ORACLE®

ATG WEB COMMERCE

Version 10.0.2

Commerce Reference Store Overview

Oracle ATG
One Main Street
Cambridge, MA 02142
USA

ATG Commerce Reference Store Overview

Document Version

CRS10.0.2 CRSOVERVIEWv1 4/15/2011

Copyright

Contents

1 Introduction

ATG Commerce Reference Store (CRS) is a group of related online stores built with ATG Commerce. CRS provides implementations of many features included with ATG Commerce, as well as customizations and extensions of ATG Commerce features. It is designed to serve as both a demonstration of many of the capabilities of ATG Commerce and as a framework to help ATG Commerce customers to build their own stores more quickly.

This chapter introduces ATG Commerce Reference Store and the sample storefronts that are included with it. It includes these sections:

Getting Started

CRS provides code, configuration, and data for simulating a complete working ATG Commerce application and supporting environment. CRS includes support for:

- A publishing server for managing and deploying site assets
- A production server that runs the site storefronts
- Catalog search facilities using ATG Search
- Internationalization, including a German related regional store, and the ability to view sites in German, Spanish, and English
- Integration with other ATG applications such as ATG Commerce Service Center
- Integration with third-party software such as CyberSource

CRS is intended to illustrate best practices for creating stores with ATG Commerce. You can use the application as a template for building your own stores. The CRS installation includes all of the CRS-specific source code used in the application, including all Java, JSP, JavaScript, and Cascading Stylesheet (CSS) files.

Java class files are located in the `/src` subdirectories of the various CRS application modules. For example, class files for CRS-specific extensions to the ATG Commerce Fulfillment system are found in:

```
<ATG10dir>/CommerceReferenceStore/Store/Fulfillment/src/
```

More general functionality is found in:

```
<ATG10dir>/CommerceReferenceStore/Store/EStore/src/
```

The CRS storefronts are run from two J2EE WAR files, `store.war` and `storedocroot.war`. JSPs, JavaScript files, and CSS files are located in subdirectories of `store.war`, while static HTML and image files are found in subdirectories of `storedocroot.war`. These WAR files are both found in:

```
<ATG10dir>/CommerceReferenceStore/Store/Storefront/j2ee-apps/Storefront/
```

Throughout this manual, references to these files are always relative to this directory. So, for example, `store.war/browse/category.jsp` refers to:

```
<ATG10dir>/CommerceReferenceStore/Store/Storefront/
j2ee-apps/Storefront/store.war/browse/category.jsp
```

Note: Although ATG Commerce Reference store simulates a production environment, the code is not truly production-ready. Before you go live with any web application built with CRS, you need to optimize the performance of the application. For example, you should create an optimized Dojo build, as discussed in the Pages and Navigation chapter.

Multisite Support

CRS is an example of a multisite application, where multiple sites are run from the same ATG instance and share certain data and other resources. CRS includes three sites: ATG Store US, ATG Store Germany, and ATG Home. See the Multisite Features chapter for information about these sites.

Configuring ATG Commerce Reference Store

This section describes mechanisms for specifying global configuration options for your installation of CRS:

- The store configuration component (`/atg/store/StoreConfiguration`) maintains a set of global settings that apply to all CRS sites, such as whether or not verification codes are required for credit-card transactions.
- The ATG modules you include in your assembled application determine whether certain features (e.g., internationalization) and integrations (e.g., CyberSource) are available.

In addition to these global settings, CRS adds site configuration properties to the `siteConfiguration` item type in the Site Repository. See the Multisite Features chapter for information about these site-specific settings.

Note that to enable CRS to send e-mail messages, you must also configure the `/atg/dynamo/service/SMTPEmail` component. See the *Email Senders and Listeners* section of the *ATG Programming Guide*.

StoreConfiguration Component

The following table describes the properties of the `StoreConfiguration` component:

Property Name	Description
`requireCreditCardVerification`	Boolean indicating whether orders paid for by credit card must include the credit card verification code. This property is set to `true` by default.
`mandatoryStateCountryList`	A list of countries for which `state` is a required field, in the form of a comma-separated list of two-letter, upper-case country codes, as defined by ISO 3166. This property is set to `US` by default.
`atgSearchInstalled`	Boolean indicating if ATG Search is available. Used on JSP pages to determine the search forms to use and to decide whether to render facets. This property is set to `false` by default, but is set to `true` if the application includes the `Store.Search.Query` module.
`useSearchForSubcategoryProductList`	Boolean indicating if an ATG Search request should be used to obtain the list of products in a subcategory. This property is set to `false` by default, which means standard catalog navigation is used to obtain the list. Using ATG Search enables the use of ATG Search Merchandising rules or ATG Search sort options to control ordering. Using standard catalog navigation gives better performance and allows merchandisers to control ordering through the category's `fixedChildProducts` list.
`defaultResourceBundle`	The resource bundle to use for resource strings, if no resource bundle is defined in the site configuration's `resourceBundle` property. This property is set to `atg.projects.store.web.WebAppResources` by default.
`defaultCssFile`	The path for the base filename of the cascading stylesheets, if no CSS location is defined in the site configuration's `cssFile` property. This property is set to `/css/site/store` by default.
`siteHttpServerName`	The name of the HTTP server. This property is used in e-mail templates to construct full URLs to storefront pages. By default, this property is linked to the value of the `siteHttpServerName` property of the `/atg/dynamo/Configuration` component. On an agent-facing server, though, it should be set explicitly to the name of the production server.

Property Name	Description
`siteHttpServerPort`	The port number of the HTTP server. This property is used in e-mail templates to construct full URLs to storefront pages. By default, this property is linked to the value of the `siteHttpServerPort` property of the `/atg/dynamo/Configuration` component. On an agent-facing server, though, it should be set explicitly to the port number of the production server.
`dojoDebug`	Boolean indicating whether Dojo debugging is enabled. This property is set to `false` by default.
`dojoUrl`	This property is not used.

Assembling Your EAR Files

CRS includes all of the code, data, and configuration necessary for setting up a complete multi-server environment. When you set up CRS using the ATG Configuration and Installation Manager (CIM), it assembles separate EAR files for your publishing server and your production server. Depending on the options you specify, these environments may include support for ATG Search, internationalization, or both. See the *ATG Commerce Reference Store Installation and Configuration Guide* for more information.

If you reassemble your EAR files, you should be sure to specify the modules for the options you want to include. The following table describes the main CRS application modules and indicates which servers each module should be run on:

Module	Description	Server
`Store.Storefront`	Runs the storefront web applications.	Production
`Store.Fulfillment`	Runs ATG Commerce Fulfillment with CRS extensions.	Production (or separate ATG instance)
`Store.DCS-CSR`	Adds configuration to ATG Commerce Service Center for working with CRS.	Agent-facing
`Store.Cybersource`	Runs ATG Commerce CyberSource integration with CRS extensions.	Production
`Store.Search`	Enables use of ATG Search.	Publishing and Production

Module	Description	Server
`Store.Estore`	Contains the back-end classes for CRS.	Publishing and Production
`Store.Estore.Versioned`	Enables use of versioned repositories.	Publishing
`Store.Estore.International`	Enables ATG Store Germany and ability to view ATG Store US and ATG Home in Spanish.	Publishing and Production
`Store.Estore.International.Search`	Enables use of ATG Search with internationalized site.	Publishing and Production

If Your Environment Does Not Include ATG Content Administration

In an environment that includes ATG Content Administration (CA), assets are imported into the publishing server during the installation process. These assets include both repository-based assets such as the product catalog, and file-based assets such as targeters and scenarios. These assets are subsequently deployed to the production server for use on the CRS storefronts. This process ensures that file-based assets are managed properly through CA’s versioned file store.

This means, however, that if you run the CRS storefronts in an environment that does not include CA, the file-based assets won’t be present, and the site will not function properly. Therefore, CRS includes a special `Store.Storefront.NoPublishing` module, which adds the file-based assets to the production server. Include this module when you assemble your production EAR file. Note that the file-based assets in this module are not accessible through the ATG Business Control Center (BCC) and cannot be easily removed from the site. Therefore, you should not use this option if your environment does include CA.

Viewing Store Assets in the ATG Business Control Center

Once you assemble, deploy, and start up your publishing EAR file, you can access the ATG Business Control Center at the following URL:

```
http://hostname:port/atg/bcc
```

The hostname and port should match the values specified when CRS is installed. See the *ATG Commerce Reference Store Installation and Configuration Guide* for more information.

Viewing the CRS Storefronts

Once you assemble, deploy, and start up your production EAR file, you can access the various CRS storefronts (sites). To access ATG Store US, the URL is:

```
http://hostname:port/crs/storeus
```

The URL for ATG Home is:

```
http://hostname:port/crs/homeus
```

The URL for ATG Store Germany is:

```
http://hostname:port/crs/storede
```

In each of these URLs, the hostname and port should match the values specified when CRS is installed. See the *ATG Commerce Reference Store Installation and Configuration Guide* for more information.

ATG Commerce Reference Store Documentation

The CRS documentation is primarily intended to address the needs of two types of users:

- Evaluators who will want to quickly install CRS and get it up and running
- Developers who want to use CRS as a starting point for their own custom development

About This Manual

The *ATG Commerce Reference Store Overview* discusses the main features of the CRS application, how these features are implemented, and the design decisions these implementations reflect. It attempts to be as broad as possible in order to cover all major aspects of CRS, and to suggest starting points for further exploration and development.

In addition to this *Introduction*, it includes the following chapters:

Multisite Features

Pages and Navigation

Personalization and Promotions

Product Catalog

Inventory Management

Order Handling

Search Engine Optimization

ATG Search Integration

ATG Commerce Service Center Integration

Internationalization

Appendix A: Implementing a Related Regional Store

Appendix B: Sample Data

Related Documents

In addition to this *Overview*, the documentation for CRS includes:

ATG Commerce Reference Store Installation and Configuration Guide
Instructions for installing, configuring, and running the ATG Commerce Reference Store application. Includes two installation paths, one for evaluators who want to get CRS up and running quickly, and a second for developers who will use CRS as a starting point for custom development.

Entity Relationship Diagrams
Entity relationship diagrams showing the default database schema and table relationships for CRS.

ATG CRS API Reference
Javadoc descriptions of the CRS classes.

2 Multisite Features

This chapter describes ATG Commerce Reference Store's implementation of multisite features. It includes these sections:

Note: This section is focused on ATG Commerce Reference Store's implementation of multisite features. For more general information on multisite applications and how to implement them, see the *ATG Multisite Administration Guide*.

Multisite Implementation

CRS is an example of a multisite implementation, where multiple sites are run from the same ATG instance and share certain data and other resources. CRS includes three sites:

- ATG Store US primarily sells men's and women's apparel. The store is based in the United States, and shares a shopping cart and comparison lists with ATG Home. It can be viewed in English or Spanish.
- ATG Store Germany has the same catalog as ATG Store US, but the store is based in Germany and prices are in euros rather than dollars. ATG Store Germany can be viewed in German or English.
- ATG Home primarily sells home furnishings. The store is based in the United States, and shares a shopping cart and comparison lists with ATG Store US. ATG Home can be viewed in English or Spanish.

The ATG Store catalog and the ATG Home catalog are distinct but have some overlap. Both catalogs share the top-level Home Accents category and products.

Note: This guide assumes you have configured your CRS implementation to include the `Store.Estore.International` module. The `International` module is necessary for sites that will support multiple languages or multiple countries. If you did not install the `International` module, your production instance of CRS will include the English versions of ATG Store US and ATG Home only. You will not see ATG Store Germany or the Spanish translations for ATG Store US and ATG Home.

Site Configuration Extensions

CRS adds a number of site configuration properties to the Site repository for specifying values used by each of its sites. The following table describes the site configuration properties added by the CRS `Store.Estore` module.

Note: This section provides a quick description of the CRS site configuration properties. Many of these properties are described in more detail elsewhere in this manual.

billableCountries, nonBillableCountries
`billableCountries` stores a list of countries that the store may bill to while `nonBillableCountries` stores a list of countries that the store cannot bill to. Use the `billableCountries` property in those instances where it is more efficient to list a small number of billable countries. By contrast, use `nonBillableCountries` when it is more efficient to list a small number of non-billable countries. Both properties take a value that is a comma-separated list of two-letter, upper-case country codes, as defined by ISO 3166.

The `/atg/store/droplet/BillingRestrictionsDroplet` servlet bean, which is used by a number of CRS gadgets, considers both properties when determining if a billing address is valid for a particular item:

- If `billableCountries` is null, all countries are billable, except those listed in `nonBillableCountries`.
- If `billableCountries` contains one or more countries, only those countries are billable and `nonBillableCountries` is ignored.

CRS sets the `billableCountries` property for all three sites to include US, CA, MX, and DE.

shippableCountries, nonShippableCountries
`shippableCountries` stores a list of countries a product may be shipped to while `nonShippableCountries` stores a list of countries a product may not be shipped to. Use the `shippableCountries` property in those instances where it is more efficient to list a small number of countries to which a product may be shipped. By contrast, use `nonShippableCountries` when it is more efficient to list a small number of countries to which a product may not be shipped. Both properties take a value that is a comma-separated list of two-letter, upper-case country codes, as defined by ISO 3166.

The `/atg/store/droplet/ShippingRestrictionsDroplet` servlet bean, which is used by a number of CRS gadgets, considers both properties when determining if a shipping address is valid for a particular item:

- If `shippableCountries` is null, all countries are shippable, except those listed in `nonShippableCountries`.
- If `shippableCountries` contains one or more countries, only those countries are shippable and `nonShippableCountries` is ignored.

CRS sets the `shippableCountries` property for all three sites to include US, CA, MX, and DE.

newProductThresholdDays
The maximum number of days since a given product's `creationDate` for the product

to be considered new. CRS sets this value to 15 days for all three sites. **Note:** Because demonstration data goes stale over time, CRS uses some additional logic to determine which products are new. See the Product Catalog chapter for more information.

resourceBundle
The path to the default resource bundle containing the translatable text for the store pages. CRS sets this property to `atg.projects.store.web.WebAppResources` for all three sites.

The `resourceBundle` property must contain the fully qualified name of the default resource bundle, but it is not necessary to specify the translated variants of the bundle. For example, `atg.projects.store.web.WebAppResources` is the default resource bundle for CRS, but CRS also includes `atg.projects.store.web.WebAppResources_de` and `atg.projects.store.web.WebAppResources_en` resource bundles to support the German and Spanish translations, respectively.

If the `resourceBundle` property is not set for a site, CRS uses the `/atg/store/StoreConfiguration` component's `defaultResourceBundle` property instead. This property is set to `atg.projects.store.web.WebAppResources` in CRS.

defaultPageSize
The default number of products to be displayed on a subcategory or search results page. CRS sets this value to 12 for all three sites.

cssFile
The path for the base filename of the cascading stylesheets for the site. CRS sets this property to:

- `/css/site/store` for ATG Store US and ATG Store Germany
- `/css/site/home` for ATG Home

The value for this property is not a fully qualified pathname. CRS includes additional logic for resolving the full pathname when it uses this property. See Cascading Stylesheets (CSS) for more information.

If the `cssFile` property is not set for a site, CRS uses the `/atg/store/StoreConfiguration` component's `defaultCssFile` property instead. This property is set to `/css/site/store` in CRS.

largeSiteIcon
The pathname of the large icon used to identify the site. CRS uses pathnames that are relative to the `storedocroot.war` web application's context root, which is `/crsdocroot`.

defaultCountry
The default country for the site. CRS sets this value to `US` for ATG Store US and ATG Home and `DE` for ATG Store Germany.

defaultLanguage
The default language for the site, in the form of a two-letter, lower-case language code as defined by ISO 639. CRS sets the `defaultLanguage` property to en for ATG Store US and ATG Home and de for ATG Store Germany. **Note:** This property is added by the `Store.Estore.International` module.

languages
The languages supported by the site, in the form of a comma-separated list of two-letter, lower-case language codes as defined by ISO 639. CRS sets the `languages` property to `en,es` for ATG Store US and ATG Home, and to `en,de` for ATG Store Germany. **Note:** This property is added by the `Store.Estore.International` module.

emailAFriendEnabled
Indicates whether the store supports the "e-mail a friend" feature. CRS sets this property to enabled for all three sites.

backInStockFromAddress, newPasswordFromAddress, orderConfirmationFromAddress, orderShippedFromAddress, promotionEmailAddress
The sender address for e-mails that:

- Notify customers a previously backordered SKU is back in stock.
- Supply new passwords to users who click the "Forgot your password" link on the Login page.
- Provide order confirmation information after an order is placed.
- Notify customers that their orders have been shipped.
- Notify customers about promotions they have been granted.

CRS sets these properties to:

- `ATGStoreService@example.com` for ATG Store US
- `ATGStoreService@example.de` for ATG Store Germany
- `ATGHomeService@example.com` for ATG Home

Site Data Sharing

This section discusses data and resources shared by the three CRS sites.

CRS Uses the Default Site Category

CRS uses the default site category for all three of its sites. For more information on the default site category, see the *ATG Multisite Administration Guide*.

Sharing Groups

CRS implements two sharing groups:

- Site Group US includes the ATG Store US and ATG Home sites and the `ShoppingCartShareableType` component. It is an example of a sharing group that specifies Nucleus components (the shopping cart and comparison lists) as the shared data.

- Related Regional Stores includes the ATG Store US and ATG Store Germany sites and the `RelatedRegionalStoresShareableType` component. It is an example of a sharing group that uses a non-Nucleus shareable type component. See the Shareable Type Components section for details on how this `ShareableType` component works.

Shareable Type Components

In addition to the `ShoppingCartShareableType` component that comes configured out of the box, CRS implements a second `ShareableType` component, `/atg/store/RelatedRegionalStoresShareableType`. `RelatedRegionalStoresShareableType` is a non-Nucleus shareable type component that represents the concept of regional peers of the same store. CRS has a sharing group, Related Regional Stores, that includes the ATG Store US and ATG Store Germany sites, along with the `RelatedRegionalStoresShareableType` component. This configuration indicates that ATG Store US and ATG Store Germany are regional peers.

CRS JSP code uses the `/atg/dynamo/droplet/multisite/SharingSitesDroplet` to determine which other sites are in a Related Regional Stores sharing group with the current site; for example, when ATG Store US is the current site, the `SharingSitesDroplet` returns ATG Store Germany, and vice versa. This approach allows CRS to implement one generic piece of JSP code that renders a widget that allows shoppers to switch among regional peer sites (see Rendering the Store and Country Pickers for more details and a code example). This approach also eliminates the need for site-specific JSP code that says "if site A is the current site, render widgets for sites B and C; if site B is the current site, render widgets for sites A and C, and so on." Adding a new site to a group of regional peers only requires adding the site to the Related Regional Stores sharing group in Site Administration. No JSP code needs to change in order to retrieve and render the additional regional peer (although presentation code may need to change if there is not enough room for the additional site to appear in the UI).

