProvided is the property name.

If you plan to use a property or rule in more than one model, you can use special symbols to specify relative paths. For example, the "*" in the following path indicates that the path begins at the root of the model group hierarchy:

```
*.memory.sim256.memoryProvided
```

Beginning the path with a period (.) indicates "from the attachment point of the rule". For example, the "." in the following path indicates that "an option item called sim256 in an option class called memory in the current model".

```
.memory.sim256
```

Planning Considerations

A model represents a configurable product. When you sit down to plan a Sterling Web implementation, you start by considering how to design a model of the product. There is no one “right” way to model a particular product. On the other hand, there are an endless number of ways to create models that, while technically correct, are inefficient and hard to maintain.

Implementers’ roles: model builder only, model builder and maintainer, model maintainer only.

The implementer’s role influences their bias. For example, a consultant given a month to implement a model which will then be handed off to another group for maintenance may focus on designing the model quickly, rather than designing a model that will be easy to maintain. If the implementer will also maintain the model, the model may take longer to design and perhaps not run as fast, but will be easy to maintain long-term. Whatever your role, your goal should be to set up a model that will avoid problems down the road.

Models

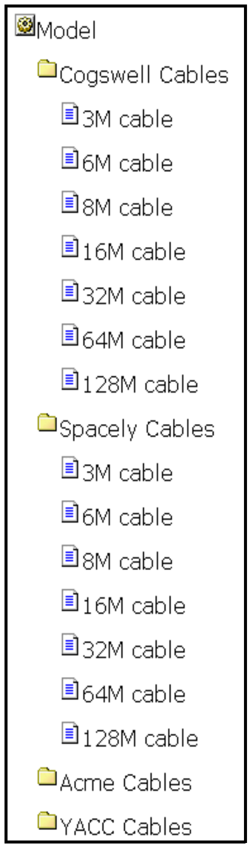
This section covers product model design considerations and provides examples of designs that work, compared with designs that work best.

Make Models as Small and Simple as Possible

Model size matters. Large models take longer to render in the browser, are harder to maintain, and during configuration the Configurator "walks" the model structure a number of times to get prices, fire rules, and so on. Keeping the model size as small as possible through the use of subassemblies and other techniques described in this chapter improves performance and decreases maintenance costs.

For example, suppose that you have a number of cable suppliers, each of which supplies a number of different lengths of cables. You want to allow the user to select the quantity, length and cable supplier.

One way to do this is to create option items in your model to represent every single one of the available options, as shown in the following figure:



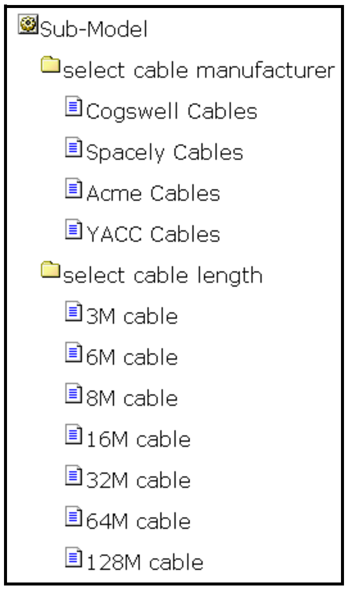
Model
- Cogswell Cables
  - 3M cable
  - 6M cable
  - 8M cable
  - 16M cable
  - 32M cable
  - 64M cable
  - 128M cable
- Spacely Cables
  - 3M cable
  - 6M cable
  - 8M cable
  - 16M cable
  - 32M cable
  - 64M cable
  - 128M cable
- Acme Cables
- YACC Cables

FIGURE 1. Cable Inventory Model by Supplier

This approach will work, although it may be a bit tedious to implement and maintain and creates a model with many options that will never be of interest to the end user.

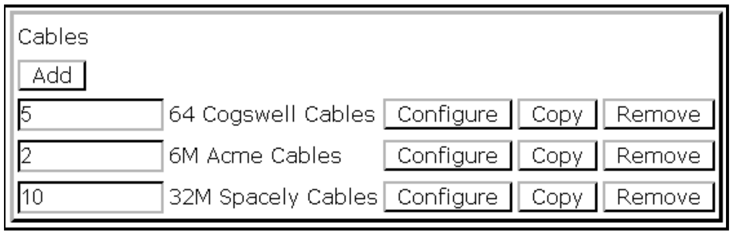
An alternative approach might be to implement the different cable length option items as an option item group, then include the cable length option item group under each of the manufacturers. This would ease maintenance: the modeler would have to look in only one place to update the cable option item information. However, this approach presents the end user with a huge list of cables to choose from and does not improve performance since there’s still a large model to walk.

A better alternative is to create a submodel that allows the user to select a cable manufacturer and length, then use dynamic instantiation to let the user add as many different cable types and lengths as necessary, as shown in the following figure:



Sub-Model
- select cable manufacturer
  - Cogswell Cables
  - Spacely Cables
  - Acme Cables
  - YACC Cables
- select cable length
  - 3M cable
  - 6M cable
  - 8M cable
  - 16M cable
  - 32M cable
  - 64M cable
  - 128M cable

The following figure shows a sample cable selection UI that uses dynamic instantiation to allow end users to configure their cable selections.



Cables

Add

5	64 Cogswell Cables	Configure	Copy	Remove
2	6M Acme Cables	Configure	Copy	Remove
10	32M Spacely Cables	Configure	Copy	Remove

FIGURE 2. Cable Selection Screen

As you can see, this approach keeps the model small. Maintaining this model will be easier since the modeler no longer has to deal with a huge number of duplicate option items, performance is better since the model is smaller, and configuration is easier for the end user since there isn’t a long list of cable types and manufacturers to look through in order to find the right one.

Use *popup-qty* Controls for Entering Quantity

Sometimes the modeler wants to allow users to select an item, then enter the number of items they want. The best way to do this is to set the Option Class Display to popup-qty. When the end user selects an item, a quantity box will display, allowing the end user to enter the number of items they want.

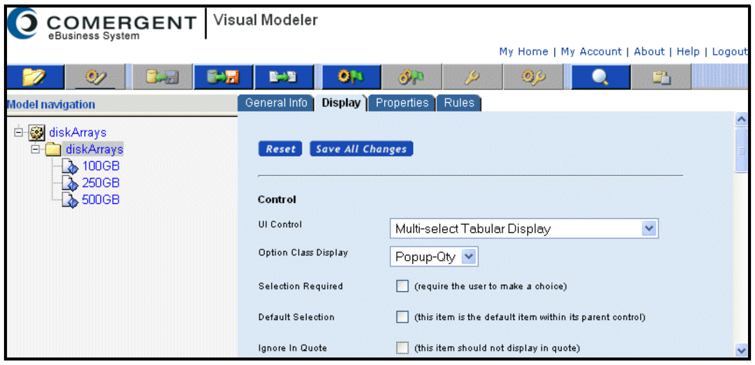
Some modelers do not like the placement of the quantity box, so instead use the User Entered Values (UEV) control to display an edit field next to the item in which users can enter a quantity. The problem is that the behavior of the popup-qty control differs significantly from the behavior of UEV controls: the popup-qty control has quantity processing built in, while UEV controls require additional work.

When an end user enters a quantity in a popup-qty box, the application automatically selects the quantity of the selected item. Any properties attached to the item are included in the configurator state (property pool), and the values of any numeric properties are multiplied by the quantity entered.

When a value is entered in a UEV control, nothing else happens. UEV controls were designed simply to capture some additional information from the user. To get the UEV to behave as a quantity, the modeler must write an expansion rule that takes the value entered in the UEV control and picks that many of the selected item. Using the value entered in the UEV control to set the _quantity property using an assignment rule will not work as expected, since this does not automatically create instances of the item's properties in the property pool.

To display a popup-qty box beside the item selected, use the popup-qty Option Class Display style and one of the tabular displays with quantity controls. This will ensure the correct number of items are selected and the correct properties are copied to the property pool.

For example, the following figure shows how to set up a popup-qty control using the Visual Modeler. From the Models and Groups panel, select the model you wish to modify, then click the Edit Model icon. The Model navigation page appears. Click the option group you wish to modify, click the Display tab, then select Multi-select Tabular Display from the UI Control drop-down list, as shown in the following figure:

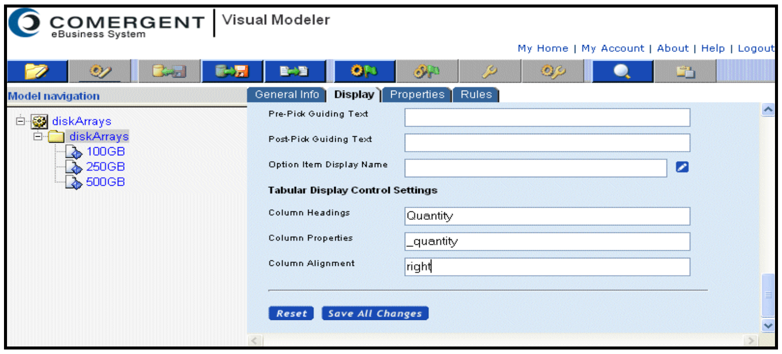


Control

UI Control	Multi-select Tabular Display
Option Class Display	Popup-Qty
Selection Required	☐ (require the user to make a choice)
Default Selection	☐ (this item is the default item within its parent control)
Ignore In Quote	☐ (this item should not display in quote)

FIGURE 3. Setting up a popup-qty control: selecting Multi-select Tabular Display

Scroll to the bottom of the page and enter the Column Headings, Column Properties, and Column Alignment settings. The following figure shows sample settings.



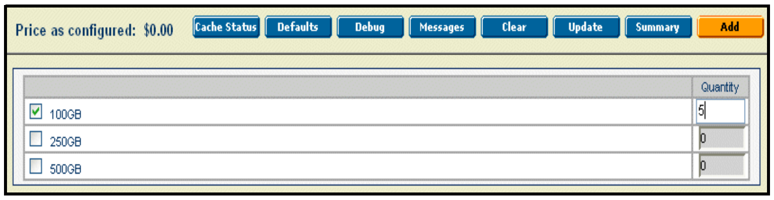
Pre-Pick Guiding Text	
Post-Pick Guiding Text	
Option Item Display Name	

Tabular Display Control Settings

Column Headings	Quantity
Column Properties	_quantity
Column Alignment	right

FIGURE 4. Sample Column Headings, Properties, and Alignment Settings

Finally, compile and test the model. You should see a popup-qty control placed as you specified on the Product Configurator page.