The `RelatedRegionalStoresShareableType` component is defined in the `/atg/store/RelatedRegionalStoresShareableType.properties` file found in CRS's `Store.Estore.International` module. Its definition looks like this:

```
$class=atg.multisite.ShareableType

# The shareable type ID used by application code
id=crs.RelatedRegionalStores

# Information used to find strings appropriate for localized UIs
displayNameResource=relatedRegionsShareableTypeName

resourceBundleName=
atg.projects.store.multisite.InternationalStoreSiteRepositoryTemplateResources
```

For more details on rendering the country picker in CRS, see Changing Site Context. For an overview of non-Nucleus shareable type components, see the *ATG Multisite Administration Guide*. For details on creating non-Nucleus shareable type components, see the *Multisite Request Processing* chapter of the *ATG Programming Guide*.

Cross-site Registration and Logins

User profiles are not segmented by site. This means that user profiles and, therefore, logins, are shared across all sites in an application. By extension, the same holds true for registration. When a customer registers on one site, he is registering for all sites.

URL Strategy

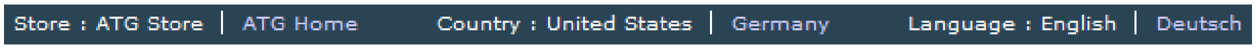
CRS implements a path-based URL strategy that defines `/crs` as the context root for the entire application along with the following virtual context roots for the individual sites:

- ATG Store US: `/crs/storeus`
- ATG Store Germany: `/crs/storede`
- ATG Home: `/crs/homeus`

For an overview of the path-based URL strategy, see the *ATG Multisite Administration Guide*. For detailed information on implementing the path-based strategy see *Multisite URL Management* in the *ATG Programming Guide*.

Changing Site Context

CRS includes the following controls for changing the store, country, and language context:


Store : ATG Store | ATG Home
Country : United States | Germany
Language : English | Deutsch

This section describes how these features are implemented.

Rendering the Store and Country Pickers

The store and country pickers allow the customer to switch between the ATG Store US, ATG Store Germany, and ATG Home sites:

- The store picker renders links to affiliated sites that share a shopping cart. CRS has two affiliated sites, ATG Store US and ATG Home, that share a shopping cart.
- The country picker renders links to regional peers of the same site. CRS has two regional peer sites, ATG Store US and ATG Store Germany.

If, in the current site context, a picker doesn't have any sites to display, the picker won't be rendered. For example, ATG Store Germany does not share its shopping cart with any other affiliated sites, so when ATG Store Germany is the current site, the store picker is not rendered.

To add controls that change site context, you must retrieve a list of site IDs for the sites to which you want to provide switching controls. CRS uses the `/atg/dynamo/droplet/multisite/SharingSitesDroplet` to retrieve these IDs:

- For the store picker, CRS passes `atg.ShoppingCart` as the shareable type ID to the `SharingSitesDroplet`, so that the droplet returns all sites that share a shopping cart with the current site. For example, if ATG Store US is the current site, `SharingSitesDroplet` returns ATG Home, and vice versa.
- For the country picker, CRS passes `atg.RelatedRegionalStores` as the shareable type ID, so that the `SharingSitesDroplet` returns all sites that are in a Related Regional Stores sharing group with the current site. For example, if ATG Store US is the current site, the droplet returns ATG Store Germany, and vice versa.

After you have retrieved a list of site IDs, you render links to the sites associated with those IDs. The `store.war/navigation/gadgets/sites.jsp` gadget, shown below, uses the `/atg/dynamo/droplet/multisite/SiteLinkDroplet` to build the links that are included in the Store picker.

```
<dsp:page>
  <dsp:importbean bean="/atg/multisite/Site" />
  <dsp:importbean bean="/atg/dynamo/droplet/multisite/SharingSitesDroplet" />
  <dsp:importbean bean="/atg/dynamo/droplet/ForEach" />

  <%-- Get the site ID for the current site. The current site should not
  be rendered as a link in the Store site picker. --%>
  <dsp:getvalueof var="currentSiteId" bean="Site.id" />

  <%-- Retrieve the sites that share a shopping cart with the current site. --%>
  <dsp:droplet name="SharingSitesDroplet">
    <dsp:param name="shareableTypeId" value="atg.ShoppingCart" />
    <dsp:oparam name="output">

      <dsp:getvalueof var="sites" param="sites" />
      <dsp:getvalueof var="size" value="${fn:length(sites)}" />

      <c:if test="${size > 1}">
        <div id="atg_store_sites">
          <h2>
            <fmt:message key=
                      "navigation_internationalStores.internationalStoresTitle" />
            <fmt:message key="common.labelSeparator" />
          </h2>

          <ul>

          <%-- Iterate over the sites in the shopping cart sharing group. --%>
          <dsp:droplet name="ForEach">
            <dsp:param name="array" param="sites" />
            <dsp:setvalue param="site" paramvalue="element" />
            <dsp:param name="sortProperties" value="-name" />

            <dsp:oparam name="output">
              <dsp:getvalueof var="size" param="size" />
              <dsp:getvalueof var="count" param="count" />
              <dsp:getvalueof var="siteName" param="site.name" />
              <dsp:getvalueof var="siteId" param="site.id" />

              <li class="<crs:listClass count="${count}" size="${size}"
                  selected="${siteId == currentSiteId}" />"
                <c:choose>

                  <%-- For the current site, render its name only. --%>
                  <c:when test="${siteId == currentSiteId}">
                    <dsp:valueof value="${siteName}" />
                  </c:when>

                 <%-- For other sites, render a cross-site link. --%>
                  <c:otherwise>
                    <dsp:include page=
                                 "/global/gadgets/crossSiteLinkGenerator.jsp">
                      <dsp:param name="siteId" value="${siteId}" />
                      <dsp:param name="customUrl" value="/" />
                    </dsp:include>
                    <dsp:a href="${siteLinkUrl}"
                           title="${siteName}">${siteName}</dsp:a>
                  </c:otherwise>

                </c:choose>
              </li>
            </dsp:oparam>
          </dsp:droplet>
          </ul>
        </div>
      </c:if>
    </dsp:oparam>
  </dsp:droplet>
</dsp:page>
```

The `store.war/navigation/gadgets/regions.jsp` gadget uses similar code to render the country picker:

```
<dsp:page>
  <dsp:importbean bean="/atg/multisite/Site" />
  <dsp:importbean bean="/atg/dynamo/droplet/ComponentExists" />
  <dsp:importbean bean="/atg/dynamo/droplet/ForEach" />
  <dsp:importbean bean="/atg/dynamo/droplet/multisite/SharingSitesDroplet" />
```

```
<%-- Verify that this is an international storefront. If so, the Country portion
of the site picker should be rendered. --%>
  <dsp:droplet name="ComponentExists">
    <dsp:param name="path" value="/atg/modules/InternationalStore" />
    <dsp:oparam name="true">

<%-- Retrieve the sites that are in the Related Regional Stores sharing group with
the current site. --%>
      <dsp:droplet name="SharingSitesDroplet">
        <dsp:param name="shareableTypeId" value="crs.RelatedRegionalStores" />
        <dsp:oparam name="output">

          <dsp:getvalueof var="sites" param="sites" />
          <dsp:getvalueof var="size" value="${fn:length(sites)}" />

          <c:if test="${size > 1}">

            <%-- Get the site ID for the current site. The current site should
            not be rendered as a link in the site picker. --%>
            <dsp:getvalueof var="currentSiteId" bean="Site.id" />

            <div id="atg_store_regions">
              <h2>
                <fmt:message key="navigation_internationalStores.RegionsTitle" />
                <fmt:message key="common.labelSeparator" />
              </h2>

              <ul>
                <dsp:droplet name="ForEach">
                  <dsp:param name="array" param="sites" />
                  <dsp:setvalue param="site" paramvalue="element" />
                  <dsp:param name="sortProperties" value="-countryDisplayName" />

                  <dsp:oparam name="output">
                    <dsp:getvalueof var="size" param="size" />
                    <dsp:getvalueof var="count" param="count" />
                    <dsp:getvalueof var="countryName"
                         param="site.countryDisplayName" />
                    <dsp:getvalueof var="siteId" param="site.id" />

                    <li class="<crs:listClass count="${count}" size="${size}"
                        selected="${siteId == currentSiteId}" />
                      <c:choose>

                        <%-- For the current site, render its name only. --%>
                        <c:when test="${siteId == currentSiteId}">
                          <dsp:valueof value="${countryName}" />
                        </c:when>

                        <%-- For other sites, render a cross-site link. --%>
                        <c:otherwise>
                          <dsp:include page=
                                     "/global/gadgets/crossSiteLinkGenerator.jsp">
                            <dsp:param name="siteId" value="${siteId}" />
                            <dsp:param name="customUrl" value="/" />
                          </dsp:include>
                          <dsp:a href="${siteLinkUrl}"
                               title="${countryName}">${countryName}</dsp:a>
                        </c:otherwise>

                      </c:choose>
                    </li>
                  </dsp:oparam>
                </dsp:droplet>
              </ul>
            </div>
          </c:if>
        </dsp:oparam>
      </dsp:droplet>
    </dsp:oparam>
  </dsp:droplet>
</dsp:page>
```

For details on the `SharingSitesDroplet` and `SiteLinkDroplet` components, see the *ATG Page Developer's Guide*.

Rendering the Language Picker

The language picker allows the customer to switch between English and Spanish when ATG Store US or ATG Home is the current site, and English and German when ATG Store Germany is the current site. In order for this functionality to be available, the `Store.Estore.International` module must be running in your production instance.

Note: When the `International` module is absent, ATG Store US and ATG Home are only available in English and ATG Store Germany is not available at all.

The language picker is implemented by the `store.war/navigation/gadgets/languages.jsp` gadget. To begin, `languages.jsp` tests whether or not the `/atg/modules/InternationalStore` component exists. This component is part of the `Store.Estore.International` module and only exists when that module is running. The presence of the `InternationalStore` component indicates that the `International` module is running and, therefore, this application has language support and a language picker should be rendered.

Next, `languages.jsp` invokes the `/atg/store/droplet/DisplayLanguagesDroplet`, passing in as parameters the `languages` and `defaultCountry` properties of the current site. The `DisplayLanguagesDroplet` is of class `atg.projects.store.droplet.DisplayLanguagesDroplet` and it has two tasks: it determines which language is currently selected and it creates a set of objects, one for each language the current site supports. In general, these objects contain a modified URL that consists of the current URL plus a locale parameter. For example, if the URL for the current page is:

```
http://localhost:8180/crs/storeus/index.jsp
```

The modified URL might be:

```
http://localhost:8180/crs/storeus/index.jsp?locale=es_US
```

There are some exceptions to this rule for parameters that are not appropriate to carry over to the modified URL. For example, the `numResults` parameter is omitted in any newly formed URLs because a search in a new language will return different results.

Finally, `languages.jsp` uses the objects returned by `DisplayLanguagesDroplet` to render the language picker. The currently selected language is rendered as a label only. Any other languages are rendered as links using the URLs in the objects the `DisplayLanguageDroplet` returned.

Note: See the Internationalization chapter for more information on the International module.

3 Pages and Navigation

This chapter describes the front-end elements of the ATG Commerce Reference Store storefronts and how they are implemented. It includes the following topics:

JSP Page Architecture

The JSP pages in CRS are designed to be modular, with individual pieces of functionality and logic encapsulated in page fragments and custom tags for easy reuse. This section describes two main elements of the page architecture:

- Gadgets, which are JSP pages containing specific functionality such as UI controls
- Custom JSP tags, especially container tags, which implement standard page elements such as headers and footers

Gadgets

A gadget is an individual piece of functionality encapsulated in a JSP file so it can easily be included in multiple pages or moved to different locations on a page. For example, the links for sorting product listings on a category page execute logic implemented by the `store.war/global/gadgets/sortGadget.jsp` file.

The use of gadgets makes it possible to build pages from reusable elements. For example, consider the `store.war/navigation/gadgets/search.jsp` gadget, which creates the search dialog box found on most pages:

```
<dsp:page>
  <dsp:importbean bean="/atg/store/StoreConfiguration" />

   <div id="atg_store_search">
    <dsp:getvalueof var="atgSearchInstalled"
         bean="StoreConfiguration.atgSearchInstalled" />
    <c:choose>
      <c:when test="${atgSearchInstalled == 'true'}">
        <dsp:include page="/atgsearch/gadgets/atgSearch.jsp" />
      </c:when>
      <c:otherwise>
        <dsp:include page="/search/gadgets/simpleSearch.jsp" />
      </c:otherwise>
    </c:choose>
  </div>

</dsp:page>
```

This gadget determines whether ATG Search is installed and, if it is, invokes another gadget that creates a search form for issuing queries to ATG Search. If ATG Search is not installed, the `store.war/navigation/gadgets/search.jsp` gadget instead invokes a gadget that creates a form for using ATG's basic search facility.

Custom Tags

ATG Commerce Reference Store includes the CRS custom tag library. These tags are implemented as `.tag` files, which are found in `store.war/WEB-INF/tags/store/`.

Some of the custom tags are similar to gadgets in that they encapsulate reusable pieces of functionality. Other tags, known as container tags, make up a key aspect of the CRS page architecture. A container tag can insert standard elements in a page, and also encloses a `<jsp:body>` tag for rendering the main page content. For example, many pages use the `<crs:pageContainer>` tag, which adds header, footer, and navigational elements to the page. This approach significantly reduces the number of JSP includes that are required.

The following example of using a container tag is adapted from `store.war/checkout/giftMessage.jsp`, an optional page in the checkout process:

```
<crs:checkoutContainer currentStage="shipping"
                       showOrderSummary="true"
                       skipSecurityCheck="true"
                       title="${title}">
  <jsp:body>
    <dsp:include page="gadgets/giftMessage.jsp" flush="true" />
  </jsp:body>
</crs:checkoutContainer>
```

In this example, the `<crs:checkoutContainer>` tag is used to display a progress bar for the checkout process and include order summary information, while the enclosed `<jsp:body>` tag includes a gadget that renders a form for entering a gift message.

JavaScript and Cascading Stylesheets

All CRS sites use the same JSPs. These pages use JavaScript and Cascading Stylesheets (CSS) to differentiate the look of the storefronts and to enhance the visual presentation of the user interface. This section describes the main uses of JavaScript and CSS in CRS.

Note that if JavaScript is disabled in the customer's browser, most of the store's JavaScript-based functionality is replaced by equivalent functionality that does not require JavaScript. For example, for pages that use the color/size picker, the picker is replaced by a dropdown menu that lists each available color/size combination.

JavaScript

All of the JavaScript used by CRS is stored in external `.js` files that are loaded by the `store.war/includes/pageStartScript.jsp` file, which is included in most CRS pages. In addition, a subset of these `.js` files is loaded by the `store.war/includes/popupStartScript.jsp` file, which is included in popup windows. Using `.js` files avoids having JavaScript behavior or logic hardcoded into the JSPs.

For a given page, only the JavaScript functions that are required are run. The JavaScript files contain logic that checks the rendered HTML for specific IDs and runs only the functions for which the associated ID is found. This makes the UI code modular and easily transportable between pages.

The CRS JavaScript files include:

- Core Dojo libraries
- Rich shopping cart widget
- Other CRS JavaScript

Core Dojo Libraries

CRS uses the Dojo 1.4.2 JavaScript library, which is distributed with the ATG platform in the `WebUI` module. This library consists of three JavaScript namespaces:

- Dojo -- core Dojo functionality
- Dijit -- reusable Dojo widgets
- DojoX -- experimental functionality

Note that the Dojo library included with the ATG platform is not optimized. Before you go live with any web application built with CRS, you should create an optimized Dojo build. The Dojo build scripts are included in this directory:

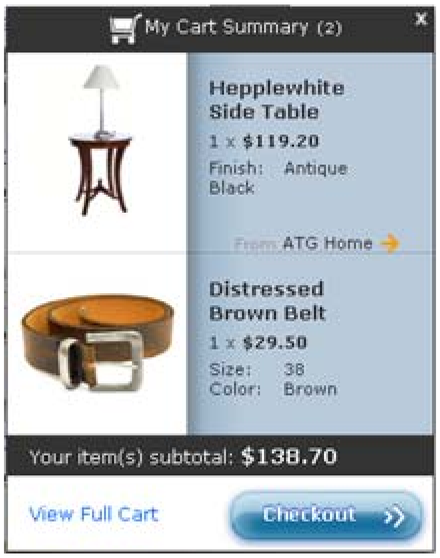
```
<ATG10dir>/WebUI/2ee-apps/WebUI.ear/dojo-1-4-2.war/util/buildscripts/
```

For information about Dojo, including instructions for building an optimized Dojo library, go to:

```
http://www.dojotoolkit.org
```

Rich Shopping Cart Widget

In addition to the widgets that are included in the Dojo libraries, CRS has a rich shopping cart user interface that is implemented as a custom Dojo widget. This widget displays the products in the shopping cart in a dropdown that is available on most pages. For example:



This widget makes it possible for customers to view the items in their cart without leaving the current page. It is implemented through three `.js` files that extend the standard Dojo widget classes:

- `store.war/javascript/widget/RichCartSummary.js` -- This is main file for the widget. It is responsible for "hijacking" the Add to Cart button, sending Ajax requests and handling responses, and displaying the cart contents.
- `store.war/javascript/widget/RichCartTrigger.js` -- This file implements the show/hide behavior of the rich cart.
- `store.war/javascript/widget/RichCartSummaryItem.js` -- This file renders the response when an item is successfully added to the cart.

Each of these files has a corresponding template file in the `store.war/javascript/widget/template/` directory.

Other CRS JavaScript

CRS also includes a number of JavaScript functions that are not implemented as Dojo widgets (although some of them do call Dojo widgets). These functions are stored in the JavaScript files in the `store.war/javascript/` directory. These files include implementations of features such as the SKU picker, as well as code for submitting faceted search requests via Ajax.

Cascading Stylesheets (CSS)

Most of the styling information for CRS is stored in external .css files that are loaded by the store.war/includes/pageStart.jsp and store.war/includes/popupStart.jsp files. Using these .css files avoids the need for inline style definitions. (The e-mail JSPs are an exception; they do contain styling information, because some e-mail clients don't support external .css files.)

CRS adds a cssFile property to the siteConfiguration item type of the Site Repository. This property is used to specify the base CSS filename for each site. The values stored for the three CRS sites are shown in the table below:

Site	Base CSS Filename
ATG Store US	/css/site/store
ATG Store Germany	/css/site/store
ATG Home	/css/site/home

The pageStart.jsp and popupStart.jsp files use the cssFile property value to construct pathnames to the site-specific CSS files located in store.war/css/site/. The pageStart.jsp gadget creates CSS links for all of the standard CRS pages, while popupStart.jsp creates CSS links for popup windows. The /css/site/ directory contains the following files:

- store.css -- Used by ATG Store US and ATG Store Germany when English is the current language.
- store_de.css -- Used by ATG Store Germany when German is the current language.
- store_de_ie.css -- Used by ATG Store Germany when German is the current language; includes additional settings for Internet Explorer.
- store_es.css -- Used by ATG Store US when Spanish is the current language.
- store_es_ie.css -- Used by ATG Store US when Spanish is the current language; includes additional settings for Internet Explorer.
- home.css -- Used by ATG Home when English is the current language.
- home_ie.css -- Used by ATG Home when English is the current language; includes additional settings for Internet Explorer.
- home_es.css -- Used by ATG Home when Spanish is the current language.
- home_es_ie.css -- Used by ATG Home when Spanish is the current language; includes additional settings for Internet Explorer.

The following section describes how pageStart.jsp and popupStart.jsp determine which CSS file to use.

Linking to the CSS Files

Most CRS pages include the `<crs:pageContainer>` tag. This tag includes the `store.war/includes/pageStart.jsp` file, which has code for determining which site-specific CSS files to use. This code retrieves the context path from the originating request and uses that, in conjunction with the base CSS filename from the site configuration's `cssFile` property, to build the pathname to the base CSS file. It then determines, depending on the current language and browser, whether to also include links to any language-specific or Internet Explorer-specific CSS files:

```
<dsp:page>
  <dsp:importbean bean="/atg/multisite/Site" />

  <dsp:getvalueof var="contextPath" bean="/OriginatingRequest.contextPath" />
  <dsp:getvalueof var="language"
     bean="/OriginatingRequest.requestLocale.locale.language" />

  ...

   <%-- Load the site specific CSS --%>
    <dsp:getvalueof var="siteId" bean="Site.id" />
    <dsp:getvalueof var="siteCssFile" bean="Site.cssFile" />
    <c:if test="${not empty siteCssFile}">
      <link rel="stylesheet" href="${contextPath}/${siteCssFile}.css"
            type="text/css" media="screen" title="no title" charset="utf-8" />

      <c:if test="${siteId eq 'homeSite'}">
        <%-- According to the conditional comment this is Internet Explorer load
             the site IE CSS --%>
        <!--[if IE]>
          <link rel="stylesheet" href="${contextPath}${siteCssFile}_ie.css"
                type="text/css" media="screen" title="no title" charset="utf-8" />
        <![endif]-->
      </c:if>

      <%-- Load language specific CSS if this is a language which requires
           additional styling --%>
      <c:if test="${language eq 'de'}">
        <link rel="stylesheet" href="${contextPath}${siteCssFile}_de.css"
              type="text/css" media="screen" title="no title" charset="utf-8" />
        <!--[if IE]>
          <link rel="stylesheet" href="${contextPath}${siteCssFile}_de_ie.css"
          type="text/css" media="screen" title="no title" charset="utf-8" />
        <![endif]-->
      </c:if>

      <c:if test="${language eq 'es'}">
        <link rel="stylesheet" href="${contextPath}${siteCssFile}_es.css"
              type="text/css" media="screen" title="no title" charset="utf-8" />
        <!--[if IE]>
```

```
          <link rel="stylesheet" href="${contextPath}${siteCssFile}_es_ie.css"
          type="text/css" media="screen" title="no title" charset="utf-8" />
        <![endif]-->
      </c:if>
    </c:if>

  ...

</dsp:page>
```

Similar logic is used to determine the CSS files for popup windows. Popup window JSPs include the `<crs:popupPageContainer>` tag, which includes the `store.war/includes/popupStart.jsp` file, which has code for determining which CSS files to use.

Other CSS Files

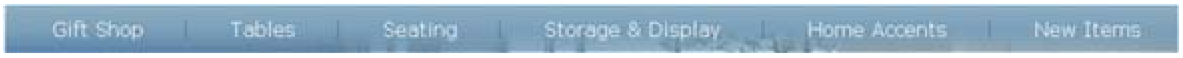
In addition to the site-specific CSS files mentioned above, CRS also has several style sheets that provide a foundation used for all CRS sites. These files are found in `store.war/css/`.

Navigation and Site Pages

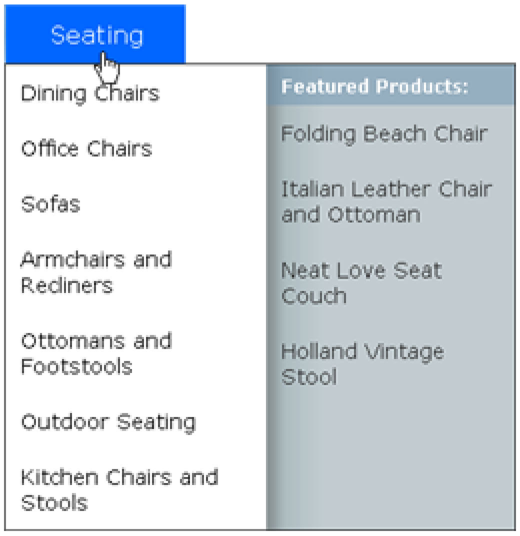
This section describes various aspects of navigating the CRS storefronts, the pages available to customers, and how the pages and navigational hierarchy are implemented.

Menu Bar

ATG Store and ATG Home have separate catalogs that share some categories. On each site, the `store.war/navigation/gadgets/catalog.jsp` gadget creates a menu bar (displayed at the top of most pages) which lists the top-level categories of the site. For example, on the ATG Home site:

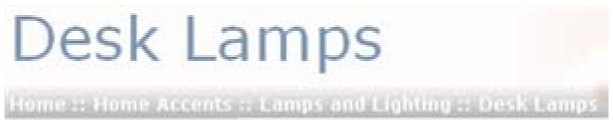


Hovering the cursor over an entry displays a menu of its subcategories, and a list of featured products in the category:



Historical Navigation

Category pages and product detail pages display a "breadcrumb" trail that reflects the navigational path to the page. For example:



Each entry in the trail is a link to the corresponding category page. So, on the page shown above, if the customer clicks Home Accents, the Home Accents category page is displayed.

If a customer accesses a product without proceeding through the navigational hierarchy, the breadcrumb trail reflects the default path to the product. There are several ways to make a navigational jump like this, such as clicking the link for a featured product or selecting a product returned by a search.

Navigation is tracked using the standard ATG Commerce historical navigation mechanism described in the *ATG Commerce Guide to Setting Up a Store*. The links to products displayed on various pages (e.g., category pages) pass parameters to the `/atg/commerce/catalog/CatalogNavHistoryCollector` servlet bean that specify modifications to the navigational history. The history is stored as a List of locations in the `navHistory` property of the `/atg/commerce/catalog/CatalogNavHistory` component. The breadcrumb trail is rendered by the `store.war/navigation/gadgets/breadcrumbs.jsp` gadget, which iterates over the `navHistory` property to display the locations it holds.

Template Pages

CRS uses a small number of template pages for displaying categories and products. The main template pages are:

- `store.war/browse/category.jsp` -- Displays a top-level category.
- `store.war/browse/subcategory.jsp` -- Displays a subcategory.
- `store.war/browse/productDetailSingleSku.jsp` -- Displays product details for a product that has a single SKU.
- `store.war/browse/productDetailMultiSku.jsp` -- Displays product details for a product with multiple SKUs. Provides a separate quantity field for each SKU.
- `store.war/browse/productDetailColorSizePicker.jsp` -- Displays product details with a color/size SKU picker.
- `store.war/browse/productDetailWoodFinishPicker.jsp` -- Displays product details with a wood finish SKU picker.

The template used to display a given item is specified by the item's `template` property. So, for example, if a customer clicks on a link to a specific product, the page used to display the product is determined by the value of this property.

Since the same page can be used for different categories or products, the actual category or product is specified through query parameters. For the category templates, the category is specified with the `categoryId` query parameter. For example:

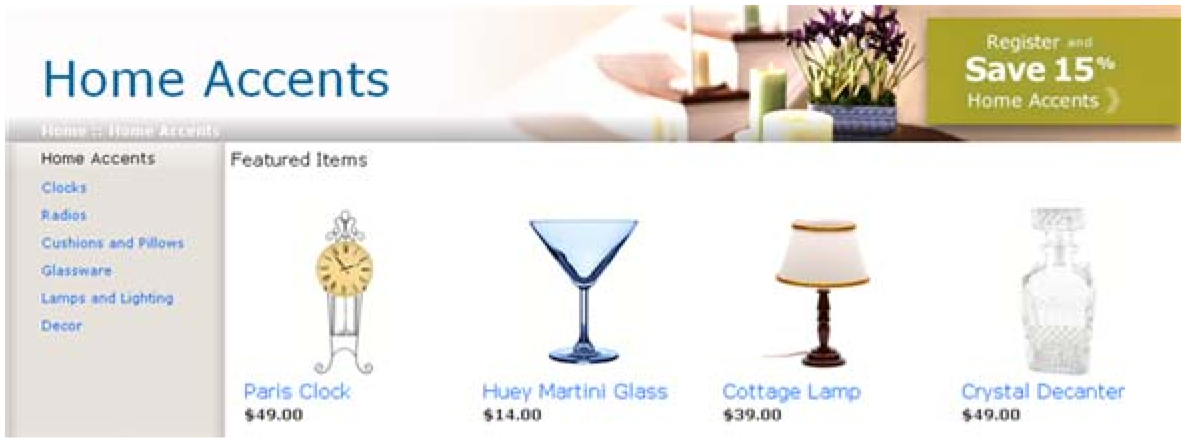
```
/browse/category.jsp?categoryId=cat10006
```

For the product templates, the product is specified with the `productId` query parameter, and its parent category is specified with the `categoryId` query parameter. For example:

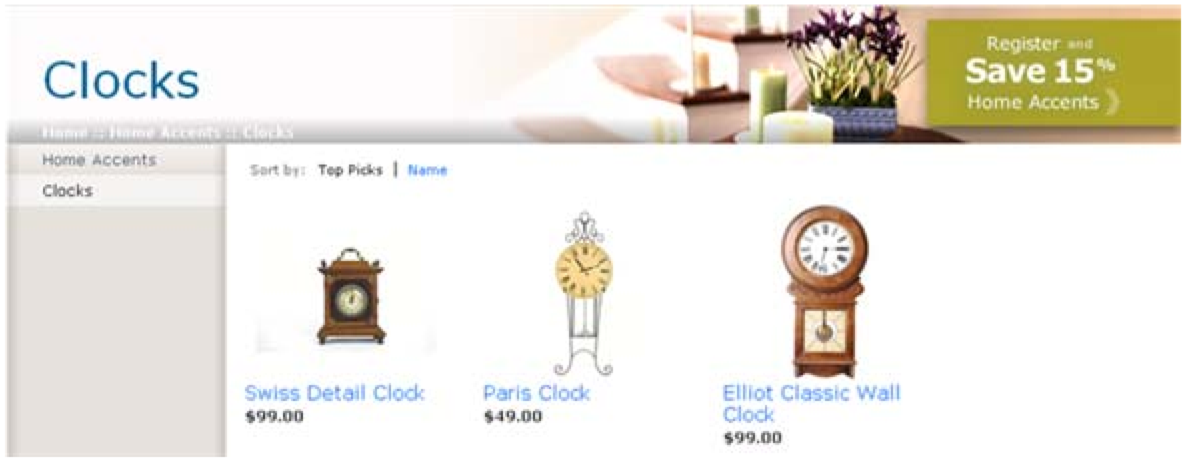
```
/browse/productDetailSingleSku.jsp?productId=xprod2035&categoryId=cat10056
```

Category Pages

CRS top-level categories contain child categories but no child products of their own. These categories use the `store.war/browse/category.jsp` template, which displays featured products from the child categories, plus links to the child categories on the left. So, for example, the page for Home Accents looks like this:

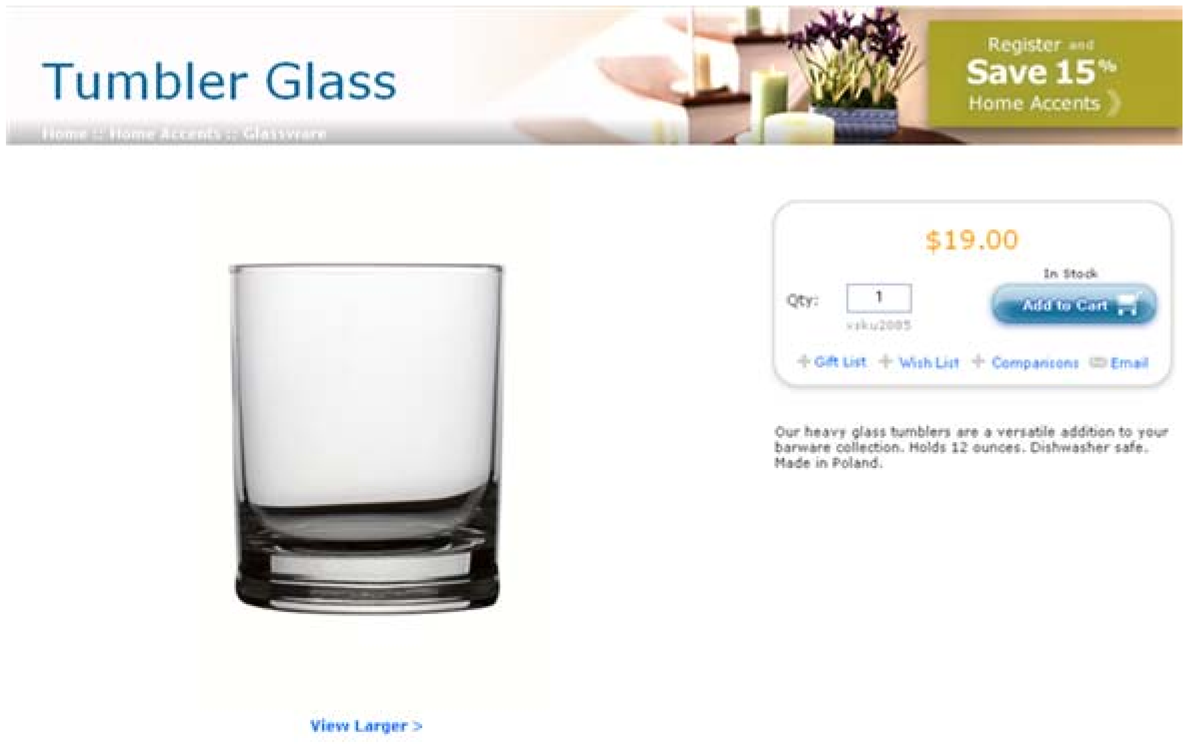


Subcategories use the `store.war/browse/subcategory.jsp` template. The page content differs depending on the whether the category contains child products or child categories. If the category includes child categories but no child products of its own, the page displayed is similar to a top-level category page; it shows featured products for the child categories, and links to the child categories. If the category includes child products, however, CRS displays a listing of those products. For example, the page for the Clocks subcategory of Home Accents looks like this:


Clocks
Register and Save 15% Home Accents
Home :: Home Accents :: Clocks
Home Accents
Clocks
Sort by: Top Picks | Name
Swiss Detail Clock
$99.00
Paris Clock
$49.00
Elliot Classic Wall Clock
$99.00

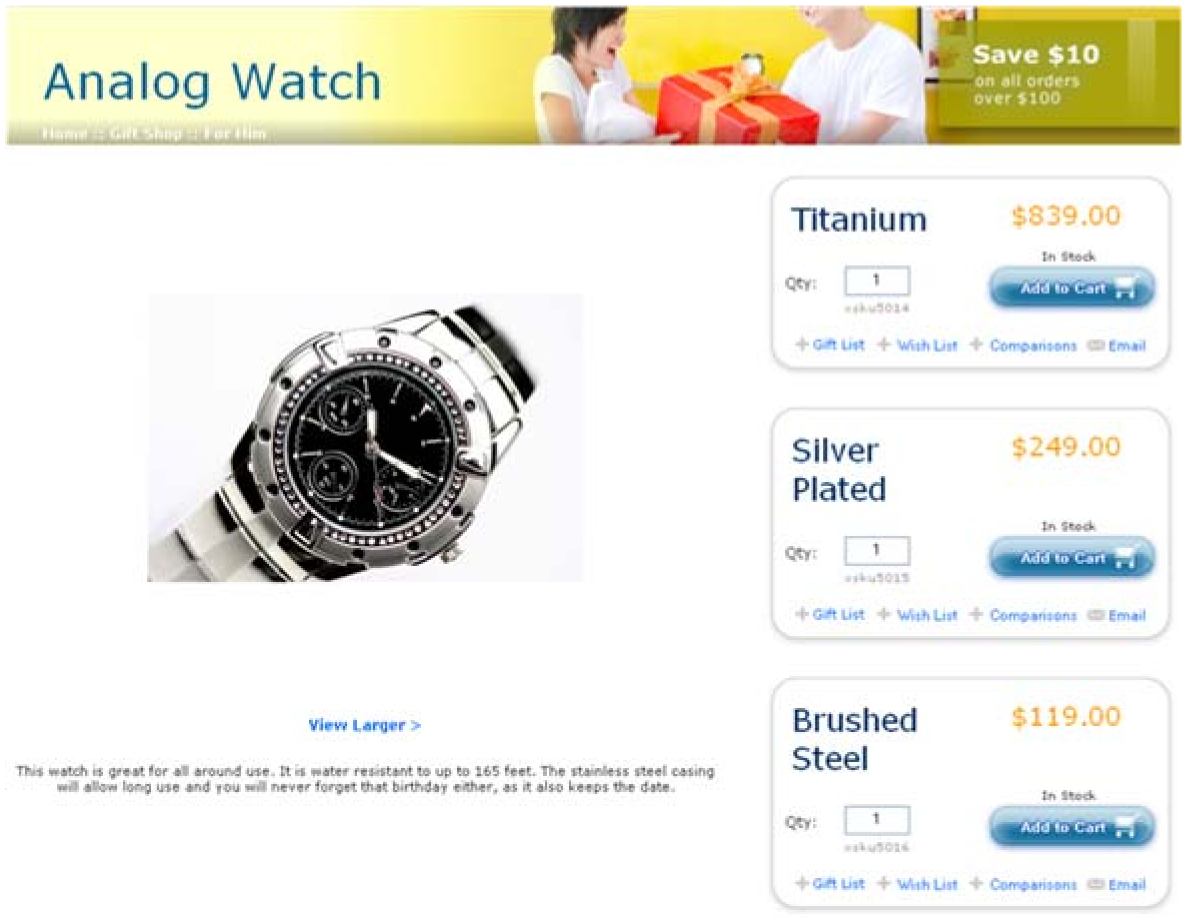
Product Detail Pages

The product detail page used for a specific product differs depending on the number and type of SKUs the product has. If the product has a single SKU, it can use the `productDetailSingleSku.jsp` template. This page displays information about the product, includes a quantity field for the SKU, an Add to Cart button, and links for adding the item to a gift list, e-mailing a friend, etc. For example:



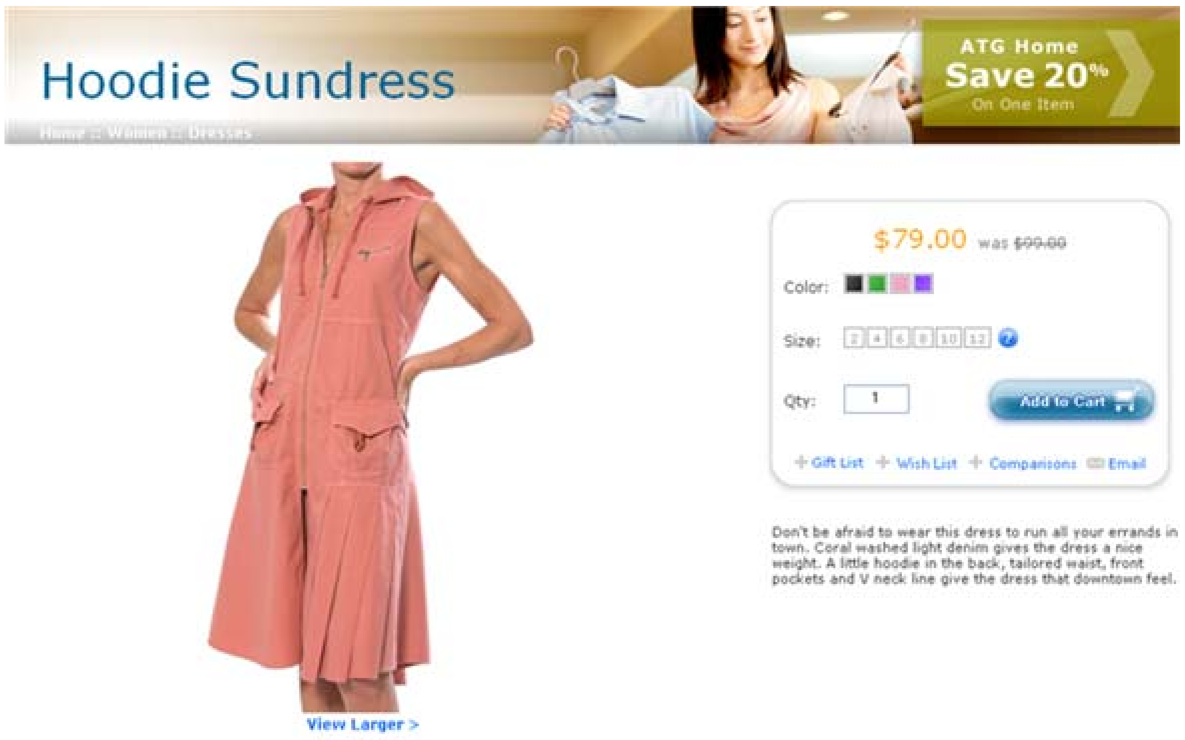
Our heavy glass tumblers are a versatile addition to your barware collection. Holds 12 ounces. Dishwasher safe. Made in Poland.