	Quantity
✓ 100GB	5
☐ 250GB	0
☐ 500GB	0

FIGURE 5. A Sample Product Configurator Page Showing popup-qty Controls

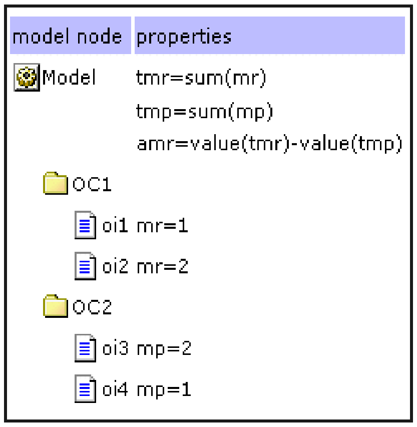
Properties

Properties are ubiquitous in the Sterling Configurator. Modelers attach properties to models, option classes and option items and then write rules that work on these properties in order to display messages, show, hide, or select items, and even set the values of other properties. Considering the important part that properties play in modeling a configurable product, some care should be given to how properties are defined and used. This section outlines some useful tips and procedures to follow when defining and attaching properties.

Use Meaningful Property Names

Sometimes when developing a model, especially when under severe time constraints, the modeler is tempted to take shortcuts in order to speed the development process. One of the most common shortcuts is to create properties with short, and often vague or cryptic, names. This may speed the development of the model in the short term, but dramatically increases the amount of effort required to maintain the model. The modeler should always design their models so that it is immediately obvious what a given property represents. The more meaningful the name you give to a property, the easier it will be to debug and maintain the model now and in the future.

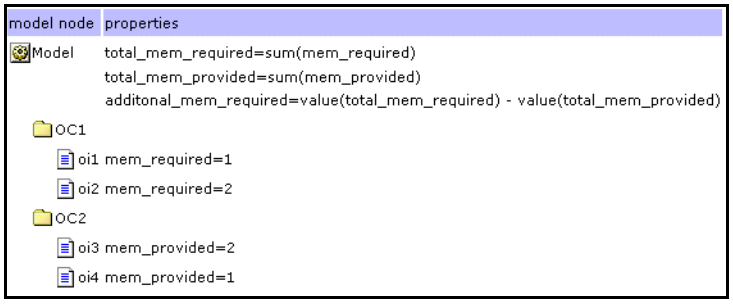
Consider the following example:



model node	properties
Model	tmr=sum(mr)
	tmp=sum(mp)
	amr=value(tmr)-value(tmp)
OC1	
oi1	mr=1
oi2	mr=2
OC2	
oi3	mp=2
oi4	mp=1

FIGURE 6. Using Cryptic Property Names

At first glance it may not be obvious what the properties assigned to this model are trying to accomplish. With a little more time spent creating meaningful names it becomes much easier to grasp the essence of all the properties and how they relate to one another.



model node	properties

Model	total_mem_required=sum(mem_required)
	total_mem_provided=sum(mem_provided)
	additonal_mem_required=value(total_mem_required) - value(total_mem_provided)

OC1	
oi1	mem_required=1
oi2	mem_required=2
OC2	
oi3	mem_provided=2
oi4	mem_provided=1

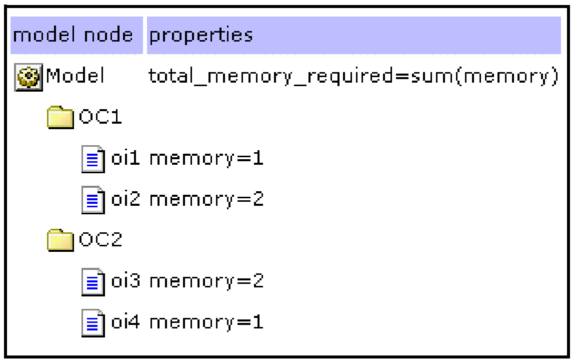
FIGURE 7. Using Meaningful Property Names

Do Not Use the Same Property to Mean Two Different Things

Often, in their haste to implement a particular feature, a modeler will reuse an existing property instead of creating a new property designed specifically for the problem at hand. This has two possible implications:

- The model may be harder to understand if the existing property name bears no relation to the problem at hand.
- Re-use of the property name may actually cause errors in the model if the re-use conflicts with the property's original use.

Let us revisit our example from the previous section. Suppose that our modeler created a property to store memory required and called it "memory" (see previous section for why that was a bad choice to begin with). Now when he is determines that he needs a property to store memory provided, he notices that he has a property called memory and decides to use it instead of creating a new property.



model node	properties
Model	total_memory_required=sum(memory)
OC1	
oi1	memory=1
oi2	memory=2
OC2	
oi3	memory=2
oi4	memory=1

FIGURE 8. Using One Property Name For Different Purposes

Now, at first glance, it looks like all option items require some amount of memory, instead of two items requiring memory and two providing it. Not only that, but our total_memory_required property will no longer have the correct value, since it now performs a sum of both memory required and memory provided. If modeled in this fashion, then the modeler will have to do extra work to separate out the specific instances of the properties he needs: such as using full or relative paths to the items containing the appropriate property instances. (See “Rules” on page 130 on why using paths to specific instances of properties can be a bad idea.)

Define Properties at the Appropriate Level in the Model Hierarchy

Properties may defined at any level in the model group hierarchy, from the root model group level to the individual model level. Where a property is defined determines which models can see and make use of the property. A little thought during the design of your models will speed model development and help prevent property clutter. Use the following guidelines to determine where a property should be defined:

If a property will only be used in a particular model, then define the property at the model level.

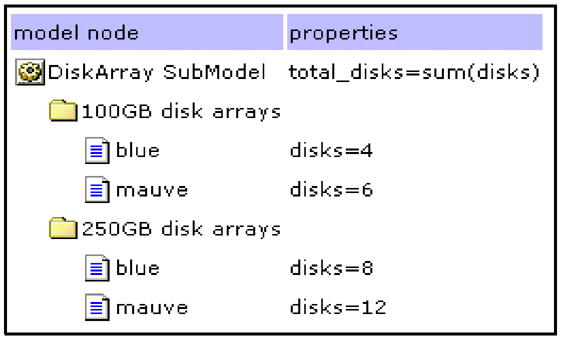
If a property will be used in more than one model within a particular model group, then define the property at that model group level.

If a property will be used in models that span model groups, then define the property in the first model groups that contains all of the model groups whose models will use the property.

As a last resort, define the property at the root model group node.

Using Multiple Properties with the Same Value

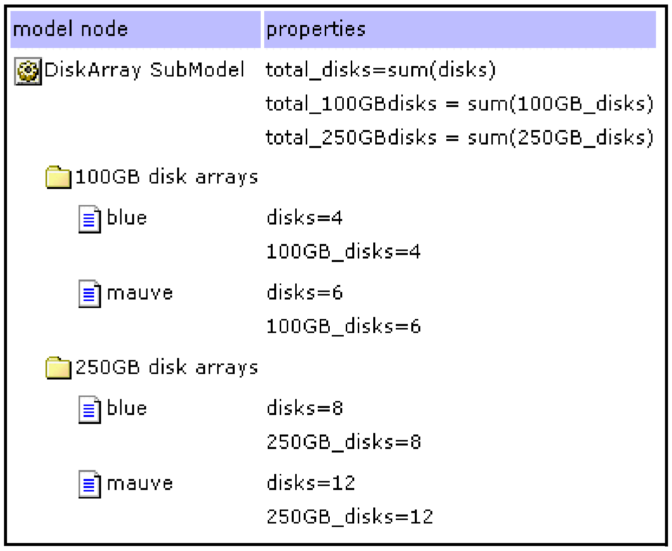
Multiple properties with the same value can sometimes make a model easier to build and maintain.This concept may be confusing at first and is best demonstrated by an example. Suppose that you are building a model that allows the user to choose from a selection of disks arrays. Each type of disk array has some number of disks associated with it. The user can choose multiple disk arrays of any type. One of the pieces of information that you need to calculate is the total number of disks that the user has selected (see below).



model node	properties
DiskArray SubModel	total_disks=sum(disks)
100GB disk arrays	
blue	disks=4
mauve	disks=6
250GB disk arrays	
blue	disks=8
mauve	disks=12

FIGURE 9. Using Multiple Property Names to Clarify Purpose

Now let us assume that you realize that you also need to know the number of 100GB disk arrays and the number of 250GB disk arrays. Instead of calculating these values by specifying item paths to the properties that we want, or writing rules that have to be attached at a particular point in the model, or re-working all the disk and total_disk properties, we can simply define a couple of new properties that have the same values as our old disk property (see below).



model node	properties

DiskArray SubModel	total_disks=sum(disks)
	total_100GBdisks = sum(100GB_disks)
	total_250GBdisks = sum(250GB_disks)

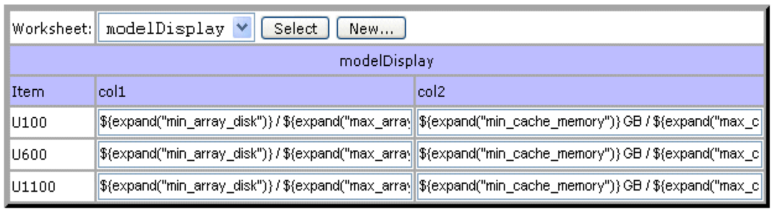
100GB disk arrays	
blue	disks=4 100GB_disks=4
mauve	disks=6 100GB_disks=6
250GB disk arrays	
blue	disks=8 250GB_disks=8
mauve	disks=12 250GB_disks=12

FIGURE 10. Using Additional Properties for Simple Calculations

Now, if we want the total disks, we can still get sum(disks). If we want the individual values, then we can get those as well: and all without specifying paths to individual properties or modifying the work that we had already done.