If a product has a small number of SKUs, it can use the `productDetailMultiSku.jsp` template. This page displays a separate quantity field for each SKU. The Analog Watch in the ATG Store Gift Shop -> For Him category illustrates this:



This watch is great for all around use. It is water resistant to up to 165 feet. The stainless steel casing will allow long use and you will never forget that birthday either, as it also keeps the date.

Most products with multiple SKUs use a template that includes a SKU picker. CRS has two such templates: `productDetailColorSizePicker.jsp`, which includes a color/size picker (for `clothing-sku` items), and `productDetailWoodFinishPicker.jsp`, which includes a wood finish picker (for `furniture-sku` items). For example, most clothing items on ATG Store include a color/size picker:



Don't be afraid to wear this dress to run all your errands in town. Coral washed light denim gives the dress a nice weight. A little hoodie in the back, tailored waist, front pockets and V neck line give the dress that downtown feel.

In most cases, templates that include a SKU picker are used only for products that have multiple SKUs. However, these templates can also be used for a product with a single SKU (e.g., a clothing item with only one `size` value and one `color` value). In this case, the picker has the single SKU preselected.

How the SKU Picker is Populated

To populate the SKU picker with colors and sizes, the `store.war/browse/gadgets/pickerContents.jsp` gadget invokes the `/atg/store/droplet/ColorSizeDroplet` servlet bean. This servlet bean examines all of the product's SKUs, and for each SKU creates a `ColorSizeDroplet.Color` object and a `ColorSizeDroplet.Size` object.

Each `ColorSizeDroplet.Color` object contains:

- the name of a color (taken from the SKU's `color` property)
- a `colorSwatch` media item (taken from the SKU's `colorSwatch` property)
- the availability status of the color, as determined by the Inventory Manager

Each `ColorSizeDroplet.Size` object contains:

- the name of a size (taken from the SKU's `size` property)
- the availability of the size, as determined by the Inventory Manager

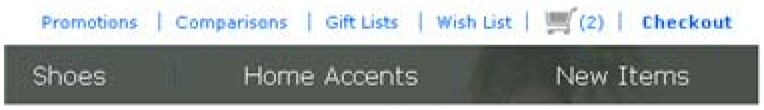
The `store.war/browse/gadgets/pickerColorPicker.jspf` page fragment uses the data in the `ColorSizeDroplet.Color` objects to render the color portion of the color/size picker. The `store.war/browse/gadgets/pickerSizePicker.jspf` page fragment uses the data in the `ColorSizeDroplet.Size` objects to render the size portion of the color/size picker.

Similar logic is used to populate the picker with wood finishes, using the `/atg/store/droplet/WoodFinishDroplet` servlet bean.

See the Inventory Management chapter for more information about availability statuses.

Product Lists

In addition to the product and catalog templates, CRS includes several pages that enable customers to compare products and maintain gift and wish lists. These pages are accessed through links that appear above the menu bar:


Promotions | Comparisons | Gift Lists | Wish List | (2) | Checkout
Shoes
Home Accents
New Items

Comparisons

Customers specify products for comparison by selecting the + Comparisons link on product detail pages. The items selected are tracked using the `/atg/commerce/catalog/comparison/ProductList` component, as described in the *ATG Commerce Programming Guide*. Clicking the Comparisons link displays the `store.war/browse/productComparisons.jsp` page, which renders a table with a column for each product. The rows contain information such as product features, available colors and sizes, etc.

Product comparisons are shared between ATG Store US and ATG Home, because the sharing of product comparisons is tied to the same shareable type component that manages shopping cart sharing. Each product name in a comparison list is a link to that product on the site it was added to the list from.

Gift Lists and Wish Lists

The CRS stores allow customers to create any number of gift lists, so that different lists can be created for different events. In addition, each customer has a single wish list. The products on the gift lists and the wish list are tracked in the `/atg/commerce/gifts/Giftlists` repository, as described in the *ATG Commerce Programming Guide*. A customer can add a product to any of these lists using options on the product detail page.

The same gift lists can be accessed on multiple sites. Gift list access is controlled by the `/atg/commerce/gifts/GiftlistManager` component's `siteScope` setting. By default, `GiftlistManager.siteScope` is set to `all`, so all gift lists can be accessed on any site. The CRS `Store.Estore` module changes this setting so that each gift list can be accessed only on sites that share a shopping cart with the site the gift list is associated with:

```
siteScope=atg.ShoppingCart
```

Because a customer can have only one wish list, it must be accessible from all sites. CRS uses the `/atg/registry/CollectionFilters/GiftlistSiteFilter` component to filter the items in a wish list to show only those items that are appropriate for the current site context.

Each product name in a gift or wish list is a link to that product on the site the customer was on when the product was added to the list.

See the *ATG Commerce Guide to Setting Up a Store* for more information about sharing gift and wish lists.

CRS Extensions

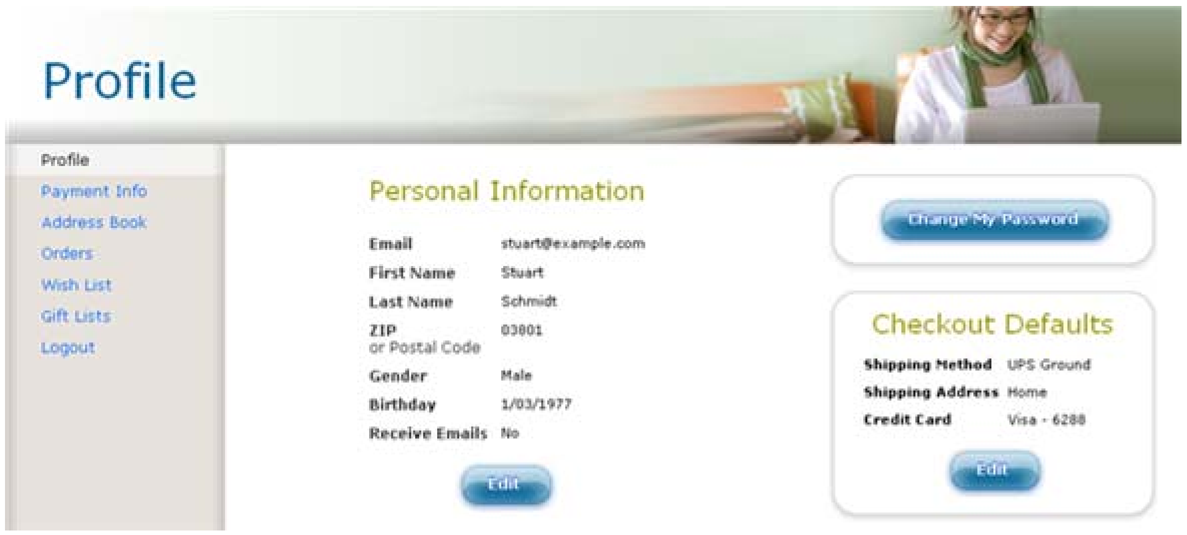
CRS uses standard ATG Commerce gift and wish list functionality, with the following extensions:

- CRS adds a derived property, `quantityRemaining`, to the `gift-item` item descriptor in the gift list repository. This property captures the quantity of a gift item that hasn't been purchased yet. It is defined in the `/atg/commerce/gifts/giftlists.xml` file in `<ATG10dir>/CommerceReferenceStore/Store/EStore/config/config.jar`. `quantityRemaining` is of type `atg.projects.store.gifts.GiftlistQuantityRemaining`, a CRS-specific class that extends `atg.repository.RepositoryPropertyDescriptor`. The `GiftlistQuantityRemaining` class has fields for quantity desired and quantity purchased, as well as a method for retrieving the remaining quantity.
- CRS changes the class of the `/atg/commerce/gifts/GiftlistFormHandler` component to `atg.projects.store.gifts.StoreGiftlistFormHandler`, which is a CRS-specific subclass of `atg.commerce.gifts.GiftlistFormHandler`. `StoreGiftlistFormHandler` includes functionality that determines if a customer is logged in before allowing the customer to add a gift to a gift list. This class also contains a number of methods that perform basic validation such as date validation, requiring gift quantities to be greater than zero, etc.

Account Pages

If a customer is not logged in or does not have an account, a Login link appears in the upper right of the page. Clicking the Login link displays the `store.war/myaccount/login.jsp` page, which has a form for creating a new account as well as a form for logging in to existing accounts.

If a customer is logged in, the Account link appears in the upper right, in place of the Login link. Clicking the Account link takes the customer to the `store.war/myaccount/profile.jsp` page. This page includes a navigation bar on the left, which is created by the `store.war/myaccount/gadgets/myAccountMenu.jsp` gadget:



Personal Information

Email	stuart@example.com
First Name	Stuart
Last Name	Schmidt
ZIP or Postal Code	03801
Gender	Male
Birthday	1/03/1977
Receive Emails	No

Checkout Defaults

Shipping Method	UPS Ground
Shipping Address	Home
Credit Card	Visa - 6288

The navigation bar is a set of links to various account management pages, including Gift Lists and Wish List (described above).

Email Templates

CRS has a number of JSPs that are used to dynamically construct e-mail messages that are sent out when customers perform certain actions. There are 6 top-level e-mail templates:

- `store.war/emailtemplates/abandonedOrderPromo.jsp` -- Message granting a promotion giving the customer 10% off his or her next order. This message is sent by a scenario that is triggered if the customer abandons a shopping cart. See the Scenarios That Grant Promotions section in the Personalization and Promotions chapter.
- `store.war/emailtemplates/backInStockNotify.jsp` -- Message informing a customer that a specific product or SKU is back in stock. This message is sent to customers who have explicitly requested notification, as described in the Back in Stock Emails section of the Inventory Management chapter.
- `store.war/emailtemplates/emailAFriend.jsp` -- Message with information about a specific product. A customer creates this message by clicking the Email link on the product detail page. CRS then displays a form that the customer fills out, specifying the recipient's email address and the message text.
- `store.war/emailtemplates/newPassword.jsp` -- Message containing a new temporary password for the customer's account. This message is sent when the customer clicks the "Forgot your password?" link on the Login page and fills in the email address associated with the account.
- `store.war/emailtemplates/orderConfirmation.jsp` -- Message confirming placement of a new order. The message is sent by a scenario that is triggered when the order is submitted. See the Scenarios that Confirm Orders and Shipment section in the Personalization and Promotions chapter.

- `store.war/emailtemplates/orderShipped.jsp` -- Message sent to a customer when an order has been shipped. The message is triggered by the Fulfillment system when the order's shipping group has been marked as shipped. See the Scenarios that Confirm Orders and Shipment section in the Personalization and Promotions chapter.

Like other top-level pages, these templates render much of their content through inclusion of gadgets and other JSP files. For example, `store.war/emailtemplates/backInStockNotify.jsp` includes `store.war/emailtemplates/gadgets/backInStockNotify.jsp`, which includes `store.war/emailtemplates/gadgets/backInStockSkuDetails.jsp`.

The e-mail templates take into account the site context, so messages can display site information, such as the large icon for the site they are associated with. To create links, the templates use the `/atg/dynamo/droplet/multisite/SiteLinkDroplet` servlet bean to render fully qualified URLs. Site configuration properties are used to set the e-mail sender addresses. For example, the sender address for `orderShipped.jsp` is taken from the site configuration's `orderShippedFromAddress` property.

Unlike other CRS pages, the e-mail JSPs contain inline styling information, rather than referring to external `.css` files. This is because some e-mail clients don't support external `.css` files. As a result, the styling for a given page is the same for all sites.

Cross-Site Links

CRS includes cross-site links to ATG Home products on the ATG Store site, and links to ATG Store products on the ATG Home site. Cross-site product links appear in the following places:

- Search results
- Recommended products on the home page
- Promotional content items that contain target links
- Related products on the product detail page
- Gift and wish lists
- Shopping cart
- Product comparisons

CRS uses the servlet beans `/atg/commerce/multisite/SiteIdForCatalogItem` (of class `atg.droplet.multisite.SiteIdForItemDroplet`) and `/atg/dynamo/droplet/multisite/SiteLinkDroplet` (of class `atg.droplet.multisite.SiteLinkDroplet`) to create cross-site links. The following example is adapted from `store.war/global/gadgets/crossSiteLinkGenerator.jsp`:

```
<%-- Pass a product repository item to the SiteIdForCatalogItem droplet to
determine the best-matching site ID for the product. --%>
<dsp:droplet name="/atg/commerce/multisite/SiteIdForCatalogItem">
   <dsp:param name="item" param="product" />
   <dsp:oparam name="output">
```

```
      <%-- Store the site ID obtained by SiteIdForCatalogItem in a productSiteId
      variable. --%>
      <dsp:getvalueof var="productSiteId" param="siteId" />

      <%-- Pass the site ID and the product's template path to SiteLinkDroplet to
      get a site-specific URL to the product template. --%>
      <dsp:droplet name="/atg/dynamo/droplet/multisite/SiteLinkDroplet">
         <dsp:param name="siteId" value="${productSiteId}" />
         <dsp:param name="path" param="product.template.url" />
         <dsp:oparam name="output">

            <%-- Render the generated URL to the product template, appended with
            the product ID. --%>
            <dsp:getvaluof var="siteLinkUrl" param="url" />
            <dsp:a href="${siteLinkUrl}">
               <dsp:param name="productId" param="product.repositoryId" />
               <dsp:valueof param="product.displayName" />
            </dsp:a>

         </dsp:oparam>
      </dsp:droplet>
   </dsp:oparam>
</dsp:droplet>
```

This example generates the URL as follows:

- The `SiteIdForCatalogItem` servlet bean determines the best site ID to use for the product.
- The `SiteLinkDroplet` servlet bean uses the `siteId` returned by `SiteIdForCatalogItem` to determine which site the URL should point to and it retrieves the value of the `productionURL` property for that site. For example, if the link is to a product on ATG Store US, the value of `productionURL` is `/crs/storeus`. Note that, in this example, `productionURL` does not contain domain information, so the URL that `SiteLinkDroplet` produces also does not include this information. Consequently, the browser will use the current page's protocol, domain, and port information for the link.
- The product template portion is taken from the product's `product.template.url` property, which is passed to `SiteLinkDroplet` in its `path` parameter.
- The `<dsp:a>` tag appends the `productId` to the URL as a query parameter.

The resulting URL looks similar to this:

```
/crs/homeus/browse/productDetailSingleSku.jsp?productId=xprod2126
```

For more information about the `SiteLinkDroplet` and `SiteIdForItemDroplet` servlet bean classes, see the *ATG Page Developer's Guide*.

4 Personalization and Promotions

This chapter discusses how ATG Commerce Reference Store customizes its content for different users. It includes the following topics:

CRS Profile Extensions

Because every ATG Commerce site has its own requirements for the information it must store for external users, most applications will need to extend the profile properties that are shipped with ATG Commerce. This section describes the custom properties and item descriptors that were added for CRS as well as changes to existing properties and item descriptors. The new and modified definitions are added to the profile repository by the `/atg/userprofiling/userProfile.xml` file in `<ATG10dir>/CommerceReferenceStore/Store/EStore/config/config.jar`.

Extensions to Profile Repository Item Descriptors

This section describes extensions to existing item descriptors in the profile repository.

Extensions to the User Item Descriptor

This section describes property additions and extensions for the `user` item descriptor.

age
The age property is a CRS-specific derived property calculated using the `dateOfBirth` property that is included in ATG Commerce. This property is used by the `ThirtySomethings` and `Young` user segments to identify customers of a particular age. See User Segments for more details.

autoLogin
CRS sets the `autoLogin` property default to `true`. This enables the ATG Commerce auto-login by profile feature, which allows registered customers to choose a “log me in automatically” option on your Web site. For more auto-login by profile information, see *Tracking Registered Users* in the *ATG Personalization Programming Guide*.

billingAddress
The `billingAddress` property stores a customer’s default billing address in an item

of type `contactInfo`. CRS extends the ATG Commerce definition of `billingAddress` with a modification to the `cascade` attribute. This attribute indicates that when changes are made to a `user` item, appropriate changes should also automatically be made to the corresponding `contactInfo` item that stores the customer's billing address. The `cascade` attribute is set to `cascade="update,delete"`. The `update` setting updates the `contactInfo` item as needed, while `delete` deletes the `contactInfo` item if the parent `user` item is deleted.

Note: For more information on the `cascade` attribute, see *Cascading Data Relationships* in the *ATG Repository Guide*.

categoryLastBrowsed
`categoryLastBrowsed` is a CRS-specific property that stores the last category a customer looked at. The `store.war/browse/gadgets/categoryLastBrowsed.jsp` gadget updates `categoryLastBrowsed` as a customer browses the ATG Store website. The `store.war/WEB-INF/tags/store/continueShopping.tag` uses `catgoryLastBrowsed` to generate a URL that returns a customer to the last category page he viewed. A number of JSP pages, such as `store.war/myaccount/gadgets/myWishListContinue.jsp`, implement the `continueShopping.tag`.

email
CRS uses the `email` property to identify a customer during login. For this reason, CRS extends the `email` property by marking it as required.

externalID
`externalID` is a CRS-specific property that provides a location to store old customer IDs. The `externalID` property allows you to import customers from an existing site, and maintain and track their original IDs.

firstName
CRS extends the `firstName` property by marking it as required.

gender
CRS extends the `gender` property by adding an `unknown` value for customers who have registered but have not provided a gender selection. Gender is not a required property in CRS.

itemsBought
`itemsBought` is a CRS-specific property in the form of a list that stores all the products a customer has purchased. `itemsBought` data is stored in a separate table, called `crs_items_bought`, with fields for user ID and SKU ID. Each time a customer submits an order in the `store.war/checkout/gadgets/confirmControls.jsp` gadget, the Nucleus component `/atg/store/order/purchase/CommitOrderFormHandler` updates the `crs_items_bought` table through a series of calls to two additional CRS-specific classes:

- `atg.projects.store.order.purchase.StoreCommitOrderHandler` is a subclass of `atg.commerce.order.purchase.CommitOrderFormHandler` that facilitates custom pre- and post-order commit functionality. `StoreCommitOrderHandler` calls `atg.projects.store.order.purchase.StoreCommitOrderProcessHelper`.

- `StoreCommitOrderProcessHelper` is a descendent of `atg.commerce.order.purchase.PurchaseProcessHelper` and it makes the updates to the `itemsBought` property.

lastName
CRS extends the `lastName` property by marking it as required.

lastPurchaseDate
`lastPurchaseDate` is a CRS-specific property that stores the date of a customer's last committed order. Each time a customer submits an order in the `store.war/checkout/gadgets/confirmControls.jsp` gadget, the Nucleus component `/atg/store/order/purchase/CommitOrderFormHandler` populates this property through a series of calls to two additional CRS-specific classes:

- `atg.projects.store.order.purchase.StoreCommitOrderHandler` is a subclass of `atg.commerce.order.purchase.CommitOrderFormHandler` that facilitates custom pre- and post-order commit functionality. `StoreCommitOrderHandler` calls `atg.projects.store.order.purchase.StoreCommitOrderProcessHelper`.
- `StoreCommitOrderProcessHelper` is a descendent of `atg.commerce.order.purchase.PurchaseProcessHelper`. It makes the update to the `lastPurchaseDate` property.

numberOfOrders
`numberOfOrders` is a CRS-specific property that tracks the number of orders a customer has placed. Each time a customer submits an order in the `store.war/checkout/gadgets/confirmControls.jsp` gadget, the Nucleus component `/atg/store/order/purchase/CommitOrderFormHandler` increments this property through a series of calls to two additional CRS-specific classes:

- `atg.projects.store.order.purchase.StoreCommitOrderHandler` is a subclass of `atg.commerce.order.purchase.CommitOrderFormHandler` that facilitates custom pre- and post-order commit functionality. `StoreCommitOrderHandler` calls `atg.projects.store.order.purchase.StoreCommitOrderProcessHelper`.
- `StoreCommitOrderProcessHelper` is a descendent of `atg.commerce.order.purchase.PurchaseProcessHelper` and it makes the update to the `numberOfOrders` property.

receivePromoEmail
`receivePromoEmail` is a CRS-specific property that allows the customer to opt in or out of marketing e-mails. The `store.war/myaccount/gadgets/profileMyInfoEdit.jsp` gadget calls the `/atg/store/profile/RegistrationFormHandler` component to set this property. `RegistrationFormHandler` is an instance of the `atg.projects.store.profile.StoreProfileFormHandler` class. The `StoreProfileFormHandler.updateReceiveEmailProperty()` method sets the `receivePromoEmail` property for the customer.

The `giveAbandonedOrderPromotion` scenario uses the `receivePromoEmail` property to determine whether or not it should send an e-mail to the customer, notifying her that she has been granted the 10% Off Order promotion that is intended to encourage order completion. See Scenarios That Grant Promotions for more information on this promotion.

referralSource
`referralSource` is a CRS-specific property that stores the selection a customer made from a list of referral sources while filling out the registration form in the `store.war/myaccount/gadgets/registrationForm.jsp` gadget.

CRS referral choices are enumerated in `userProfile.xml` and include the following (the display string that appears in ATG Store is provided in parentheses):

- `unknown` (Unknown)
- `tv` (TV)
- `radio` (Radio)
- `pressArticle` (Press Article)
- `onlineAdLink` (Online Ad or Link)
- `friend` (Friend)
- `justTypedURL` (Just typed in your URL)

The display strings that the storefronts use when rendering the referral sources are contained in the `/atg/projects/store/web/WebAppResources.properties` file in `<ATG10dir>/CommerceReferenceStore/Store/EStore/lib/classes.jar`.

Note: The display name for `referralSource` in the ATG Business Control Center is "Where did you hear about us."

secondaryAddresses
The `secondaryAddress` property is a map of all of the items of type `contactInfo` that are associated with the customer (including the default shipping and billing addresses also referenced by the `billingAddress` and `shippingAddress` properties). Each `contactInfo` item contains a single address associated with the customer.

shippingAddress
The `shippingAddress` property stores a customer's default shipping address in an item of type `contactInfo`. CRS extends the ATG Commerce definition of `shippingAddress` with a modification to the `cascade` attribute. In ATG Commerce, the `cascade` attribute is set to `cascade="insert,update"`. This setting causes a new, empty shipping address to be created for each newly registered user. Such empty addresses can cause issues when the customer selects a shipping address during checkout, however, so CRS changes the `cascade` attribute to `cascade=""`.

Note: For more information on the `cascade` attribute, see *Cascading Data Relationships* in the *ATG Repository Guide*.

Extensions to the credit-card Item Descriptor

CRS extends the `credit-card` item descriptor so that the credit card number is encrypted using the Triple DES encryption algorithm from the Sun JCE security provider. Out of the box, ATG Commerce doesn't apply any encryption to credit card information.

Custom Item Descriptor in the Profile Repository

CRS adds one new item descriptor to the profile repository, `backInStockNotifyItem`.

backInStockNotifyItem Item Descriptor

The `backInStockNotifyItem` item descriptor supports functionality in the `store.war/browse/gadgets/notifyMeRequest.jsp` gadget that allows customers to request notification when an item is back in stock. A `backInStockNotifyItem` creates a relationship between an e-mail address to be notified and the product and SKU ID for the product that is out of stock. When the `BackInStockNotification` service receives notification, generally from an external system, that a SKU is back in stock, it does the following:

- Queries the profile repository for `backInStockNotifyItem` items that have matching product and SKU IDs.
- Sends an e-mail to the addresses in the matching `backInStockNotifyItem` items.
- Deletes the matching `backInStockNotifyItem` items.

This section describes the `backInStockNotifyItem` item descriptor's properties.

Note: You can find additional information on back-in-stock notification in the Inventory Management chapter.

id
A unique identifier for the `backInStockNotifyItem`.

emailAddress
The e-mail address to which the back in stock notification will be sent.

catalogRefId
The SKU ID for the out-of-stock item.

locale
Stores the customer's locale when the back-in-stock notification request was made. This information ensures that the back-in-stock notification e-mail uses the same locale that was in effect when the request was made.

productId
The product ID for the out-of-stock item.

siteId
Stores the customer's site ID when the back-in-stock notification request was made. This information ensures that the back-in-stock notification e-mail uses the same site ID that was in effect when the request was made.

User Segments

User segments represent portions of your audience, allowing you to customize content for specific customers. You create user segments by defining rules that include or exclude customers according to their profile properties. User segments are viewed and managed through the ATG Business Control Center.

CRS has implemented user segments in two ways:

- Using a simplistic rule that manually assigns a sample customer to a user segment.
- Using a set of more realistic rules that dynamically place customers into segments based on demographic data.

For completeness, this section describes both implementations. However, ATG recommends that you create user segments dynamically, using demographic data. The more simplistic implementation is legacy code that is used for demonstration purposes only.

Simplistic User Segments

One of the sample customers included in CRS, the customer Kim, has had her `description` property populated with the string FASHIONISTA. A corresponding `Fashionista` user segment rule exists that follows this logic:

```
People whose {Description is FASHIONISTA}.
```

This rule places Kim in the `Fashionista` user segment. When Kim logs in, she is presented with content that is targeted at `Fashionista` customers only. Other sample customers will not see this content.

Dynamic User Segments

You create dynamic user segments by defining rules that include or exclude customers based on demographic characteristics. CRS ships with four dynamic user segments.

- ThirtySomethings Segment: This user segment is defined as customers whose property `age` is greater than or equal to 30 and less than 40.
- Women Only Segment: This user segment is defined as customers whose `gender` property is `female`.
- Men Only Segment: This user segment is defined as customers whose `gender` property is `male`.
- Young Segment: This user segment takes advantage of site-specific override rules to define `Young` customers differently on different sites. It defines a default rule and two site-specific override rules:
    - The default rule defines `Young` customers as people whose `age` is less than 35. ATG Store US uses the default rule.
    - For ATG Home, the default rule is overridden to specify that `Young` customers are people whose `age` is less than 40.
    - For ATG Store Germany, the override rule specifies that no customers fall into the `Young` segment.

User Segment Implicit Properties

At runtime, a customer's profile is appended with implicit properties that represent the user segments the customer belongs to, in the form of *userSegmentId*=[`true`|`false`]. These properties are not stored in the database, although the rules that are used to generate them are stored in the personalization repository.

User Segments in CRS JSP Pages

A number of CRS pages display different content depending on the segments a customer belongs to. These pages include:

- `store.war/atgsearch/atgSearchResultsContainer.jsp`
- `store.war/browse/categoryDisplay.jsp`
- `store.war/browse/subcategoryDisplay.jsp`
- `store.war/navigation/gadgets/homePromotions.jsp`
- `store.war/navigation/gadgets/homePromotionalItems.jsp`
- `store.war/promo/gadgets/homeFeaturedProducts.jsp`
- `store.war/search/searchResults.jsp`

All of these pages implement content targeters that define different content for different user segments. See Using Targeters, Slots, and Scenarios to Personalize Content for more details.

For More Information

For more detailed information on user segments and content targeters, see the *ATG Personalization Programming Guide* and the *ATG Personalization Guide for Business Users*.

Personalizing Content for Customers

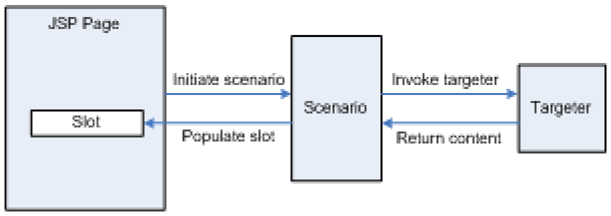
ATG Commerce has four features that allow you to personalize content for your customers: targeters, slots, scenarios, and promotions. CRS implements all of these features, as described below.

Using Targeters, Slots, and Scenarios to Personalize Content

CRS uses a common approach to personalizing its content that involves a combination of targeters, slots, and scenarios. Targeters determine which content to show to individual customers. Slots function as placeholders in JSP pages where customized content can be stored. Scenarios create the link between slots and targeters in the following way:

- When a customer accesses a JSP page that has a slot, the slot requests content.
- The slot's content request initiates a scenario.
- The scenario contains rules that tell it which targeter to run.

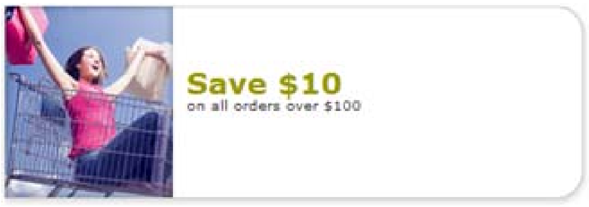
- The targeter determines the appropriate content for the customer and returns it to the scenario.
- The scenario returns the content to the JSP page and the slot is populated. Note that slots only store content. Other logic must exist on the JSP page to render the content:



The following section describes the targeters, slots, and scenarios implemented for CRS.

CRS Targeters

Each targeter returns either a product ID or a `promotionalContent` item. A `promotionalContent` item is a CRS-specific item type that provides information about a promotion or marketing campaign on ATG Store. Essentially, a `promotionalContent` item is an image, sometimes with supporting text, although the item type itself has a number of other useful properties. Note that `promotionalContent` items are informational only. The actual granting of promotions is handled by other mechanisms. This illustration shows two examples of `promotionalContent` items:



Hubbard Chairs
Save 10% on 2
Save 15% on 4
Save 20% on 6

Targeters are viewed and managed through the ATG Business Control Center. This section describes the CRS targeters, broken down into targeters that return product IDs and targeters that return `promotionalContent` items.

Notes:

- See promotionalContent Item Descriptor in the Product Catalog chapter for more information on the `promotionalContent` item type.
- See CRS Promotions for more information about granting promotions in CRS.
- CRS no longer uses the `CurrentPromotions` and `GlobalBestSellers` targeters.

Targeters That Return Product IDs

The following targeters return product IDs.

GiftWrapItem

The `GiftWrapItem` targeter is an exception to the slot/scenario/targeter workflow described in Using Targeters, Slots, and Scenarios to Personalize Content. The `GiftWrapItem` targeter is called directly by the `store.war/cart/gadgets/giftWrap.jsp` page and it returns the product ID for the gift wrap item from the product catalog. This functionality supports the Gift Wrap feature that appears on the CRS shopping cart page.

Encapsulating the gift wrap product ID in a targeter provides two benefits:

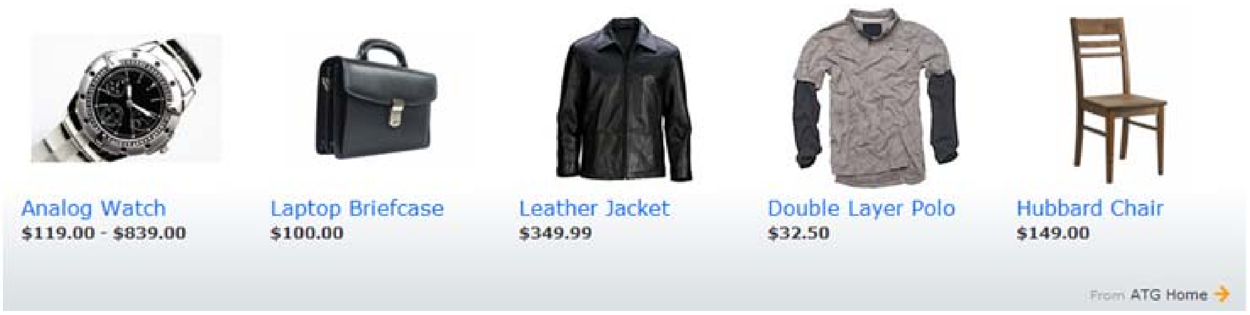
- It gives merchandisers access to the gift wrap item, through the ATG Business Control Center, so that they can change it to point to a different product ID, if necessary. Because the gift wrap item is not part of the regular catalog hierarchy, the targeter is the only location where merchandisers can makes changes to the gift wrap item.
- Changes to the gift wrap item's product ID do not require changes to the JSP code.

For more information on CRS gift wrap features, see Gift Wrap in the Order Handling chapter.

HomeFeaturedProduct1, HomeFeaturedProduct2, HomeFeaturedProduct3, HomeFeaturedProduct4, and HomeFeaturedProduct5

The `HomeFeaturedProduct1`, `HomeFeaturedProduct2`, `HomeFeaturedProduct3`, `HomeFeaturedProduct4`, and `HomeFeaturedProduct5` targeters each return a single product, based on user segment and site context. Each of these targeters defines a default rule set, which is used by ATG Store US, and override rule sets for the ATG Home and ATG Store Germany sites.

The `HomeFeaturedProducts` scenario uses these targeters to populate the recommended products section of the home page. For example, the following products are shown to the Men Only user segment on the ATG Store US site:



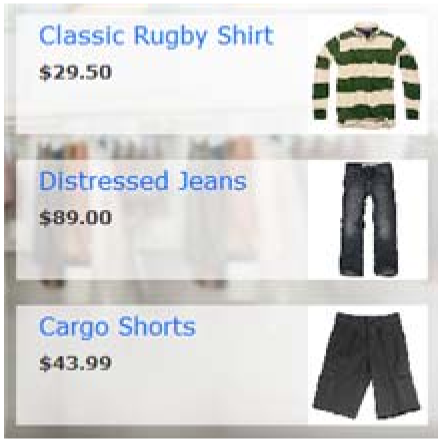
HomePromotionalItem1, HomePromotionalItem2, and HomePromotionalItem3

The `HomePromtionalItem1`, `HomePromotionalItem2`, and `HomePromotionalItem3` targeters return different product IDs, based on site context and user segment. For example, when the current site is ATG Store US, `HomePromotionalItem1` returns the following product IDs:

- If the customer is in the `Fashionista` user segment, return the Pearl Necklace.

- If the customer is in the womenOnly user segment, return the Elegant Sandal.
- If the customer is in the menOnly user segment, return the Classic Rugby Shirt.
- For all other customers, return the Cowgirl Bag.

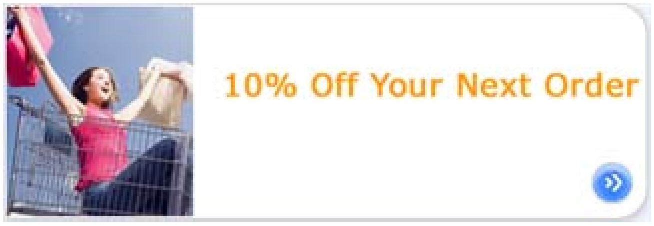
All of these targeters define a default rule set and an override rule set for the ATG Store US and ATG Store Germany sites. The homePromotionalItems scenario uses the HomePromtionalItem1, HomePromotionalItem2, and HomePromotionalItem3 targeters to populate slots on the home page with product images that are customized based on user segment. For example, the following image shows the three products that are returned for the Men Only user segment on ATG Store US:



Targeters That Return promotionalContent Items

The following targeters return promotionalContent items.

AbandonedOrderPromotion
The AbandonedOrderPromotion targeter returns the 10% off your next order promotional content item to all customers in all site contexts. The store.war/emailtemplates/abandonedOrderPromo.jsp e-mail template calls this targeter directly when it constructs the e-mail that is sent to customers, notifying them that they have been granted the abandoned order promotion.

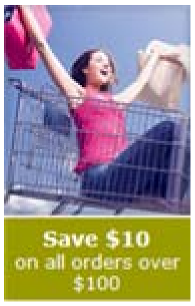


CategoryPromotionContent1, CategoryPromotionContent2
The CategoryPromotionContent1 and CategoryPromotionContent2 targeters return different promotionalContent items, based on site context and the

customer's current location in the category tree. For example, when the current site is ATG Home, `CategoryPromotionContent1` returns the following:

- If the customer is viewing the Gift Shop, Tables, or Home Accents category, show the Save $10 on orders over $100 `promotionalContent` item.
- If the customer is viewing the Seating category, show the Free Shipping on ATG Home `promotionalContent` item.
- If the customer is viewing the Home Storage category, show the ATG Home Save 20% On One Item `promotionalContent` item.

Both targeters define a default rule set, which is used by ATG Store US, and override rule sets for the ATG Home and ATG Store Germany sites. The `categoryPromotions` scenario uses the `CategoryPromotionContent1` and `CategoryPromotionContent2` targeters to populate slots on category and search result pages with media informing the customer of various promotions, for example:



HomeTheme

The `HomeTheme` targeter returns a `promotionalContent` item that contains a splash image for the home page.