Use Worksheets to Simplify Property Assignment

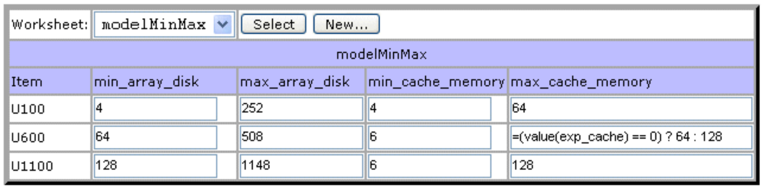
When developing a model, it is often necessary to assign the same set of properties to multiple option classes or option items. Worksheets are very useful in this case, since they allow you to rapidly set the values for a particular property on any number of option classes or items. this is especially true when using a formula to set the value of a property in multiple places. The modeler can simply copy and paste the formula onto all the items he wishes. An example of this is shown below. Here we have some display properties that are set for each item within a tabular display. We use a worksheet to allow us to easily cut and paste the formulas for col1 and col2 to each item in the option class.



modelDisplay		
Item	col1	col2
U100	${expand("min_array_disk")} / ${expand("max_array	${expand("min_cache_memory")} GB / ${expand("max_c
U600	${expand("min_array_disk")} / ${expand("max_array	${expand("min_cache_memory")} GB / ${expand("max_c
U1100	${expand("min_array_disk")} / ${expand("max_array	${expand("min_cache_memory")} GB / ${expand("max_c

FIGURE 11. Using Worksheets to Simplify Property Assignment

An added benefit of using worksheets is that can provide a concise picture of a section of the model. With a little thought and planning, a worksheet can provide an overview of a particular section of the model or a complete representation of the solution to a particular problem. Below is a different view of the same option class. In this case, we are interested in seeing all the min and max properties that are set for each of the option items.


Worksheet: modelMinMax Select New...

modelMinMax				
Item	min_array_disk	max_array_disk	min_cache_memory	max_cache_memory
U100	4	252	4	64
U600	64	508	6	=(value(exp_cache) == 0) ? 64 : 128
U1100	128	1148	6	128

FIGURE 12. Worksheet Showing Min and Max Properties For Each Option Item

Avoid Chaining Property Formulas

Properties attached to an item do not have any notion of sequence. By this we mean that, when using formulas to set property values, we cannot rely on any particular order of evaluation of the formulas. If property A contains a formula and property B contains a formula that relies on property A, then we have no guarantee that the rule created from formula B will fire after the rule created for formula A. In order to get around this issue, the modeler has two choices:

- Turn the first formula into a rule that fires before the second formula is evaluated. All rules generated from formulas have a priority of 50. By creating a rule for the first formula, and setting its priority to be less than 50, we ensure that the value of property A will be set before the value of property B is calculated.
- Turn on repeat rule firing. In this case the first phase of rule-firing will calculate the value for property A. The second pass of the rule-firing loop will calculate the value of property B based on the value of property A computed in the first pass. Note: massive amounts of chaining of formulas in this way may result in degradation of performance due to the number of passes through the rule-firing loop necessary to satisfy all the conditions. For this reason, we recommend the first alternative and advocate limiting formula chaining as much as possible.

Rules

Rules affect the efficiency and ease of maintenance of your model. This section describes considerations to keep in mind while writing rules.

Rule Firing Conditions

Rule conditions are created by applying boolean operations to relational expressions. A relational expression is the comparison of one function/property pair with another function/property pair using relational operations such as less than, equal to, greater than, in, not in, and so on. The result is either true or false. Boolean operators like AND and OR wrap sets of these relational expressions. The relational expressions are called *fragments*, as they are fragments of a rule. The left-hand-side of the relational operator is often abbreviated LHS, while RHS stands for right-hand-side.

Order Rule Fragments So That Rules Fire Only When Necessary

The evaluation of rule fragments determines when a rule fires, so the order in which fragments appear in a rule is important. The more quickly the model can determine whether a rule is true or false, the more efficient the model can be. And of course, the more quickly the model determines that a rule should not be fired, the sooner the model can continue to other processing. Placing rule fragments in order, from most likely to prevent the rule from firing to least likely to prevent the rule from firing, can improve performance.

Always test your rules to ensure that they fire only when appropriate. Knowing under what circumstances a rule's results will or will not be used is also important. For example, an expansion rule that always fires but will not pick something in the expansions section if the quantity formula results in zero, or if there are not any matches for the formula in the > and <= fields in the expansions section, is very inefficient.

Create General-Purpose Rules

Whenever possible, write rules that are as general as possible. For example, the following rule can be attached to any product to which the productType and handsetType properties are attached:

```
If  propval(productType) != value(selectProductType)
and propval(handsetType) != value(phonePreference)
    set _isVisible=0
```

This rule fires only for products where the productType property is attached AND does not match the selected product types AND if the selected phone preferences do not match the current product's preferences. A general rule such as this one can replace dozens of other specific rules such as the following specific ones:

```
If  propval(productType) == literal("handset")
and propval(handsetType) != literal("camera")
and value(phonePreference) == literal("camera")
    set _isVisible=0
If  propval(productType) == literal("handset")
and propval(handsetType) != literal("flip")
and value(phonePreference) == literal("flip")
    set _isVisible=0
```

…

Use Formulas Where Appropriate

In many circumstances, formulas can be used instead of rules. During modeling, formulas are maintained as attached properties that have as their value an expression that is evaluated at runtime. If any of the functions referenced in the expression cannot be evaluated, the formula acts like a rule that hasn't fired. If multipass rule firing is turned on, the formula will be reevaluated during each firing pass until rule firing ends or until the formula produces its result.

Use a formula rather than a rule when the only condition for requiring that you compute a result is that the function/properties used in the formula have values.

For example, suppose that you want to compute the turning radius for truck components such as axel and wheelbase to ensure that a user's choice of truck components makes sense. You might attach a formula to the relevant truck components to compute the turningRadius as follows:

```
turningRadius = value(axelTurnFactor) * value(wheelBaseTurnFactor)*
sum(turningElements)
```

This formula will fire when each of the value(axelTurnFactor), value(wheelBaseTurnFactor), and sum(turningElements) expressions all produce numeric results.

The equivalent rule is as follows:

```
if (value(axelTurnFactor) >= 0 or value(axelTurnFactor) < 0)
and (value(wheelBaseTurnFactor)>= 0 or value(wheelBaseTurnFactor)< 0)
and (sum(turningElements) >=0 or sum(turningElements) <0)
      turningRadius = value(axelTurnFactor) * value(wheelBaseTurnFactor)*
sum(turningElements)
```

The condition portion of the rule is quite long and seems to always evaluate to true. However, functions can return NULL if a property that they reference does not exist, so this rule is really checking that the result is non-NULL by evaluating whether a returned value is >= 0 or < 0.

Avoid Specifying Paths to Instances of Items or Properties

The LHS and RHS of a rule fragment consist of a function and a property name. The property name can contain both relative and absolute path information. However, specifying a property's path information in a rule fragment can result in the rule becoming inoperable if the path information or option classes change.

For example, the following rule references wheelSize and wellSize using fully specified path information. If the modeler ever needs to rename either the wheels or fender option classes, or wishes to reuse the rule in some other model, the rule may not operate correctly.

```
   If value(*.wheels.wheelSize) == literal("17in")
   and value(*.fender.wellSize) < literal(17)
       set _isVisible=0
```

Use path information only if you want to access one specific instance of a property, and then only if it isn't possible to make a new property type to hold this value. If you must reference a property's path name, it is often better to use relative pathnames rather than absolute pathnames.

Constraint Tables vs. Rules

This section explains the trade-off between using constraint tables to limit customer choices vs. using rules. Constraint tables limit a customer's choice of one or more option items based on the customer's choice of another option item. For example, the choice of an exterior color for a car might limit the choice of interior colors.

Constraint tables work best for simple validation, for example, an option item does or doesn't work with another option item. Simple constraint tables are easier to maintain than rules. However, large, complex constraint tables can be hard to maintain and can lead to performance issues.

Constraint tables are turned into rules internally.

Rules are best for expressing complex validation issues, and are more versatile than constraint tables. While both constraint tables and rules can display error messages, you can also create rules to set properties or make choices.

Modular Development

This section explains some of the techniques for simplifying model creation and maintenance. Selecting the appropriate technique may have a significant impact on model performance.

- Using Option Class Groups, Option Item Groups, and Sub-assemblies:

  This technique works well when a group of options is repeated in many different models.

  For example, suppose that every computer you sell includes a list of hard drives that the user can choose from. Creating Option Class Groups, Option Item Groups, and Sub-assemblies allows the modeler to create and maintain common information in one place, then use it in many places.

  One drawback is that this technique can lead to overly large models if a sub-assembly is included in the same model many times.

- Sub-model punch-in and punch-out:

  This technique is useful when a configuration contains a selection that is also configurable. You can use sub-model punch-in and punch-out to nest complex configurations within one overall selling model.

  One drawback is that all copies of the configured product will have the same configuration.

- Dynamic instantiation:

  This technique allows multiple instances of a configured product within a single model. Each instance can have a different configuration

Tools

Modeling can be a time consuming and tedious exercise, but in the end the correctness of the modeling and the scalability of the created solution are key to the success of the project. To aid in creating scalable and correct models, we have developed a collection of tools that can be used in various phases of development to guide the modeler. During development, the trace log and the model reporting tool can help the modeler determine which models to debug. Before pushing models into production, their scalability and stability can be tested using the load testing platform. Finally, during execution, the model cache status page can provide insights into the model's usage of the system, and the log analyzer can be used to make sense out of megabytes worth of log information.

Using the Trace Log

The trace log shows the execution of the rules engine. This is often, though not always, the most time consuming part of each request that the configurator makes to the server. The trace log is designed to provide the information necessary to debug rules that are misbehaving and to track the execution time of rules, so always start your debugging by reviewing the trace log.