The `HomeTheme` targeter defines a default rule set that returns a generic splash screen. This default rule set gets used by any newly added sites, thereby ensuring that a newly added site will always have a splash screen. The `HomeTheme` targeter also includes two override rule sets, one for ATG Store US and ATG Store Germany and another for ATG Home. These overrides define different splash screens based on site context and user segment. For example, the following is the home page image for the Men Only user segment on ATG Store US:

The `Theme` scenario uses the `HomeTheme` targeter to populate the splash image on the home page.

CRS Slots

You must view CRS slots using the ATG Control Center (ACC). Slots are not viewable in the ATG Business Control Center (BCC).

To view CRS slots:

1. Start up CRS and the ATG Control Center. For details on starting CRS, see the *ATG Commerce Reference Store Installation and Configuration Guide*. For details on starting the ATG Control Center, see *ATG Personalization Guide for Business Users*.
2. From the navigation menu, select Scenarios, then select Slots.

CRS slots are configured so that, when a customer views a page that contains a slot, the slot requests content, thereby initiating a scenario. In the ATG Control Center, this behavior is controlled by setting the Event Generation option to When Empty (that is, the slot is empty and requires content). Note that this is not the default configuration for slots. For more details, see *Using Slots* in the *ATG Personalization Programming Guide*.

Because slots are intimately tied to scenarios, we refrain from discussing them as individual entities here. For details on a given slot, refer to its related scenario, provided in the table below.

Slot	Related Scenario
CategoryPromotionContent1	categoryPromotions
CategoryPromotionContent2	categoryPromotions
HomeFeaturedProduct1	HomeFeaturedProducts
HomeFeaturedProduct2	HomeFeaturedProducts
HomeFeaturedProduct3	HomeFeaturedProducts

HomeFeaturedProduct4	HomeFeaturedProducts
HomeFeaturedProduct5	HomeFeaturedProducts
HomePromotionalItem1	HomePromotionalItems
HomePromotionalItem2	HomePromotionalItems
HomePromotionalItem3	HomePromotionalItems
HomeTheme	Theme
RelatedItemsOfCart	RelatedItemsSlot

Note: CRS no longer uses the `CurrentPromotions` and `GlobalBestSellers` slots.

CRS Scenarios

A scenario is a choreographed series of interactions with your Web site visitors. In a scenario, you anticipate and track the actions of the people who visit your Web site, and respond appropriately by, for example, tailoring the content of the Web site, offering price promotions, or sending targeted e-mail messages. You must view CRS scenarios using the ATG Control Center. Scenarios are not viewable in the ATG Business Control Center.

To view CRS scenarios:

1. Start up CRS and the ATG Control Center. For details on starting CRS, see the *ATG Commerce Reference Store Installation and Configuration Guide*. For details on starting the ATG Control Center, see *ATG Personalization Guide for Business Users*.
2. From the navigation menu, select Scenarios.
3. Make sure By Folders is selected from the filtering menu.
4. Expand `store`. CRS scenarios are stored under the `/store` directory.
5. Click a `/store` subdirectory, then select a scenario to view it in the Scenarios editor.

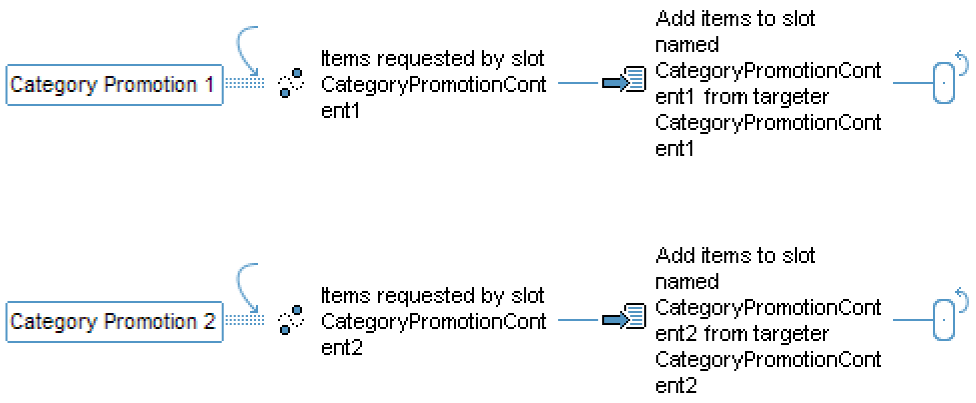
This section describes the scenarios implemented in CRS. These include scenarios that provide content, scenarios that grant promotions, and scenarios that confirm order placement and shipping.

Note: CRS no longer uses the `CurrentPromotions` and `BestSellers` scenarios.

Scenarios That Provide Content

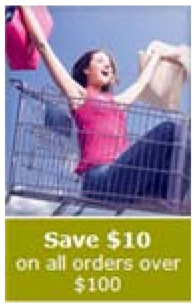
The scenarios described in this section provide content to JSP pages and use the slot/scenario/targeter mechanism described in Using Targeters, Slots, and Scenarios to Personalize Content.

categoryPromotions
The `categoryPromotions` scenario, located in `/store/category`, looks like this:


Category Promotion 1
Items requested by slot CategoryPromotionContent1
Add items to slot named CategoryPromotionContent1 from targeter CategoryPromotionContent1
Category Promotion 2
Items requested by slot CategoryPromotionContent2
Add items to slot named CategoryPromotionContent2 from targeter CategoryPromotionContent2

The `categoryPromotions` scenario has two segments. Both segments perform identical functions in that they provide a slot on a JSP page with a `promotionalContent` item, but they use different targeters to determine which item to return.

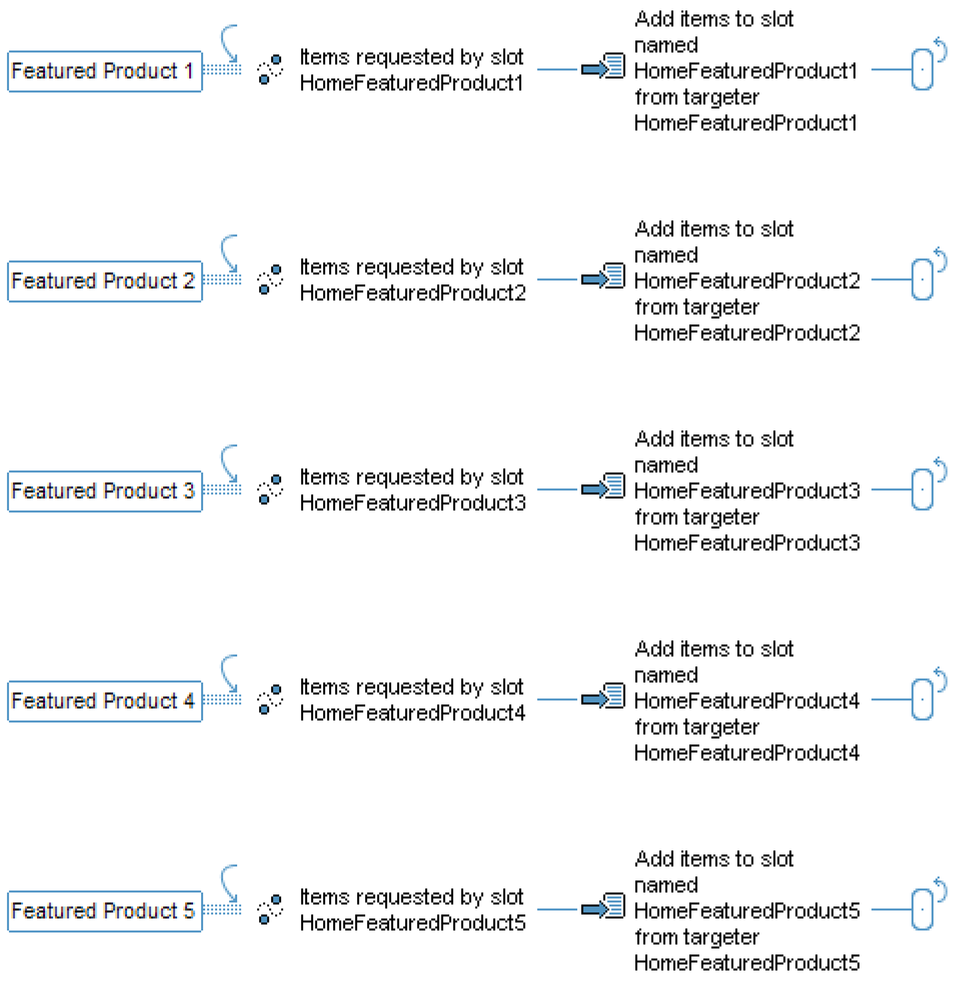
The first segment is invoked when the `CategoryPromotionContent1` slot requests content. This slot appears in several places in CRS, such as `store.war/atgsearch/atgSearchResultsContainer.jsp` and `store.war/browse/categoryDisplay.jsp`. To supply the slot with a `promotionalContent` item, the `categoryPromotions` scenario calls the `CategoryPromotionContent1` targeter. This targeter returns different `promotionalContent` items, based on site context and the customer's current location in the category tree. For example, when the current site is ATG Home and the customer is viewing the Gift Shop, Tables, or Home Accents category, `CategoryPromotionContent1` returns the Save $10 on orders over $100 `promotionalContent` item.



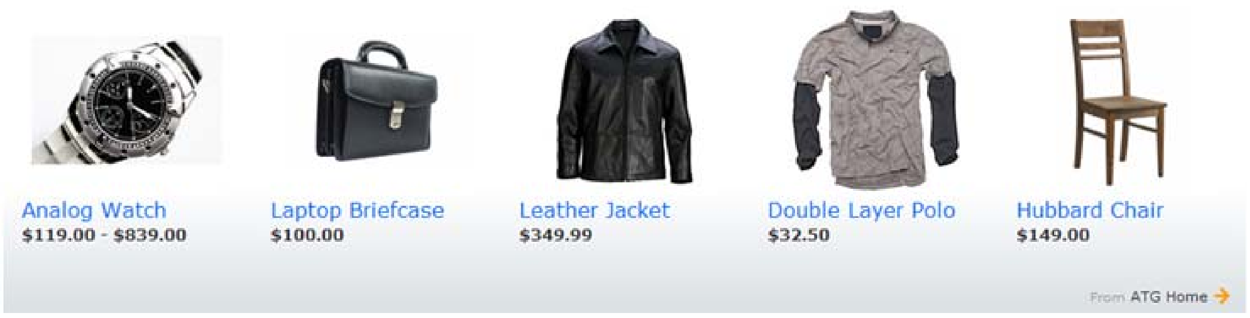
The second segment is invoked when the `CategoryPromotionContent2` slot requests content and it uses the `CategoryPromotionContent2` targeter to determine which `promotionalContent` item to return.

HomeFeaturedProducts

The `HomeFeaturedProducts` scenario, located in `/store/homepage`, looks like this:


Featured Product 1
Items requested by slot HomeFeaturedProduct1
Add items to slot named HomeFeaturedProduct1 from targeter HomeFeaturedProduct1
Featured Product 2
Items requested by slot HomeFeaturedProduct2
Add items to slot named HomeFeaturedProduct2 from targeter HomeFeaturedProduct2
Featured Product 3
Items requested by slot HomeFeaturedProduct3
Add items to slot named HomeFeaturedProduct3 from targeter HomeFeaturedProduct3
Featured Product 4
Items requested by slot HomeFeaturedProduct4
Add items to slot named HomeFeaturedProduct4 from targeter HomeFeaturedProduct4
Featured Product 5
Items requested by slot HomeFeaturedProduct5
Add items to slot named HomeFeaturedProduct5 from targeter HomeFeaturedProduct5

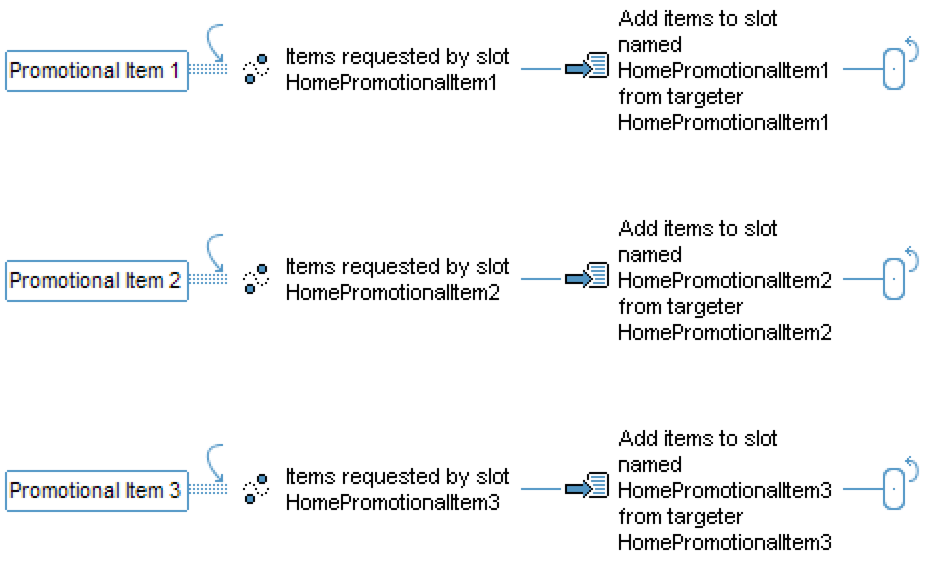
The `HomeFeaturedProducts` scenario has five segments, which all follow the same pattern: a slot initiates a segment by requesting content and the scenario invokes a targeter to provide that content. All of the targeters return a static product ID, based on user segment and site context. For example, the following products are shown to the Men Only user segment on the ATG Store US site:



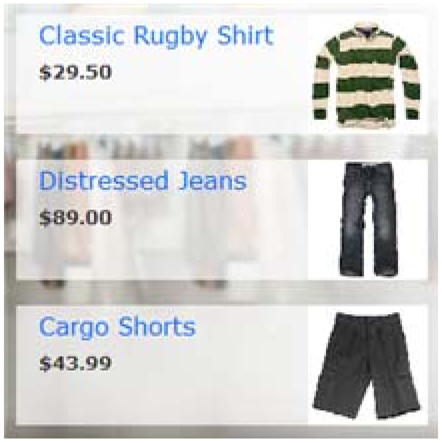
The first segment of the `HomeFeaturedProducts` scenario is initiated by the `HomeFeaturedProduct1` slot, the second is initiated by the `HomeFeaturedProduct2` slot, the third is initiated by the `HomeFeaturedProduct3` slot, and so on. All five slots appear in the `store.war/promo/gadgets/homeFeaturedProducts.jsp` gadget. Content for each slot is provided by the `HomeFeaturedProduct1`, `HomeFeaturedProduct2`, `HomeFeaturedProduct3`, `HomeFeaturedProduct4`, and `HomeFeaturedProduct5` targeters, respectively.

HomePromotionalItems

The `HomePromotionalItems` scenario, located in `/store/homepage`, looks like this:

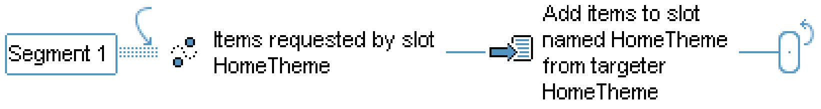

Promotional Item 1
Items requested by slot HomePromotionalItem1
Add items to slot named HomePromotionalItem1 from targeter HomePromotionalItem1
Promotional Item 2
Items requested by slot HomePromotionalItem2
Add items to slot named HomePromotionalItem2 from targeter HomePromotionalItem2
Promotional Item 3
Items requested by slot HomePromotionalItem3
Add items to slot named HomePromotionalItem3 from targeter HomePromotionalItem3

The `HomePromotionalItems` scenario has three segments, which all follow the same pattern: a slot initiates a segment by requesting content and the scenario invokes a targeter to provide that content. All of the targeters return a product ID, based on user segment and site context. For example, the following image shows the three products that are returned for the Men Only user segment on ATG Store US:



The first segment of the `HomePromotionalItems` scenario is initiated by the `HomePromotionalItem1` slot, the second is initiated by the `HomePromotionalItem2` slot, and the third is initiated by the `HomePromotionalItem3` slot. All three slots appear in the `store.war/navigation/gadgets/homePromotionalItems.jsp` gadget. Content for each slot is provided by the `HomePromotionalItem1`, `HomePromotionalItem2`, and `HomePromotionalItem3` targeters, respectively.

Theme
The `Theme` scenario, located in `/store/homepage`, looks like this:


Segment 1
Items requested by slot HomeTheme
Add items to slot named HomeTheme from targeter HomeTheme

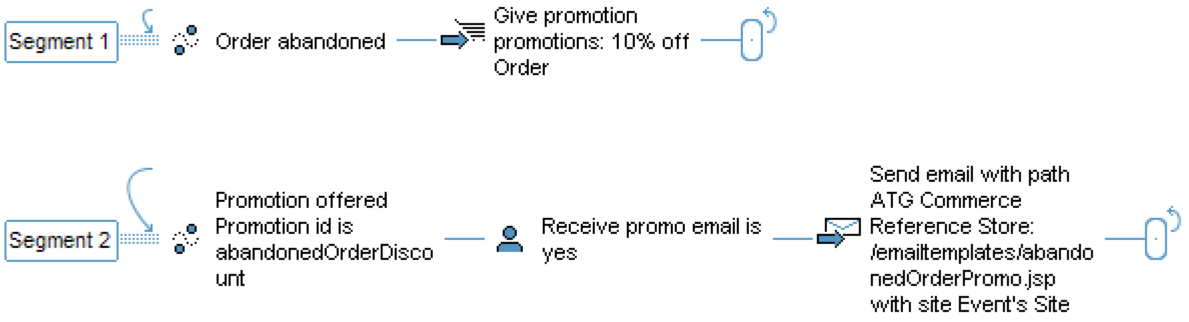
The `Theme` scenario is initiated when the `HomeTheme` slot requests content. This slot appears in the `store.war/navigation/gadgets/homePromotions.jsp` gadget and provides the splash image for the home page. To supply the slot with content, the `Theme` scenario calls the `HomeTheme` targeter. The `HomeTheme` targeter returns a `promotionalContent` item, based on user segment and site context, that contains the splash image. For example, the following is the home page image for the Men Only user segment on ATG Store US:

Scenarios That Grant Promotions

The scenarios described in this section grant promotions to customers.

Note: For detailed information on the promotions themselves, see CRS Promotions.

giveAbandonedOrderPromotion
The `giveAbandonedOrderPromotion` scenario, located in `/store/abandonedorders`, looks like this:


Segment 1
Order abandoned
Give promotion promotions: 10% off Order
Segment 2
Promotion offered Promotion id is abandonedOrderDiscount
Receive promo email is yes
Send email with path ATG Commerce Reference Store: /emailtemplates/abandonedOrderPromo.jsp with site Event's Site

The `giveAbandonedOrderPromotion` scenario encourages customers to return and complete abandoned orders. `giveAbandonedOrderPromotion` has two segments. Segment 1 is responsible for granting the 10% off Order promotion to a customer when the customer has an abandoned order.

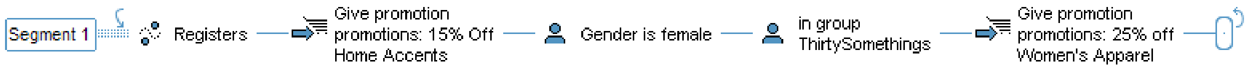
The ATG Commerce `AbandonedOrderService` checks the order repository for orders that have been started but not checked out. You configure when and how often the `AbandonedOrderService` runs, as well as the age of the orders you want it to find. For example, you can set the service to find incomplete orders whose last modified date is at least a week old. When the `AbandonedOrderService` identifies an order as abandoned, segment 1 is invoked and the customer is granted the 10% off Order promotion. *Granting* a promotion to a customer adds the promotion to the customer's `activePromotions` profile property. The `AbandonedOrderService` also marks the order as abandoned, so it won't find the order again the next time the service runs.

Segment 2 is responsible for notifying customers of the promotion they have been granted. When the 10% off Order promotion has been added to a customer's `activePromotions` property, segment 2 is invoked. This segment determines if the customer's `receivePromoEmail` profile property is set to Yes (code 1) and, if so, it sends an e-mail with details about the promotion to the customer. The e-mail is sent using the `store.war/emailtemplates/abandonedOrderPromo.jsp` template. The `giveAbandonedOrderPromotion` scenario passes the abandoned order's site ID to `abandonedOrderPromo.jsp`, so that the e-mail it generates is site-specific.

Note that, because this scenario relies on a customer's profile, it only applies to registered customers.

For detailed information on the `AbandonedOrderService`, see *Using Abandoned Order Services* in the *ATG Commerce Programming Guide*. For detailed information on the `activePromotions` property, see the *ATG Commerce Programming Guide*.

GrantPromotions
The `GrantPromotions` scenario, located in `/store/global`, looks like this:


Segment 1
Registers
Give promotion promotions: 15% Off Home Accents
Gender is female
in group ThirtySomethings
Give promotion promotions: 25% off Women's Apparel

The `GrantPromotions` scenario grants a promotion to a newly registered customer. This scenario listens for the `atg.dps.Register` event message that is generated each time a customer registers. When it detects a `Register` message, the scenario does the following:

- Grants the ATG Store – 15% Off Home Accents promotion to the customer.
- If the customer is female and belongs to the `ThirtySomethings` user segment, the scenario also grants the 25% Off Women's Apparel promotion to the customer.

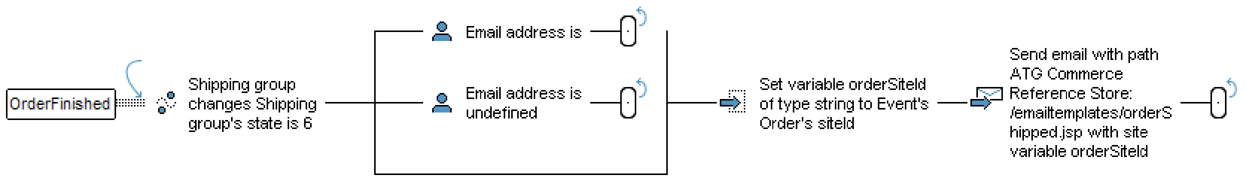
Granting a promotion to a customer adds the promotion to the customer's profile in the `activePromotions` property.

Note: For more information on the `atg.dps.Register` event message, see *Using Scenario Events* in the *ATG Personalization Programming Guide*.

Scenarios That Confirm Orders and Shipment

This section describes scenarios that confirm order placement and shipment.

OrderShipped
The `OrderShipped` scenario, located in `/store/orders`, looks like this:


OrderFinished
Shipping group changes Shipping group's state is 6
Email address is
Email address is undefined
Set variable orderSiteId of type string to Event's Order's siteId
Send email with path ATG Commerce Reference Store: /emailtemplates/orderShipped.jsp with site variable orderSiteId

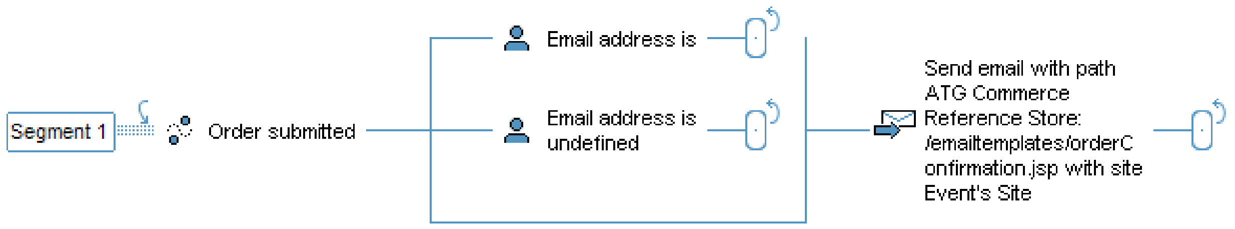
The `OrderShipped` scenario sends a confirmation e-mail after a customer's order has been shipped. This scenario listens for the `atg.commerce.fulfillment.scenario.ShippingGroupModified` event message that indicates that a shipping group has been shipped. These messages are sent by the `/atg/commerce/fulfillment/OrderChangeHandler` component. When the `OrderShipped` scenario detects a `ShippingGroupModified` event message that indicates a shipping group has been shipped, it does one of the following:

- If the customer's e-mail address is blank, the scenario ends without sending e-mail.
- If the customer's e-mail address is undefined, the scenario ends without sending e-mail.
- For all other cases, `OrderShipped` sends a confirmation e-mail to the address stored in the customer's profile, using the `store.war/emailtemplates/orderShipped.jsp` template. The `OrderShipped` scenario passes the order's site ID to `orderShipped.jsp`, so that the e-mail it generates is site-specific.

Note: For more information on the `ShippingGroupModified` event message, see the *ATG Commerce Guide to Setting Up a Store*. For more information on the `OrderChangeHandler` component, see the *ATG Commerce Programming Guide*.

ordersubmit

The `ordersubmit` scenario, located in `/store/orders`, confirms a customer's order. It looks like this:


Segment 1
Order submitted
Email address is
Email address is undefined
Send email with path ATG Commerce Reference Store: /emailtemplates/orderConfirmation.jsp with site Event's Site

The `ordersubmit` scenario sends a confirmation e-mail after a customer has submitted an order. This scenario listens for the `atg.commerce.fullfillment.SubmitOrder` event message that is generated each time a customer submits an order. When it detects a `SubmitOrder` message, the scenario does one of the following:

- If the customer's e-mail address is blank, the scenario ends without sending e-mail.
- If the customer's e-mail address is undefined, the scenario ends without sending e-mail.

- For all other cases, `ordersubmit` sends a confirmation e-mail to the address stored in the customer's profile, using the `store.war/emailtemplates/orderConfirmation.jsp` template. The `ordersubmit` scenario passes the order's site ID to `orderConfirmation.jsp`, so that the e-mail it generates is site-specific. If the order is split over multiple shipping groups, an e-mail is sent for each shipping group.

CRS Promotions

Promotions are repository items, stored in the `catalog` repository. The table below describes the promotions that ship with CRS.

Promotion	Description	Details
$10 Off Orders Over $100	A discount of $10 is applied on orders over $100.	**Condition and Offer Type:** Spend Y Get Order Discount **Granted to:** Everyone **Sites:** All sites
10% off Order	A discount of 10% is applied to the entire order.	**Condition and Offer Type:** Get Order Discount **Granted to:** Customers with an abandoned order. **Sites:** All sites
10LIEFERUNG Coupon – 10% Off Order	A discount of 10% is applied to the entire order when the customer enters the 10LIEFERUNG coupon code.	**Condition and Offer Type:** Get Order Discount **Granted to:** Customers who enter the 10LIEFERUNG coupon code **Sites:** ATG Store Germany
10LIEFERUNG Coupon – Free Shipping	A shipping discount, for free shipping, is applied to the entire order when the customer enters the 10LIEFERUNG coupon code.	**Condition and Offer Type:** Get Shipping Discount **Granted to:** Customers who enter the 10LIEFERUNG coupon code **Sites:** ATG Store Germany
15% Off Home Accents	Products in the Home Accents category are discounted 15%. If the product is a member of multiple sites, it will still qualify for the discount, regardless of the site it was added from.	**Condition and Offer Type:** Get Item Discount **Granted to:** Newly registered customers. **Sites:** All sites

Promotion	Description	Details
25% Off Women's Apparel	Products in the Women's Apparel category are discounted 25% for female customers who are part of the `ThirtySomething` user segment.	**Condition and Offer Type:** Advanced Item Promotion **Granted to:** Newly registered customers who are female and part of the `ThirtySomething` user segment **Sites:** All sites
ATG Home – Free Shipping	When the customer checks out on ATG Home, they receive free shipping. This promotion raises customer awareness of the ATG Home site.	**Condition and Offer Type:** Get Shipping Discount **Granted to:** Everyone **Sites:** ATG Home
ATG Home Save 20% On One Item	A discount of 20% is applied to the highest priced SKU in the customer's shopping cart that has a `SKU.sites` property that contains `homeSite` (the ID for ATG Home). If the SKU is a member of multiple sites, it will still qualify for the discount, regardless of the site it was added from.	**Condition and Offer Type:** Get Item Discount **Granted to:** Everyone **Sites:** ATG Home, ATG Store US
Hubbard Chair Tiered Price Break	A discount is applied based on the number of chairs the customer purchases. Purchasing two chairs results in a 10% discount, four chairs results in a 15% discount, and six chairs results in a 20% discount.	**Condition and Offer Type:** Tiered Price Break **Granted to:** Everyone **Sites:** ATG Home, ATG Store US
TENSHIP Coupon – 10% Off Order	A discount of 10% is applied to the entire order when the customer enters the TENSHIP coupon code.	**Condition and Offer Type:** Get Order Discount **Granted to:** Customers who enter the TENSHIP coupon code **Sites:** ATG Home, ATG Store US
TENSHIP Coupon – Free Shipping	A shipping discount for free shipping is applied to the entire order when the customer enters the TENSHIP coupon code.	**Condition and Offer Type:** Get Shipping Discount **Granted to:** Customers who enter the TENSHIP coupon code **Sites:** ATG Home, ATG Store US

Granting CRS Promotions

When defining a promotion, you have the option of configuring it so that it is available to everyone on the Availability & Coupons tab. When you choose Available to everyone, the promotion is automatically granted to all customers, making the promotion a global promotion. When this setting is not chosen, promotions must be explicitly granted to customers through the use of scenarios or coupons. The following table lists the granting mechanism for CRS promotions.

Promotion	**Granting Mechanism**	**Promotion Availability Setting**
$10 Off Orders Over $100 ATG Home – Free Shipping ATG Home Save 20% On One Item Hubbard Chair Tiered Price Break	Automatically granted to all customers.	Available to everyone
10LIEFERUNG Coupon – 10% Off Order 10LIEFERUNG Coupon – Free Shipping	Both promotions are granted after the customer enters the 10LIEFERUNG Coupon code.	Requires customer action (any user)
TENSHIP Coupon – 10% Off Order TENSHIP Coupon – Free Shipping	Both promotions are granted after the customer enters the TENSHIP Coupon code.	Requires customer action (any user)
10% off Order	Granted by the `giveAbandonedOrderPromotion` scenario when a customer has an abandoned order.	Requires customer action (any user)
15% Off Home Accents	Granted by the `GrantPromotions` scenario when a new customer registers.	Requires customer action (registered users only)
25% Off Women's Apparel	Granted by the `GrantPromotions` scenario when a new customer registers that is female and part of the `ThirtySomethings` user segment.	Requires customer action (registered users only)

Granting a promotion adds the promotion to the customer's profile in the `activePromotions` property, after which the customer may be notified of the promotion.

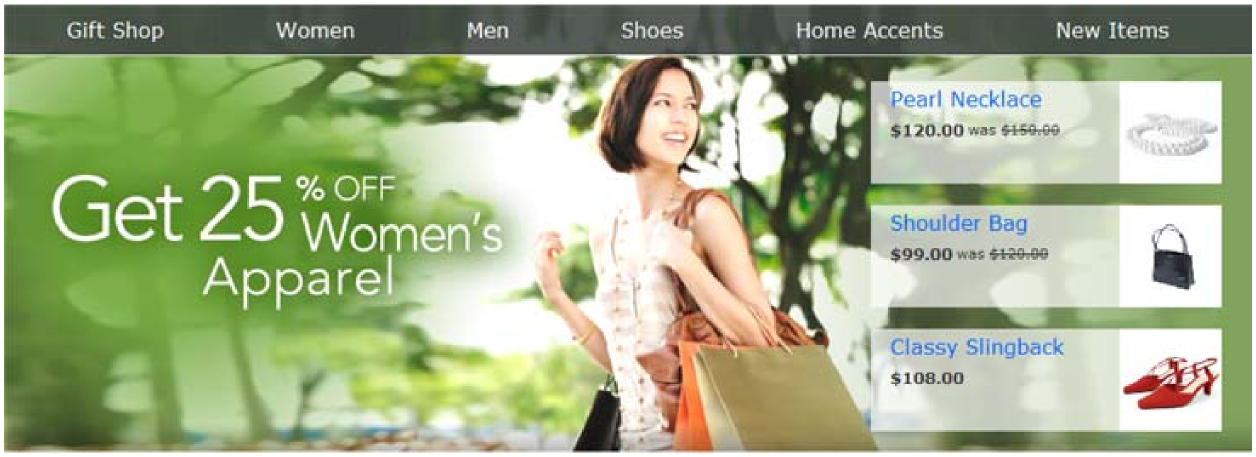
Note: For detailed information on the `activePromotions` property, see the *ATG Commerce Programming Guide*.

Notifying a Customer of a Promotion

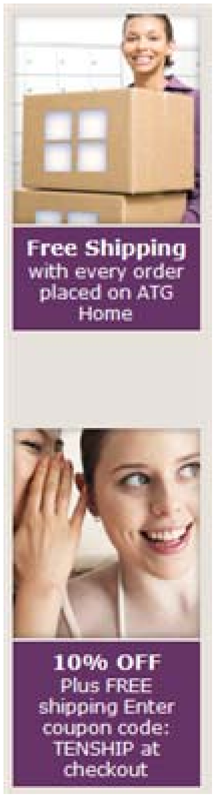
After a promotion has been granted, CRS must notify customers to make them aware of the promotion and encourage purchasing behavior. Notification is done through several different mechanisms, described below.

Email Notification
The `giveAbandonedOrderPromotion` scenario sends an e-mail to notify customers when they are granted the 10% Off Order promotion. See Scenarios That Grant Promotions for more details.

Notification on JSP Pages Using Home Page Splash Images
The home page splash image is used to notify customers about some of the CRS promotions. For example, the home page splash image for registered customers who are female and a member of the `ThirtySomethings` user segment notifies customers about the 25% Off Women's Apparel promotion:

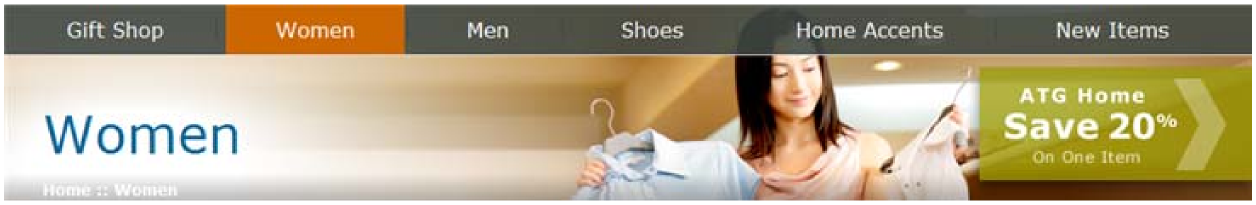


Notification on JSP Pages Using Slots
The `categoryPromotions` scenario uses the `categoryPromotionContent1` and `categoryPromotionContent2` targeters to populate slots on lower left portion of the category and search result pages with `promotionalContent` items for a number of CRS promotions, for example:


Free Shipping
with every order placed on ATG Home
10% OFF
Plus FREE shipping Enter coupon code: TENSHIP at checkout

Notification in Category and Subcategory Hero Image Area

The `store.war/browse/gadgets/itemHeader.jsp` gadget renders the menu bar and header area for categories. The right-hand portion of this area includes an image that notifies the customer of various promotions:



As it builds the menu bar and header area, `itemHeader.jsp` includes the `store.war/browse/gadgets/categoryPromotions.jsp` gadget. `categoryPromotions.jsp` uses the current category's `feature` property to retrieve a URL to the category's promotional image. `itemHeader.jsp` then renders the promotional image so that it overlaps the splash image.

Note: For more information on the feature property, see the Product Catalog chapter.

Notification of All Promotions on promotions.jsp Page

The `store.war/global/gadgets/promotions.jsp` page displays the promotions a customer is eligible for. The `promotions.jsp` page uses the value of the

`/atg/store/pricing/PromotionFilter.siteGroupPromotions` property to render the customer's promotions. This property lists all of the promotions that are appropriate for the customer in the current site context. The `siteGroupPromotions` property is populated as follows:

- Another `PromotionFilter` property, `pricingModelHolder`, points to the `/atg/commerce/pricing/UserPricingModels` component. The `UserPricingModels.getAllPromotions()` method returns an unfiltered collection of promotions for the customer that includes global promotions plus any promotions in the customer's `activePromotions` profile property.
- A third `PromotionFilter` property, `filter`, points to the `/atg/store/collections/filter/PromotionSiteGroupFilter` component. `PromotionFilter` invokes the `PromotionSiteGroupFilter.generateFilteredCollection()` method and passes it the unfiltered collection of promotions.
- `generateFilteredCollection()` filters the promotions in the collection to include only those promotions that are appropriate for the current site context. Promotions that have a site ID that matches the ID of the current site, or the IDs of any sites that share a shopping cart with the current site, are returned. Promotions that have no site ID are also returned. Promotions that do not have a matching site ID are filtered out.

Finally, `promotions.jsp` renders the promotions returned by `generateFilteredCollection()`.

Note: See the *ATG Commerce Programming Guide* for more details on the `UserPricingModels` component.

5 Product Catalog

This chapter discusses how ATG Commerce Reference Store implements its product catalog. It includes the following topics:

Catalog Structure

This section provides an overview of the CRS catalog structure.

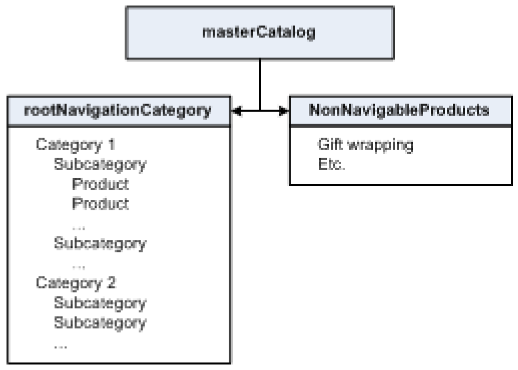
CRS Master Catalogs

CRS has two catalog items in the Catalog repository, `masterCatalog` and `homeStoreCatalog`. ATG Store US and ATG Store Germany both use `masterCatalog` as their default catalog. ATG Home uses `homeStoreCatalog` as its default catalog.