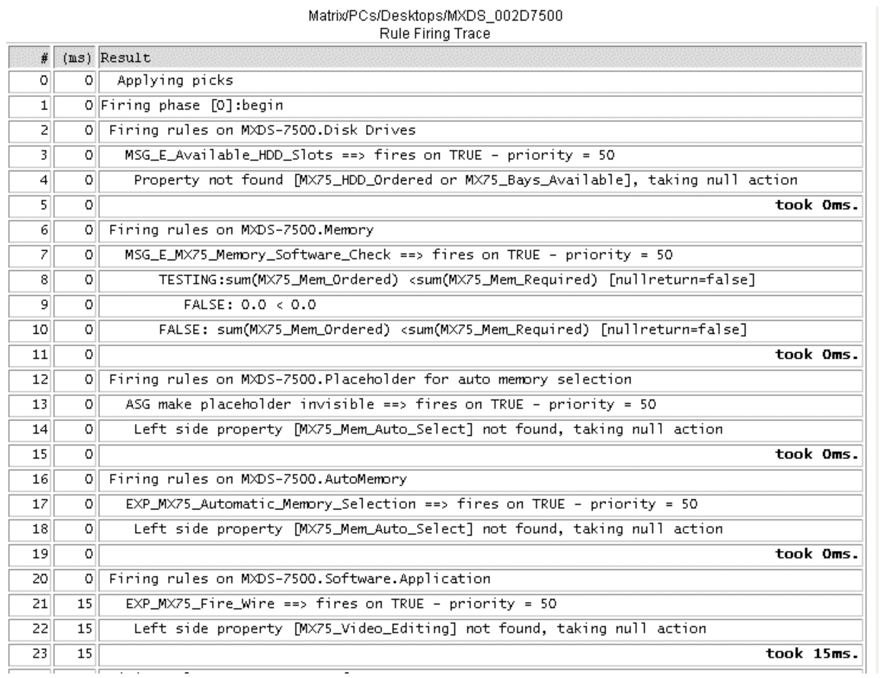
You create trace logs using the Visual Modeler. To do so:

1. Go to Model Group navigation and navigate to the model you wish to debug.
2. Select the model from the Models and Groups panel.
3. The model displays in the Model Preview tab.
4. Click the Test icon.
5. The model runs in a separate window.
6. Click Debug.

The trace log appears in a separate window.

The log consists of two sections. The first section is the rule firing trace and the second section is the property pool as it exists at the end of rule firing.

The following illustration shows a section of a sample rule firing trace.



Matrix/PCs/Desktops/MXDS_002D7500
Rule Firing Trace

#	(ms)	Result
0	0	Applying picks
1	0	Firing phase [0]:begin
2	0	Firing rules on MXDS-7500.Disk Drives
3	0	MSG_E_Available_HDD_Slots ==> fires on TRUE - priority = 50
4	0	Property not found [MX75_HDD_Ordered or MX75_Bays_Available], taking null action
5	0	took 0ms.
6	0	Firing rules on MXDS-7500.Memory
7	0	MSG_E_MX75_Memory_Software_Check ==> fires on TRUE - priority = 50
8	0	TESTING:sum(MX75_Mem_Ordered) <sum(MX75_Mem_Required) [nullreturn=false]
9	0	FALSE: 0.0 < 0.0
10	0	FALSE: sum(MX75_Mem_Ordered) <sum(MX75_Mem_Required) [nullreturn=false]
11	0	took 0ms.
12	0	Firing rules on MXDS-7500.Placeholder for auto memory selection
13	0	ASG make placeholder invisible ==> fires on TRUE - priority = 50
14	0	Left side property [MX75_Mem_Auto_Select] not found, taking null action
15	0	took 0ms.
16	0	Firing rules on MXDS-7500.AutoMemory
17	0	EXP_MX75_Automatic_Memory_Selection ==> fires on TRUE - priority = 50
18	0	Left side property [MX75_Mem_Auto_Select] not found, taking null action
19	0	took 0ms.
20	0	Firing rules on MXDS-7500.Software.Application
21	15	EXP_MX75_Fire_Wire ==> fires on TRUE - priority = 50
22	15	Left side property [MX75_Video_Editing] not found, taking null action
23	15	took 15ms.

FIGURE 13. Sample Rule Firing Trace

The rule firing trace has three columns:

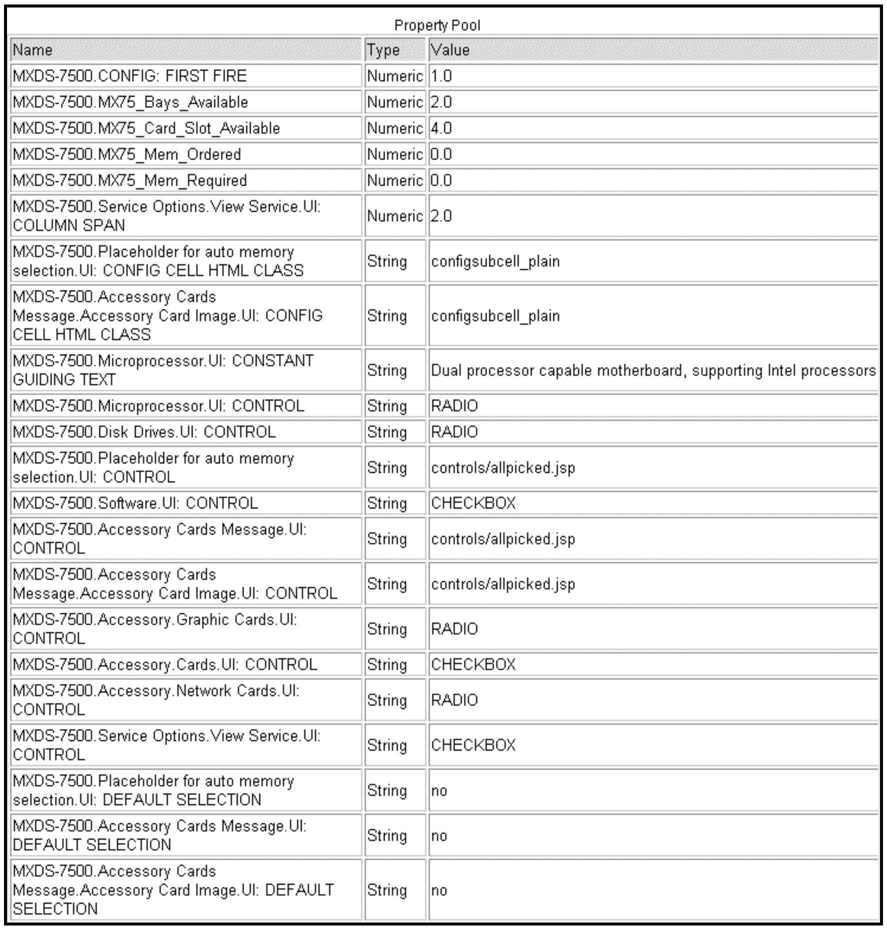
- A sequence number, useful for communicating with others about rule issues. It’s easy to tell someone, "See line 42 where it says Xxx?".
- Elapsed time. This logs how long it took from the time the log entry was made until the start of rule firing.
- The body of the trace log. This shows aspects of the rule firing, such as a condition being evaluated, an assignment occurring, the start of a rule or the conclusion of a rule, and so on.

The log shows the number of milliseconds needed to fire a rule after each rule firing entry. The total number of milliseconds needed to run the model is logged at the end of the rule firing trace.

The property pool trace also presents three columns:

- Name is the full path name to the item and the property on that item.
- Type is the property type for the named property, such as Numeric, List, or String.
- Value is the value of the property after the rule has fired.

The following illustration shows a section of a sample property pool trace.



Property Pool

Name	Type	Value
MXDS-7500.CONFIG: FIRST FIRE	Numeric	1.0
MXDS-7500.MX75_Bays_Available	Numeric	2.0
MXDS-7500.MX75_Card_Slot_Available	Numeric	4.0
MXDS-7500.MX75_Mem_Ordered	Numeric	0.0
MXDS-7500.MX75_Mem_Required	Numeric	0.0
MXDS-7500.Service Options.View Service.UI: COLUMN SPAN	Numeric	2.0
MXDS-7500.Placeholder for auto memory selection.UI: CONFIG CELL HTML CLASS	String	configsubcell_plain
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: CONFIG CELL HTML CLASS	String	configsubcell_plain
MXDS-7500.Microprocessor.UI: CONSTANT GUIDING TEXT	String	Dual processor capable motherboard, supporting Intel processors
MXDS-7500.Microprocessor.UI: CONTROL	String	RADIO
MXDS-7500.Disk Drives.UI: CONTROL	String	RADIO
MXDS-7500.Placeholder for auto memory selection.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Software.UI: CONTROL	String	CHECKBOX
MXDS-7500.Accessory Cards Message.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Accessory.Graphic Cards.UI: CONTROL	String	RADIO
MXDS-7500.Accessory.Cards.UI: CONTROL	String	CHECKBOX
MXDS-7500.Accessory.Network Cards.UI: CONTROL	String	RADIO
MXDS-7500.Service Options.View Service.UI: CONTROL	String	CHECKBOX
MXDS-7500.Placeholder for auto memory selection.UI: DEFAULT SELECTION	String	no
MXDS-7500.Accessory Cards Message.UI: DEFAULT SELECTION	String	no
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: DEFAULT SELECTION	String	no

FIGURE 14. Sample Property Pool Trace

Use this log "single user" to get a feel for how extensive the rules are per click. Check how long is it taking to fire the rules. If the answer is more than 100-200ms you may have scalability problems. If you do, use the trace log to figure out if any particular rules are performing badly.

Using the Model Reporting Tool

The model reporting tool can provide an overview of a model's size relative to other models. Use it to help make decisions about which models to test. You can track the test results over time so that you can determine the amount of change to the model.

Using Load Testing Tools

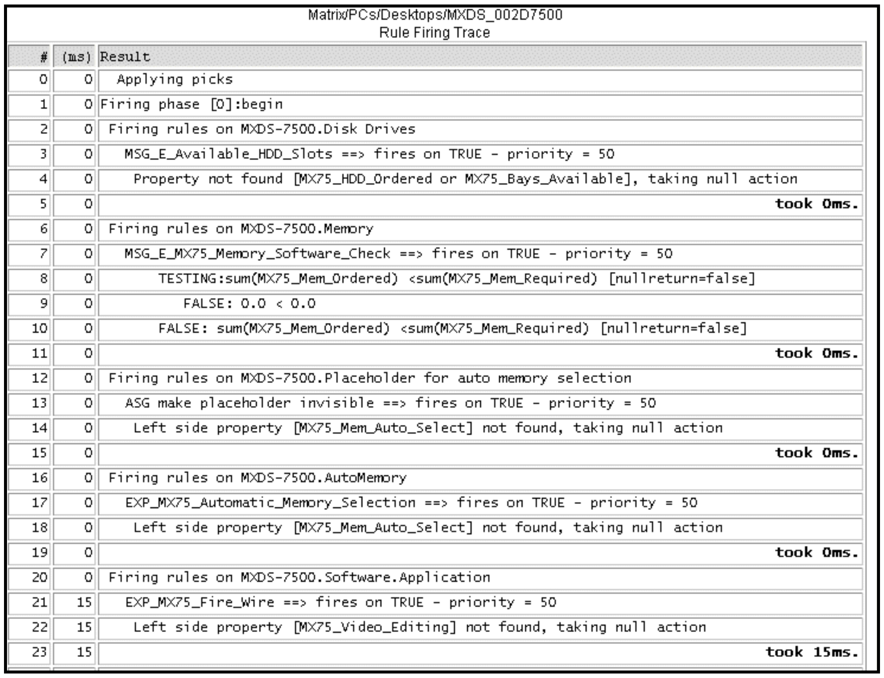
Load testing tools help you determine how your model will perform once deployed. Before using the load testing tools:

- Understand what is being tested.
- Isolate your test cases so that you know what the impact means (local vs. remote LAN testing, testing with and without clustering, with and without web fronting, and so on).

- Understand that as models change, so must any scripts that you use to perform testing and replay test scenarios.

For more information about load testing, see *Verifying Performance*.

  - If they are more global than the current model group, then define them at the lowest point in the model group tree that is an ancestor of a model where you wish to use the property.



Matrix/PCs/Desktops/MXDS_002D7500
Rule Firing Trace

```
 #  (ms) Result
 0    0    Applying picks
 1    0  Firing phase [0]:begin
 2    0   Firing rules on MXDS-7500.Disk Drives
 3    0     MSG_E_Available_HDD_Slots ==> fires on TRUE - priority = 50
 4    0      Property not found [MX75_HDD_Ordered or MX75_Bays_Available], taking null action
 5    0                                                                          took 0ms.
 6    0   Firing rules on MXDS-7500.Memory
 7    0     MSG_E_MX75_Memory_Software_Check ==> fires on TRUE - priority = 50
 8    0         TESTING:sum(MX75_Mem_Ordered) <sum(MX75_Mem_Required) [nullreturn=false]
 9    0            FALSE: 0.0 < 0.0
10    0         FALSE: sum(MX75_Mem_Ordered) <sum(MX75_Mem_Required) [nullreturn=false]
11    0                                                                          took 0ms.
12    0   Firing rules on MXDS-7500.Placeholder for auto memory selection
13    0     ASG make placeholder invisible ==> fires on TRUE - priority = 50
14    0      Left side property [MX75_Mem_Auto_Select] not found, taking null action
15    0                                                                          took 0ms.
16    0   Firing rules on MXDS-7500.AutoMemory
17    0     EXP_MX75_Automatic_Memory_Selection ==> fires on TRUE - priority = 50
18    0      Left side property [MX75_Mem_Auto_Select] not found, taking null action
19    0                                                                          took 0ms.
20    0   Firing rules on MXDS-7500.Software.Application
21   15     EXP_MX75_Fire_Wire ==> fires on TRUE - priority = 50
22   15      Left side property [MX75_Video_Editing] not found, taking null action
23   15                                                                         took 15ms.
```

FIGURE 15. Sample Rule Firing Trace

The rule firing trace has three columns:

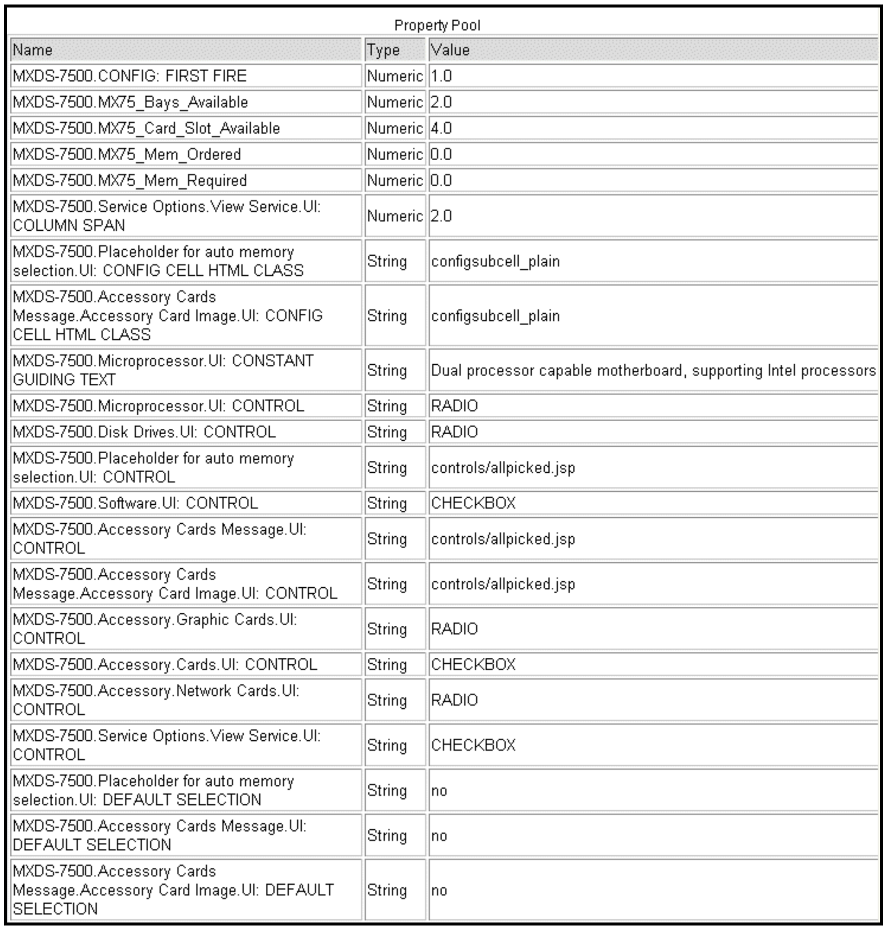
- A sequence number, useful for communicating with others about rule issues. It’s easy to tell someone, "See line 42 where it says Xxx?".
- Elapsed time. This logs how long it took from the time the log entry was made until the start of rule firing.
- The body of the trace log. This shows aspects of the rule firing, such as a condition being evaluated, an assignment occurring, the start of a rule or the conclusion of a rule, and so on.

The log shows the number of milliseconds needed to fire a rule after each rule firing entry. The total number of milliseconds needed to run the model is logged at the end of the rule firing trace.

The property pool trace also presents three columns:

- Name is the full path name to the item and the property on that item.
- Type is the property type for the named property, such as Numeric, List, or String.
- Value is the value of the property after the rule has fired.

The following illustration shows a section of a sample property pool trace.



Property Pool

Name	Type	Value
MXDS-7500.CONFIG: FIRST FIRE	Numeric	1.0
MXDS-7500.MX75_Bays_Available	Numeric	2.0
MXDS-7500.MX75_Card_Slot_Available	Numeric	4.0
MXDS-7500.MX75_Mem_Ordered	Numeric	0.0
MXDS-7500.MX75_Mem_Required	Numeric	0.0
MXDS-7500.Service Options.View Service.UI: COLUMN SPAN	Numeric	2.0
MXDS-7500.Placeholder for auto memory selection.UI: CONFIG CELL HTML CLASS	String	configsubcell_plain
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: CONFIG CELL HTML CLASS	String	configsubcell_plain
MXDS-7500.Microprocessor.UI: CONSTANT GUIDING TEXT	String	Dual processor capable motherboard, supporting Intel processors
MXDS-7500.Microprocessor.UI: CONTROL	String	RADIO
MXDS-7500.Disk Drives.UI: CONTROL	String	RADIO
MXDS-7500.Placeholder for auto memory selection.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Software.UI: CONTROL	String	CHECKBOX
MXDS-7500.Accessory Cards Message.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Accessory.Graphic Cards.UI: CONTROL	String	RADIO
MXDS-7500.Accessory.Cards.UI: CONTROL	String	CHECKBOX
MXDS-7500.Accessory.Network Cards.UI: CONTROL	String	RADIO
MXDS-7500.Service Options.View Service.UI: CONTROL	String	CHECKBOX
MXDS-7500.Placeholder for auto memory selection.UI: DEFAULT SELECTION	String	no
MXDS-7500.Accessory Cards Message.UI: DEFAULT SELECTION	String	no
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: DEFAULT SELECTION	String	no

FIGURE 16. Sample Property Pool Trace

Use this log "single user" to get a feel for how extensive the rules are per click. Check how long is it taking to fire the rules. If the answer is more than 100-200ms you may have scalability problems. If you do, use the trace log to figure out if any particular rules are performing badly.

Using the Model Reporting Tool

The model reporting tool can provide an overview of a model’s size relative to other models. Use it to help make decisions about which models to test. You can track the test results over time so that you can determine the amount of change to the model.

Using Load Testing Tools

Load testing tools help you determine how your model will perform once deployed. Before using the load testing tools:

- Understand what is being tested.
- Isolate your test cases so that you know what the impact means (local vs. remote LAN testing, testing with and without clustering, with and without web fronting, and so on).

- Understand that as models change, so must any scripts that you use to perform testing and replay test scenarios.

For more information about load testing, see *Verifying Performance*.

Cache Status

- cmd=configstatus shows the current contents of the cache

Performance

Rules

- Excessive paths to items:
- A rule that adds memory by:

```
totalMem = value(*.adapter.1.memory) + value(*.adapter.2.memory) +
value(*.adapter.3.memory) + value(*.adapter.4.memory)
```

- will perform much more slowly than:
- totalMem = sum(memory)
- If the memory property exists in other places for other uses so that sum(memory) would produce the wrong value, then introduce additional properties on the adapter items 1-4 called adapterMemory, and use:
- totalMem = sum(adapterMemory).
- This is much less maintenance effort that maintaining:

```
totalMem = value(*.adapter.1.memory) + value(*.adapter.2.memory) +
value(*.adapter.3.memory) + value(*.adapter.4.memory)
```

- Write rules to fire only when they are needed:
- A rule that assigns totalMem = sum(mem) only needs to fire if count(mem) > 0

Properties

- Define properties at the correct position in the model group hierarchy:
  - If they are local only to this model, then define them in the model.
  - If they may be used by other models within this model group, then define them in the current model group.
  - If they are more global than the current model group, then define them at the lowest point in the model group tree that is an ancestor of a model where you wish to use the property.

Data Validation

The Sterling Web application provides the Data Validation functionality for validating and sanitizing request inputs and outputs. You can use the Data Validation functionality to, allow only the explicitly defined characteristics in the input and output requests, and drop all the other data. You can define your own validation rules for validating different request parameters. You can also encode the data before sending it back to the user interface.

Data validation or sanitization can be performed for various kinds of inputs such as parameter name, parameter value, cookie name, cookie value, and so on. Sterling Web supports regular expression based validation.

Input Validator

The Selling and Fulfilment Foundation Input Validator finds all the validation rules that are registered for a particular input, and invokes the validation. Sterling Web validator is called by a request wrapper to validate request inputs.

Validation Rule

A validation rule performs validation and sanitization of the input. A validation rule contains a property as input identifier for which validation has to happen. A validation is invoked whenever the corresponding input request is accessed. A validation rule must specify the name of the input, it has to validate. For example, if you want to validate the value of a particular parameter, the validation rule must specify the name of that parameter. Multiple inputs with the same name can exist. All the validation rules must be registered with the Input Validator in order to validate the corresponding input.

Note: No validation rules are defined for a given input. The default validation rules specified earlier will be used to validate the input.

You can define the following types of validation rules:

- Regular Expression-Based Validation Rule—This type of validation rule is designed to perform regular expression-based validations. This validation rule type supports multiple whitelist and blacklist regular expressions.
- Java-Based Validation Rule—This type of validation rule is designed to perform Java-based validation and sanitization of inputs. This validation rule type validates an input and then calls the getValidInput() method of the implementation class.

Disabling Data Validation

By default, data validation that is enabled on all input requests will be validated against the registered validation rules. You can disable the data validation on input requests by adding a context parameter in the module configuration file. To disable data validation, in the `web.xml` file located in your `EARFILE/WARFILE/WEB-INF` folder add an entry for the context-param element as follows:

```
<context-param>
```

```
    <param-name>scui-suppress-request-validation</param-name>
    <param-value>TRUE</param-value>
</context-param>
```

Bypassing Data Validation for an URI

By default, data validation is enabled on all input requests. You can, however, bypass data validation on input requests for some specific universal resource indicators (URIs) by adding bypass URI as context parameters in the module configuration file.

To bypass data validation, in the `web.xml` file located in your `EARFILE/WARFILE/WEB-INF` folder add an entry for the config-param element for each such URI, for example:

```
<context-param>
    <param-name>request.validation.bypass.uri.1</param-name>
    <param-value>/console/login.jsp</param-value>
</context-param>
<context-param>
    <param-name>request.validation.bypass.uri.2</param-name>
    <param-value>/console/start.jsp</param-value>
</context-param>
<context-param>
    <param-name>request.validation.bypass.uri.endswith.1</param-name>
    <param-value>.js</param-value>
</context-param>
<context-param>
    <param-name> request.validation.bypass.uri.regex.1</param-name>
    <param-value>^.*test.jsp$</param-value>
</context-param>
```

These context parameters can have names starting with request.validation.bypass.uri, or request.validation.bypass.uri.endswith, or request.validation.bypass.uri.regex, as described in the following list. You can define multiple entries for these context parameters.

- request.validation.bypass.uri—Any request with an URI that is the same as the value specified in the param-value element of the context parameter will be bypassed and not validated.
- request.validation.bypass.uri.endswith—Any request with an URI that ends with the value specified in the param-value element of the context parameter will be bypassed and not validated.
- request.validation.bypass.uri.regex—Any URI request that matches the regular expression, as specified in the param-value element of the context parameter, will be bypassed and not validated.

Implementing Data Validation

To implement data validation, perform the following tasks:

- Create a Validation Rule
- Register a Validation Rule
- Use an URI-Based Validation Rule
- Defining an Adapter to Find Validation Rules
- Delete the Registered Validation Rules

Create a Validation Rule

A validation rule performs validation and sanitization of the input. A validation rule contains a property as input identifier for which validation has to be performed. Validation is performed whenever a specified input is accessed.

To create a validation rule, implement the ISCValidationRule interface and implement the following interface methods:

- getName()—Returns the name of the input for which this validation rule should be applied.
- isValid(String context, String input)—should do the actual validation
- getSafe(String context, String input)—should return the safe output after removing all the unwanted characters from the input.
- getTypeName()

You can use _global_ and _default_ as names for the global and default rules respectively.

Sterling Web provides two implementation classes of the ISCValidationRule interface. These implementation classes validate input based on regular expressions or Java calls. You can use these implementation classes based on your requirement.

SCRegexValidationRule Implementation Class

This implementation class of the ISCValidationRule interface helps in validating an inputs based on regular expression whitelists and blacklists. You can use this class for creating validation rules that are based on regular expressions.

To create a validation rule using this implementation class:

1. Create an instance of the SCRegexValidationRule class. For example:

   ```
   ISCValidationRule SCRegexValidationRule newValRule = new
   SCRegexValidationRule();
   ```

2. Set the name of the validation rule as follows:

   - If you are defining the validation rule for a parameter value, set the name of the validation rule as parameter name. For example:
   - newValRule.setName("<parameter_name>");

- If you are defining the validation rule for a parameter name, set the name of the validation rule as follows:

  newValRule.setName(newValRule.RULE_NAME_GLOBAL);

Note: You can use _global_ and _default_ as names for the global and default rules respectively.

3. Add the whitelist pattern of regular expression against which you want to validate the parameter name. For example:

   newValRule.addWhitelistPattern("^[a-zA-Z0-9.\\-\\/+=_ :]*$");
4. Set the AllowNull value. For example:

   newValRule.setAllowNull(true);
5. Register this new validation rule with the Input Validator. For more information about registering validation rules, refer to the topic "Register a Validation Rule".

SCJavaValidationRule Implementation Class

This abstract class of the ISCValidationRule interface helps in validating an input based on the Java call. You can implement this abstract class for creating a validation rule that requires a Java call to validate inputs.

To create a validation rule using this abstract class, implement this abstract class and the getValidInput() and getSafe() methods of this abstract class.

Register a Validation Rule

In order to be able to validate inputs, you must register the validation rules that you created in the Input Validator.

Note: Global rules are applied on all inputs for an inputType. Default rules are applied on an inputType if no other rules either specific or global are found for an inputType. If invalid input is found, either an SCValidationException is thrown, or the exception is added to the passed SCValidationErrorList. For more information about exception handling during data validation, refer to the topic " Exception Handling".

To register a validation rule:

1. Get the Application ID:
2. If you are using an application that is based on the Web UI framework, use for the getApplicationId() method of the SCUIUtils class. For example:

   String appId = SCUIUtils.getApplicationId(config.getServletConext());

   If you are using an application that is based on the HTML UI framework, use the getConsoleApplicationId() method of the YFCContextParams class. For example:

   String appId = YFCContextParams.getInstance(config).getConsoleApplicationId()

Note: Ensure that the same application ID is defined as context parameter in the `web.xml` file located in your `EARFILE/WARFILE/WEB-INF` folder. For example:

```
<context-param>   <param-name>console-application-id</param-name>
  <param-value><Application ID></param-value> </context-param>
```

3. Get an instance of the SCValidator using the getInstanceFor() method with the Application ID that you stored in Step 1. For example:

   SCValidator validator = SCValidator.getInstanceFor(appId);

4. Register the validation rule with the SCValidator instance that you created in Step 3.

   - If you are defining the validation rule for a parameter value, use the INPUT_TYPE_PARAMETER_VALUE constant. For example:

     validator.addRule(newValRule, SCUIWebValidationConstants.INPUT_TYPE_PARAMETER_VALUE);

   - If you are registering the validation rule for a parameter name, use the INPUT_TYPE_PARAMETER_NAME constant. For example:

     validator.addRule(newValRule, SCUIWebValidationConstants.INPUT_TYPE_PARAMETER_NAME);

Note: When you register the validation rules for inputs, the following inputTypes constants must be used:

- INPUT_TYPE_PARAMETER_VALUE
- INPUT_TYPE_PARAMETER_NAME
- INPUT_TYPE_COOKIE_VALUE
- INPUT_TYPE_COOKIE_NAME
- INPUT_TYPE_HEADER_VALUE
- INPUT_TYPE_HEADER_NAME
- INPUT_TYPE_SCHEME
- INPUT_TYPE_SERVER_NAME
- INPUT_TYPE_CONTEXT_PATH
- INPUT_TYPE_PATH
- NPUT_TYPE_QUERY_STRING
- INPUT_TYPE_URI
- INPUT_TYPE_URL
- INPUT_TYPE_JSESSIONID
- INPUT_TYPE_SERVLET_PATH

Registering a Validation Rule for the Parameter Value Input Using datatypes.xml

This is an optional task. You can also register a validation rule using the datatypes file. This method of registering a validation rule can only be used for parameter value inputs. The datatype for a parameter is deduced using the datatypes map, and the parameter value is validated by using the validation rules registered against that datatype.

Applications based on the HTML UI framework can register a regular expression-based or Java-based validation rule in the `datatypes.xml` file in the following manner:

```
<DataType Name="Address" Size="70" Type="NVARCHAR">
        <UIType Size="30" UITableSize="30"/>
        <Validation>
                <Regex JavaPattern="<pattern>" JSPattern="<pattern>"/>
                <Impl JavaClass="com.sterlingcommerce.test.MyRuleClass"
                      JSFunctionName="myJavascriptFunction"/>
        </Validation>
</DataType>
```

By default, for a <Regex> element, the maximum size of the validation rule is set to the size of the datatype. You can override the maximum size of the validation rule by using the `MaxLength` attribute. Also, you can set the minimum size of the validation rule by using the `MinLength` attribute. For example:

```
<DataType Size="5" Name="Foo" Type="Bar">
  <Validation>
    <Regex MaxLength="200" MinLength="3" JavaPattern="^[a-zA-Z0-9.,!\-/+=_
:]*$"            JSPattern="^[a-zA-Z0-9.,!\-/+=_ :]*$"/>
  </Validation>
</DataType>
```

Note: A Java-based validation rule class must have a fully qualified class name, and this class must implement the ISCValidationRule interface.

In the `datatypes.xml` file, you can also define Javascript patterns and functions to validate the input on the client itself. These client side validations will be fetched on the client and all the corresponding inputs will be validated against these client side validations.

Use an URI-Based Validation Rule

An URI-based validation rule can be used to suppress certain rules for certain input requests. To use an URI-based validation rule, perform the following tasks:

1. Create a validation rule. For more information about creating a validation rule, refer to the topic "Create a Validation Rule".
2. Create an instance of the ISCValidationRuleKey using the SCUIURIContextValidationRuleKey(String uri, String name) class. For example:

   SCUIURIContextValidationRuleKey key = new SCUIURIContextValidationRuleKey(uri, name);

   Here, URI refers to the input path on which the rule should be applied. The URI should not contain the context path and should start with "/". name is the name of the input that has to be validated. You can use ISCValidationRule.RULE_NAME_GLOBAL and ISCValidationRule.RULE_NAME_DEFAULT as names to make a rule either a global rule or a default rule respectively.
3. Register the validation rule that you created along with the validation rule key. For example:

   SCValidator.getInstanceFor(SCUIWebValidationUtils.getApplicationID(servletContext)).addRule(key, rule, SCUIWebValidationConstants.INPUT_TYPE_PARAMETER_VALUE);

Defining an Adapter to Find Validation Rules

This is an optional task. You can define an adapter to find the validation rules that will be used to validate the parameter values. This adapter class must implement the ISCUIInputValidationAdapter interface, and must be registered with the application as a context parameter. For example:

```
<context-param>
    <param-name>scui-param-value-validation-adapter</param-name>
    <param-value>test.MyParamValueValidationAdapter</param-value>
</context-param>
```

You must implement the getValidationRules() method of the ISCUIInputValidationAdapter interface and pass the parameter name in the name argument.

When validating a parameter value, the system will call the registered adapter to find the rules against which the parameter value should be validated. The getValidationRules() method can either return all the rules registered for the passed parameter name, or have some logic to find other rules too. If no adapter is registered, the system will use all the rules registered for the given parameter name, along with the global rules or the default rules, to validate the parameter value.

Delete the Registered Validation Rules

You can delete the registered validation rules by calling any of the following methods:

- `removeDefaultRules(String inputType)`
- `removeGlobalRules(String inputType)`
- `removeRules(String name, String inputType)`
- `removeRules(ISCValidationRuleKey name, String inputType)`

Exception Handling

While validating a request, if an invalid input is found, an SCUIRequestValidationException is thrown. You can override this default behavior by adding the scui-suppress-validation-exception context parameter with the value as TRUE in the `web.xml` file located in your `EARFILE/WARFILE/WEB-INF` folder. For example:

```
<context-param>
    <param-name>scui-suppress-validation-exception</param-name>
    <param-value>TRUE</param-value>
</context-param>
```

When you set this parameter's value as TRUE, all the validation exceptions are added to a list, the list can be accessed by running:

ArrayList< SCUIRequestValidationException>
SCUIWebValidationUtils.getValidaionErrorList(HttpServletRequest request)

You can also define a global exception handler. If any validation exception has not been detected, and it goes back to the SCUISafeRequestFilter, the request will be sent to the corresponding global error handler servlet container.

This global exception handler and the request method can be defined as context parameters in the `web.xml` file, located in your `EARFILE/WARFILE/WEB-INF` folder. For example:

```
<context-param>
    <param-name>scui-global-validation-exception-handler-path</param-name>
    <param-value><path_to_global_exception_handler></param-value>
</context-param>
<context-param>
    <param-name>scui-global-validation-exception-handler-method</param-name>
    <param-value>FORWARD|INCLUDE|REDIRECT</param-value>
</context-param>
```

For applications that are based on the Web UI framework, Sterling Web provides `/jsps/datavalidationerror.jsp` as the default exception handler.

For applications that are based on the HTML UI framework, Sterling Web provides `/common/datavalidationerror.jsp` as the default exception handler.

The Web UI framework has also added a Struts action result, "DATAVALIDATIONERROR", which will be returned to the exception handler if the request is invalid. You can define this result type and the corresponding path, say, `/jsps/datavalidationerror.jsp`, for the Struts actions.

By default, the global exception handler method is set to FORWARD.

Default Validation Rules in Sterling Web

By default, Sterling Web provides and registers validation rules for specific parameters and rules pertaining to requests, based on specific inputs.

To validate the request inputs (such as parameter value, parameter name, etc.), the input validator uses following regular expressions:

- HTTPScheme=^(http|https)$
- HTTPServerName=^[a-zA-Z0-9_.\\-]*$
- HTTPParameterName=^[a-zA-Z0-9_\\-\\.]*$
- HTTPParameterValue=^[a-zA-Z0-9.!\\-\\/+=_]*$
- HTTPCookieName=^[a-zA-Z0-9\\-_\\.]*$
- HTTPCookieValue=^[a-zA-Z0-9.!\\-\\/+=_:]*$
- HTTPHeaderName=^[a-zA-Z0-9\\-_]*$
- HTTPHeaderValue=^[a-zA-Z0-9()\\-=\\*\\.\\?;,+\\/:&_\"]*$
- HTTPContextPath=^[a-zA-Z0-9.\\-_/]*$
- HTTPPath=^[a-zA-Z0-9.\\-_\\/!]*$
- HTTPURI=^[a-zA-Z0-9()\\-=\\*\\.\\?;,+\\/:&_ !\\$]*$
- HTTPURL=^.*$
- HTTPJSESSIONID=^[a-zA-Z0-9!_:\\-]*$
- HTTPQueryString=The HTTPQueryString will be validated based on the individual parameter names and values.

These validation rules are invoked for all the inputs of the same kind. For example, all HTTP Header names are validated against "HTTPHeaderName" regular expression.

Sterling Web also overrides the above validation rules for specific parameters and specific input-types by providing entries in the following property files:

- `swc_ParamValue_ValidationRules.properties`: This file is located in the `EARFILE/swc.jar/com/sterlingcommerce/webchannel/core/security/xss` folder. It contains validation rules that are specific to the request-input HTTPParameterValue for specific request parameter.
- `swc_InputType_ValidationRules.properties`: This file is located in the `EARFILE/swc.jar/com/sterlingcommerce/webchannel/core/security/xss/` folder. It contains validation rules that are specific to any of the request inputs.

Data Validation Failure

Data validation fails may fail if a user provides an invalid input. In such a scenario, Sterling Web enables a user to identify it by the way of displaying appropriate error message.

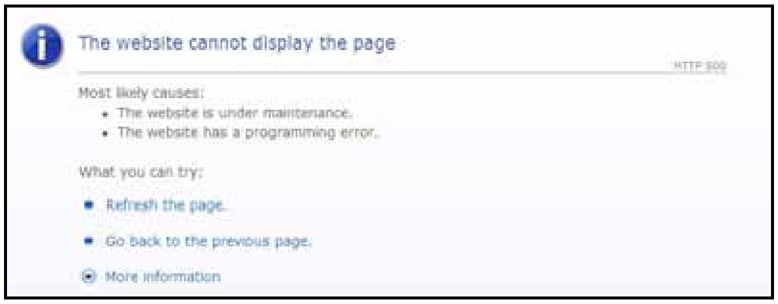
For example, if a user enters <script> as one of the basic search criteria in the Sterling Web Home page, the basic search field is validated. In such a scenario, the input validation fails because <script> is not a valid input. The details pertaining to validation failure are logged in the server log file. A user can view the server log file to understand the cause of validation failure, and verify whether it was an authentic validation failure. If required, a user can create validation rules based on the user's requirements.

Following is an example of an error trace generated in the server log:

```
3415041 [qtp0-5] WARN  DataValidationLogger  - SECURITY-FAILURE - Input does not
conform to pattern: context=HTTP parameter value (Name:searchTerm),
pattern=^[a-zA-Z0-9\:.\-\/\+=_\@\x11\x12\x13\x14\=\&\!\?;\"\(\)\*\'\[\] ]*$,
Input=<script/>

    ValidationException @
com.sterlingcommerce.security.dv.SCValidationException.<init>(SCValidationExce
ption.java:76)
```

The following figure shows an example of an input validation failure that is displayed in the Internet Explorer 7 and the Internet Explorer 8 browser:



The website cannot display the page

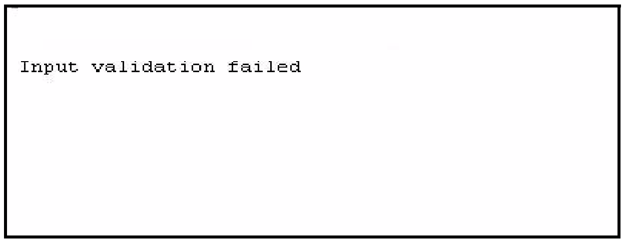
HTTP 500

Most likely causes:
- The website is under maintenance.
- The website has a programming error.

What you can try:
- Refresh the page.
- Go back to the previous page.
- More information

This is a minimal response page with the 500 response status and Multipurpose Internet Mail Extensions (MIME) type set to either text or plain. Because the response status is sent as 500 and the MIME type is set as either text or plain, some versions of the Internet Explorer browser may replace this error page with the browser’s custom error page. This is because of the preferences that have been set using **Internet Options** > **Advanced** tab to display user friendly HTTP error messages.

The following figure shows an example of an input validation failure that is displayed in the Mozilla Firefox 3.5 browser:



```
Input validation failed
```

Note: By default, Sterling Web provides validation rules that are specific to `en_US` locale. To support validation rules that are specific to a different locale, the corresponding user must customize the validation rules. For more information about customizing validation rules, refer to the *Sterling Web: Customization Guide*.

User Interface Validations

Sterling Web requires user interface-level validations for various pages in the user interface (UI). This requirement is met by Sterling Web's validation framework, which is based on the Apache Struts validation framework. Sterling Web also provides a hook for Ajax validation support. Sterling Web's Ajax validation framework provides a handle to enable Struts to perform all types of UI-level validation. Using either the common validators or the custom validators, validations can be performed across the pages in the Sterling Web UI.

A user can perform the following validations using Ajax validation support:

- Data type validation for all the fields that are in the form of a JSP.
- Action-specific validation that is specific to a Struts action or a page. For example, the following validations are specific to a page, and are performed using the Ajax validation framework:
  - Mandatory fields
  - Check box selection
  - Conditional fields
  - Combination of fields

Struts Validation Framework

The Struts validation framework works on the following tasks:

- Defining a Struts UI theme
- Validate attribute of a Struts form in a page
- Defining valdiator definitions for the form fields. These fields are defined in the corresponding validation XML pertaining to the Struts action of the form.

By default, wctheme, which is the Struts UI theme is associated with Sterling Web. For more information about defining custom themes, refer to the *Selling and Fulfillment Foundation: Customizing the Web UI Framework Guide.*

Consider the Add User page in the Sterling Web application. The JSP page has a form definition with saveNewUserInfo as the corresponding Struts action, and SaveUserInfoAction as the corresponding action class. The value of the `validate` attribute of the form must be set to true. The corresponding validation XML file is located in the same folder as the action class for implementing UI validation of the form in the Add User page. The validation XML filename is `SaveUserInfoAction-saveNewUserInfo-validation.xml`. The validation XML can be named either in `ActionClassname-ActionName-validation.xml` format or in the `Actionclassname-validation.xml` format.

The validation XML file contains the definitions of field names and the corresponding field validators. For example, in the following sample validation XML, the required validator must be used for validating the title form field:

```
<validators>

<field name="title">
```

```
<field-validator type="required">
<message>Error</message>
</field-validator>
</field>
</validators>
```

Note: The _required_ validator is registered in the `validators.xml` file.

Ajax Validation Framework

Sterling Web adds swcajax validator type as a custom Struts validator. The implementation class for this validator is com.sterlingcommerce.webchannel.core.validators.SWCAjaxServerSideValidator. In the validation XML, if at least one of the validator for a field is defined as swcajax, the Ajax validation framework is invoked. The Ajax validation framework performs the following tasks:

- Validates the values of all the parameters in the form with the corresponding data type.
- Invokes the action-specific validation class to perform action-specific validations. The name of this class is defined in the Struts action definition as the validationClass parameter.

  Ajax validation is performed by making an Ajax call to the ajaxValidateJson action. The implementation class for this action is com/sterlingcommerce/webchannel/core/validators/AjaxValidatedJson.java. This action class returns a json object that contains the errors corresponding to the form fields that are displayed when validation fails.This action class internally invokes an action-specific validation class to perform the corresponding action-specific validation.

To perform validations using the Ajax Valdiation framework, a user must adhere to the following specifications:

- A user must use the Struts tag must be used for forms and form fields.
- A user must use only wctheme for the forms and form fields.Simple theme should not be used for the forms and form fields.
- A form must have a `validate` attribute set as true.
- The following parameters must be passed as hidden parameters in a form.
  - #action.name (The name of the corresponding action)
  - #action.namespace (The namespace of the corresponding action)
- At least one of the form fields must have swcajax validator type as one of the validators in the validation XML.
- For action-specific validation, a new parameter, validationClass, must be added to the Struts action definition with the exact path to the class pertaining to the action-specific validation class.
- Action-specific validation class should extend the abstract class, AbstractWCFormFieldValidation, and implement the validate method for this class.
- For the form field data type validation, corresponding binding XMLs are used. Therefore, all the fields of the form must have an entry in the binding file in one of the following formats:
  - key=formFieldName

- key=#parameters.formFieldName

- If the binding XML is not present for any action, the form fields that require validation must have a corresponding hidden parameter in the form with the following naming convention:

  formFieldName.type = ApiAttributeName
- A form must be submitted only through the `s:submit` tag or the `<input type="submit"...>` tag.

The following types of validations are supported by the Ajax validation framework:

- Data type validations of form fields: This validation is supported by Selling and Fulfillment Foundation. For more information about data type files, refer to the S*elling and Fulfilment Foundation: Extending the Database Guide*.
- Length validation: This validation is performed to verify the permissible length of the length fields based on the data type in the form.
- Numeric validation: This validation is performed to verify the number format of the numeric fields in the form, for example, the **Quantity** field in the Cart Details page.
- E-mail format validation: The validation is done based on the swc.email.validator.regex property. This property is registered as one of the system properties. The value of this property contains the regex value, based on which, the e-mail values must be validated. By default, the value of this property provided by Sterling Web is ^[a-zA-Z]{1}[\\w-\\.]+@[\\w-\\d]+\\.\\w+$.
- URL format validation.
- Phone number format validation: The validation is performed by the class registered as the swc.validator.phone system property. The default implementation class for this property is com.sterlingcommerce.webchannel.core.validators.WCDefaultPhoneNumberFormatter. This class implements the IWCPhoneNumberFormatter interface. The validation logic in this default implementation class is based on the phone numbers based in US format, that is, +xxx (yyy) zzz-zzzz, where xxx is the Country code, yyy is the Area or City code, and zzz-zzzz is the Local number.
- Credit card number validation.
- Date format validation: This validation is performed to verify the date format of the date fields based on a user's locale.

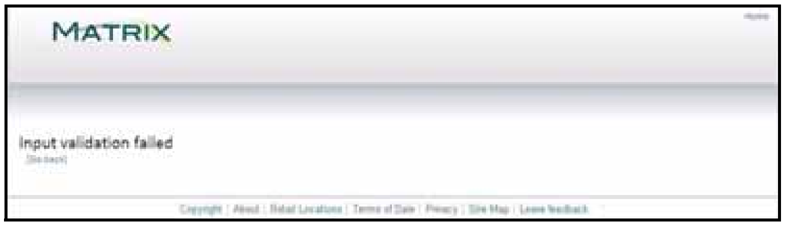
The action-specific validations can implement the following types of validations:

- Mandatory fields
- Check box selection
- Conditional validation of fields
- Combination of fields
- Any other kind of validation as required by the corresponding page

Sterling Web Validation Interceptor Layer

Sterling Web's validation framework is based on a JavaScript function on the client side. This JavaScript function is based on the validation XML and is associated with the onsubmit event of a JSP form. Because the execution of this function is carried out at the client side, validation can be avoided by either turning off the JavaScript or by modifying the function. To avoid such a scenario, a server-side Validation Interceptor layer is implemented. This layer performs the validation again before submitting to the Struts action. This ensures that all the UI-level validations are performed. For the layer to perfrom validation again, the wcServerSideValidationInterceptor is added in the default interceptor stack of Struts. The implementation class for this interceptor is com.sterlingcommerce.webchannel.core.validators.WCServerSideValidationInterceptor. All the validation errors captured in the framework are displayed in the UI adjacent to the corresponding form fields. However, if any validation error is captured in the Sterling Web Validation Interceptor layer, a generic input validation error page is displayed. All the validation errors and the corresponding error messages are logged in the application server log by the Validation Interceptor.

The following diagram displays an input ValidationInterceptor failure:



Input validation failed

[Go back]

MATRIX
Home
Copyright | About | Retail Locations | Terms of Sale | Privacy | Site Map | Leave feedback

Securing Data Paths

Sterling Web not only supports HTTP and HTTPS, but can also be configured to switch between HTTP and HTTPS. This enables a user to take care of performance and security concerns at the same time. .

HTTP and HTTPS interaction and switching

When a user visits a web site, all the interactions with the corresponding server are carried out using HTTP. If a user either uses an HTTPS URL to come to a Web site or modifies the HTTP protocol manually, all the interactions will be carried out using HTTPS from that instance onwards. After the user logs in, all the interactions remain in HTTPS throughout the session.

If the SSL switching is enabled, a user is switched to HTTP when the user logs out by clicking the Log Out hyperlink. This user is either a guest user or a remembered user, and all the interactions pertaining to this user in future will be carried out using HTTP. If the SSL switching is disabled, the original browsing setting is retained.

Sterling Web- specific cookies such as Remembered User, Active Cart, and Current Storefront are stored in the insecure mode to enable the sharing of these cookies between HTTP and HTTPS interactions. This type of storing is done only when SSL switching is enabled.

If SSL switching is enabled, a user is switched to HTTPS when the user performs any of the following tasks:

- Clicks the Sign-On hyperlink.
- Clicks a link that requires authorization.
- Session expires and the user logs in again.

Enable switching between HTTP and HTTPS

To enable switching between HTTP and HTTPS certain properties in the `yfs.properties_ysc_ext.in` file should be modified. This file can be overridden by `customer_overrides.properties` file. For more information about overriding properties, refer to the *Selling and Fulfillment Foundation: Properties Guide*.

Perform the following configurations in the `yfs.properties_swc_ext.in` properties file:

- Set the swc.ssl-switching.enabled property to Y.
- Set the swc.url.http.port property to configure the port number for HTTP.
- Set the swc.url.https.port property to configure the port number for HTTPS.
- Set the swc.url.servername property to override the servername property from the URL of the incoming request.

Index