`masterCatalog` contains two categories:

- `rootCategory` is the root category for the hierarchical category and product tree.
- `NonNavigableProducts` provides a location for organizing products, such as gift wrapping, that are non-navigable (that is, not part of the hierarchical category and product tree).

Neither the `rootNavigationCategory` nor the `NonNavigableProducts` category is visible to a Web site visitor. The following figure illustrates the `masterCatalog` structure:



Similarly, `homeStoreCatalog` has two root categories, `homeStoreRootCategory` and `homeStoreNonNavigableProducts`, that perform the same functions for `homeStoreCatalog` that `rootCategory` and `NonNavigableProducts` do for `masterCatalog`.

`masterCatalog` and `homeStoreCatalog` share the Home Accents category and all of its constituent products and SKUs. Therefore, changes made to the Home Accents category will affect both catalogs.

Assigning Site IDs to Catalog Items

Each catalog, category, product, and SKU in the Catalog repository has one or more site IDs assigned to it to indicate which sites the item is affiliated with. When you associate a site with a catalog, you are assigning that site's ID to the catalog. You can create these associations in two ways:

- By assigning a default catalog to a site in Site Administration.
- By assigning a catalog to a site in ATG Merchandising.

After you associate a catalog with a site, two services manage cascading the catalog's site IDs down to its constituent categories, products, and SKUs:

- In the ATG Content Administration environment, `CatalogCompletionService` keeps site IDs updated for catalog folders, catalogs, and categories as you work with your catalogs in ATG Merchandising. Site IDs for products and SKUs are derived in this environment and mirror the site IDs for the parent categories of the products and SKUs.
- During deployment, the site IDs for catalog folders, catalogs, and categories are pushed to the production environment and then `CatalogMaintenanceService` is automatically run to convert the derived site IDs for products and SKUs into database-backed values.

Note: `CatalogCompletionService` is a dynamic, running process that keeps catalog properties up to date in real time as you work in ATG Merchandising. `CatalogMaintenanceService` is a batch process that converts the dynamically generated properties into database-backed values for performance reasons. See the *ATG Commerce Programming Guide* for more information on these two services.

Catalog Extensions

This section describes new properties and item descriptors that were added to the catalog repository for CRS, as well as changes to existing properties and item descriptors. The `Store.Estore` module adds the new and modified definitions to the catalog repository in the `/atg/commerce/catalog/custom/customCatalog.xml` file.

Note: Some of the properties that appear in the `/atg/commerce/catalog/custom/customCatalog.xml` file are not used in CRS. This chapter covers those that are.

Extensions to Catalog Repository Item Descriptors

This section describes extensions to existing item descriptors in the catalog repository.

Extensions to the Catalog Item Descriptor

CRS extends the `catalog` item descriptor with one property called `rootNavigationCategory`. The `rootNavigationCategory` property is of type `category` and it references the `rootCategory` category. `rootCategory` functions as the root of the navigable category/product tree. It is a sibling of the `NonNavigableProducts` category, which provides a location for organizing products, such as gift wrapping, that are non-navigable (that is, not part of the category tree). Neither the `rootCategory` nor the `NonNavigableProducts` category is visible to a website visitor.

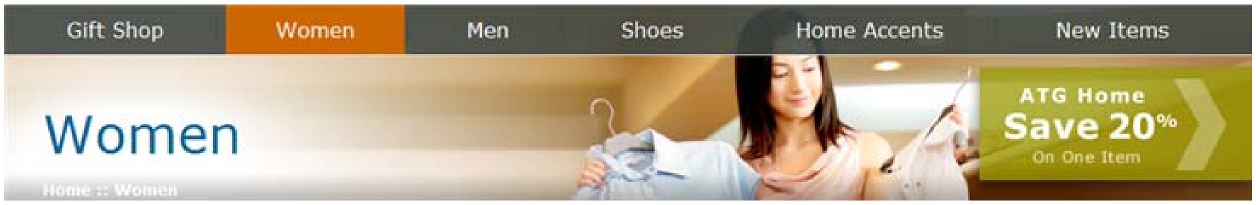
Extensions to the Category Item Descriptor

This section describes additions and modifications to the `category` item descriptor.

feature and myFeature
The `feature` and `myFeature` properties work together to ensure that a promotional image, such as the one below, is always returned for a category:

A category's `myFeature` property contains a promotional image, in the form of a `promotionalContent` item. `feature` is a derived property that looks at the `myFeature` property of the current category to find a promotional image. If the current category's `myFeature` property is null, `feature` looks at the `feature` property of the category's parent, effectively repeating the process with the parent. That is, it looks at the parent's `myFeature` property and, if the property is null, it proceeds to the `feature` property for the parent's parent. Using this methodology, a call for a category's promotional image will move up the category tree until it finds a promotional image. Note that it is the `feature` property that is used in the CRS JSP code.

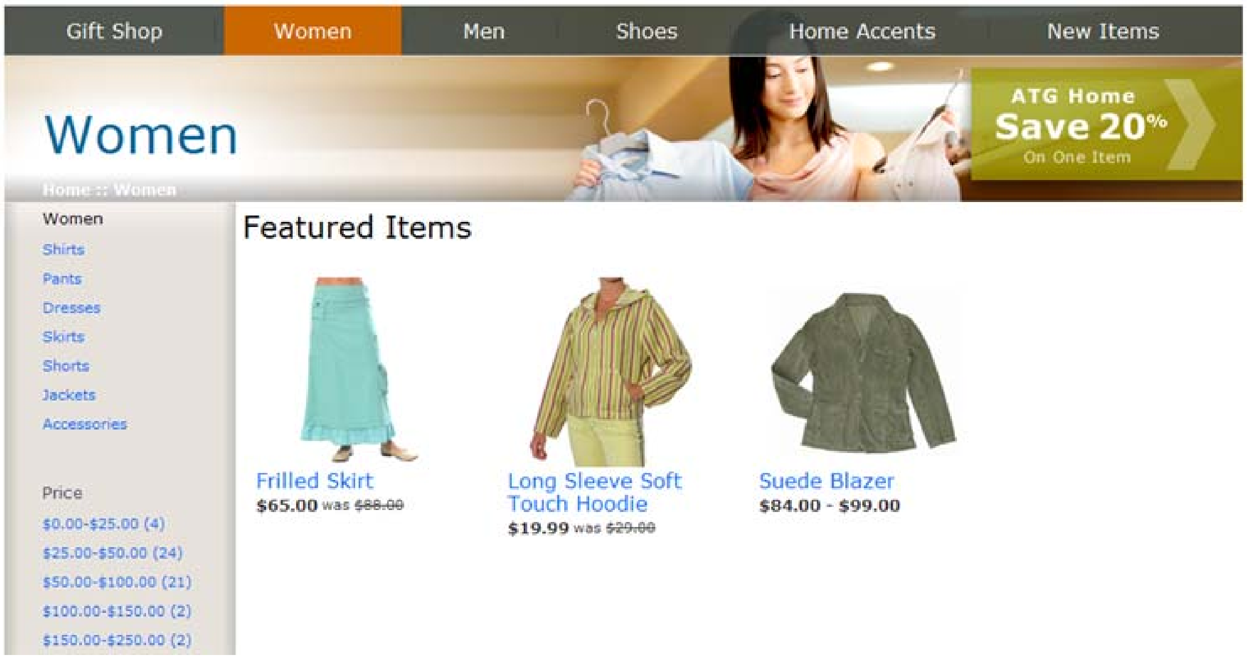
The `store.war/browse/gadgets/itemHeader.jsp` gadget renders the menu bar and header area for categories. As it builds the menu bar and header area, `itemHeader.jsp` includes the `store.war/browse/gadgets/categoryPromotions.jsp` gadget, which is responsible for retrieving the category's promotional image (that is, its `feature` property). `itemHeader.jsp` then renders the promotional image so that it overlaps the splash image.



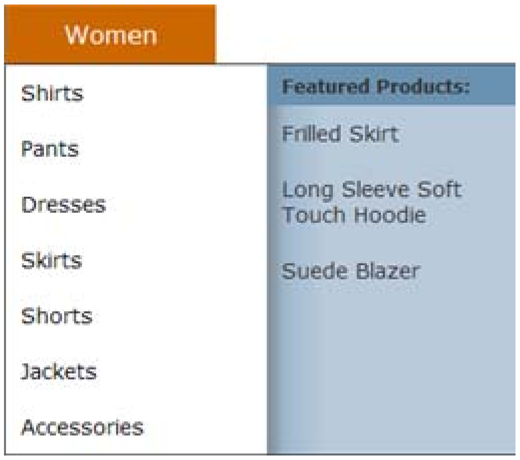
Note: `promotionalContent` is a CRS-specific item type. For detailed information, see Custom Item Descriptors in the Catalog Repository.

relatedProducts

`relatedProducts` stores a list of items of type `product` that represent a set of products in a category. The `store.war/browse/gadgets/featuredProducts.jsp` gadget uses the `relatedProducts` property to populate the list of featured products on the category page:



The `store.war/navigation/gadgets/catalog.jsp` gadget uses `relatedProducts` to populate the list of featured products that appear in the menu bar:



titleImage

`titleImage` stores a splash image for the category in the form of a `media` item. This property is used by the `store.war/browse/gadgets/itemHeader.jsp` gadget to display a background image for the category menu bar and header area.

Extensions to the media-internal-text Item Descriptor

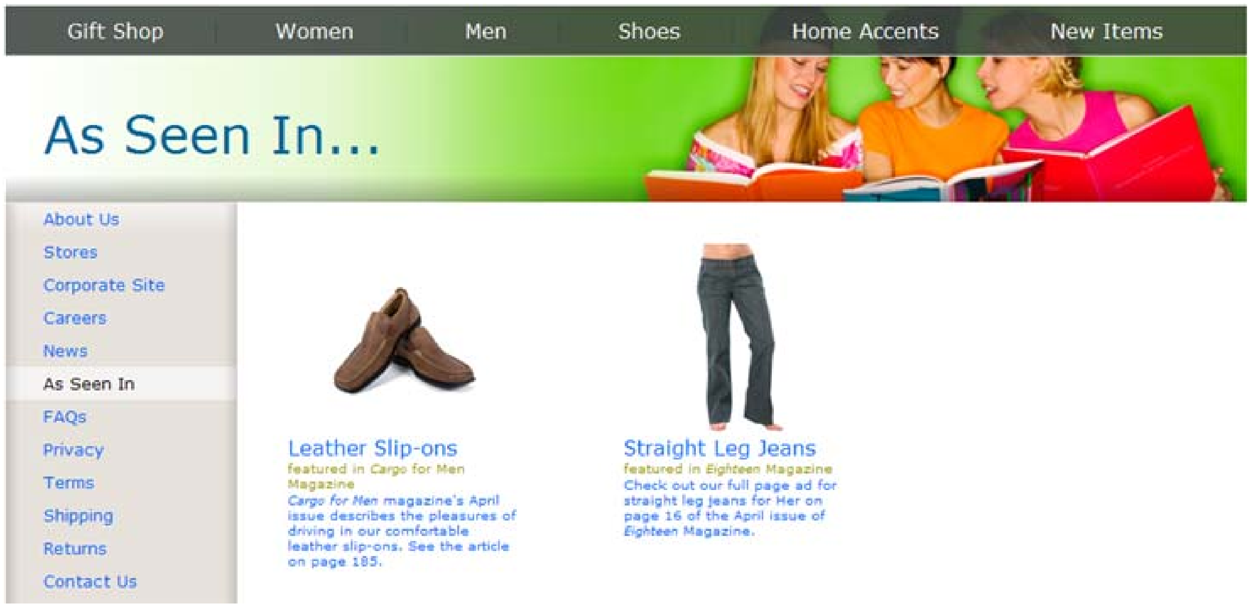
CRS extends the `media-internal-text` item descriptor with changes that allow these items to be edited in the ATG Control Center. These changes don't affect how items of this type function in ATG Store.

Extensions to the Product Item Descriptor

This section describes additions and modifications to the `product` item descriptor.

asSeenIn

`asSeenIn` stores information about the locations where a customer might see a product, such as a magazine article or television show. The `asSeenIn` property is of type `asSeenIn`. The `store.war/browse/asSeenIn.jsp` page uses this property to populate the As Seen In page in ATG Store:

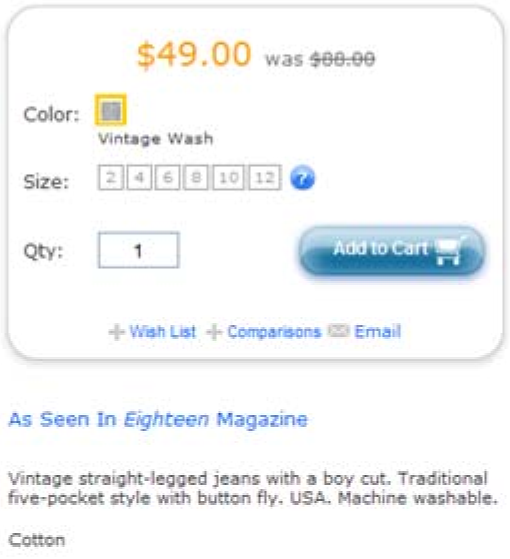


Leather Slip-ons
featured in *Cargo* for Men Magazine
Cargo for Men magazine's April issue describes the pleasures of driving in our comfortable leather slip-ons. See the article on page 185.

Straight Leg Jeans
featured in *Eighteen* Magazine
Check out our full page ad for straight leg jeans for Her on page 16 of the April issue of *Eighteen* Magazine.

`asSeenIn.jsp` uses the `/atg/store/droplet/AsSeenInRQL` component to retrieve all products where the `asSeenIn` property is not null, regardless of site context. It then uses `/atg/store/droplet/CatalogItemFilterDroplet` to limit the products to those that are relevant for the current site.

The `store.war/browse/gadgets/productAsSeenIn.jsp` gadget also uses the `asSeenIn` property to populate As Seen In information that appears below the color/size picker on the product detail page for certain products:



As Seen In *Eighteen* Magazine

Vintage straight-legged jeans with a boy cut. Traditional five-pocket style with button fly. USA. Machine washable.

Cotton

Note: `asSeenIn` is a CRS-specific item type. For detailed information, see Custom Item Descriptors in the Catalog Repository.

briefDescription
`briefDescription` provides a brief description of the product. This property is included in the ATG Search index.

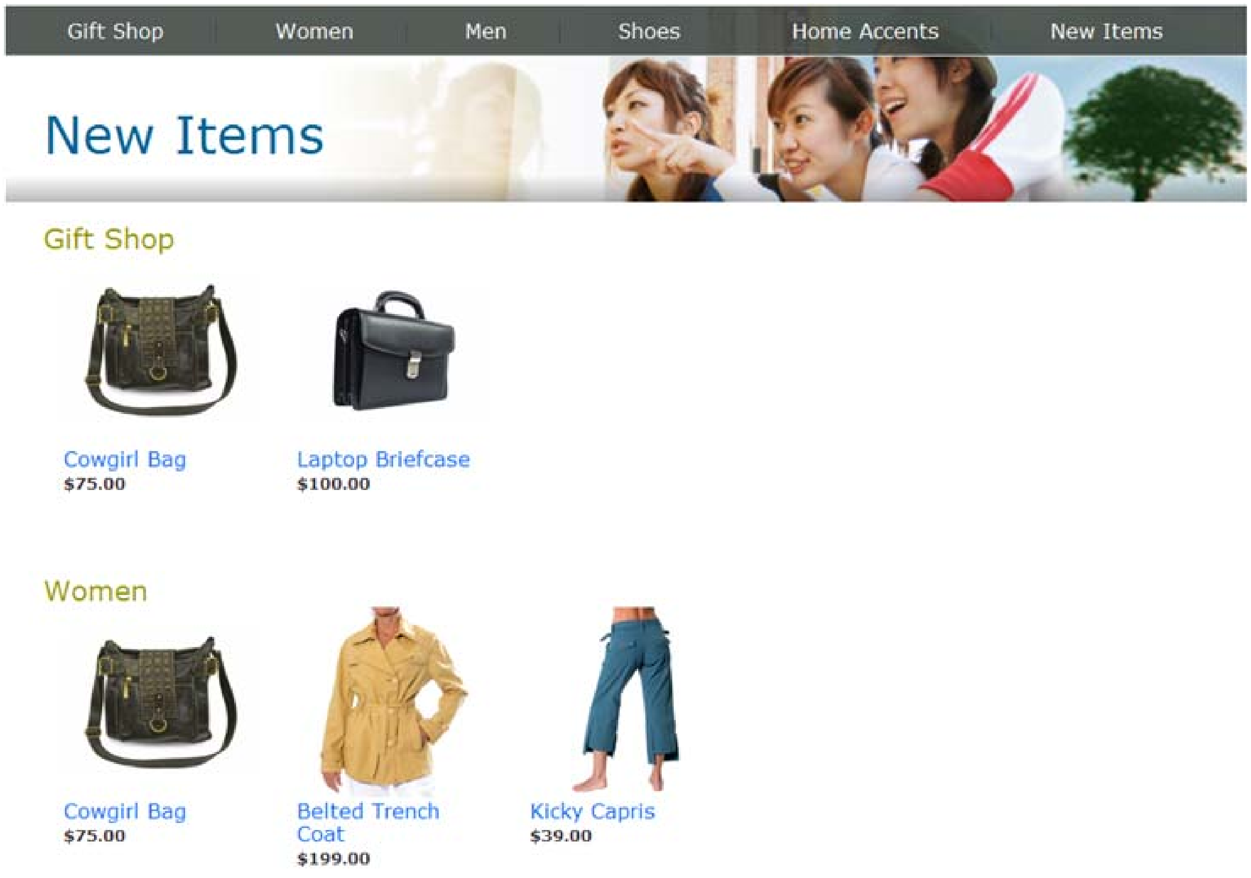
emailAFriendEnabled
`emailAFriendEnabled` is a Boolean property that controls whether an Email link appears in the product detail page that allows you to e-mail a product description to a friend:

The `store.war/browse/gadgets/productAction.jsp` gadget uses the `emailAFriendEnabled` property to determine if the Email To A Friend option should appear.

features
`features` is a list property of items of type `feature`. It stores product characteristics (for example, material, color, and so on). The `features.displayName` property is used as a global faceting property. See Faceted Search in the ATG Search Integration chapter for more details.

Note: `feature` is a CRS-specific item type. For detailed information, see Custom Item Descriptors in the Catalog Repository.

new
new is a Boolean property that indicates whether an item is new or not. This property is manually defined for each product in the CRS sample data. The `/browse/gadgets/newItemList.jsp` gadget uses new to determine which products to render on the New Items page:



The best practice approach to identifying new products is to calculate which products are new based on date. Because demonstration data gets stale quickly, CRS cannot use this methodology, so `newItemList.jsp` uses the manually defined `new` property instead. If the `new` property is not set to `true`, `newItemList.jsp` compares a product's `daysAvailable` property against the site's `newProductThresholdDays` property. If the number of days the product has been available is less than the new product threshold, the product is considered new and rendered accordingly.

preorderable, preorderEndDate, and useInventoryForPreorder
CRS includes three properties for managing preorderable products:

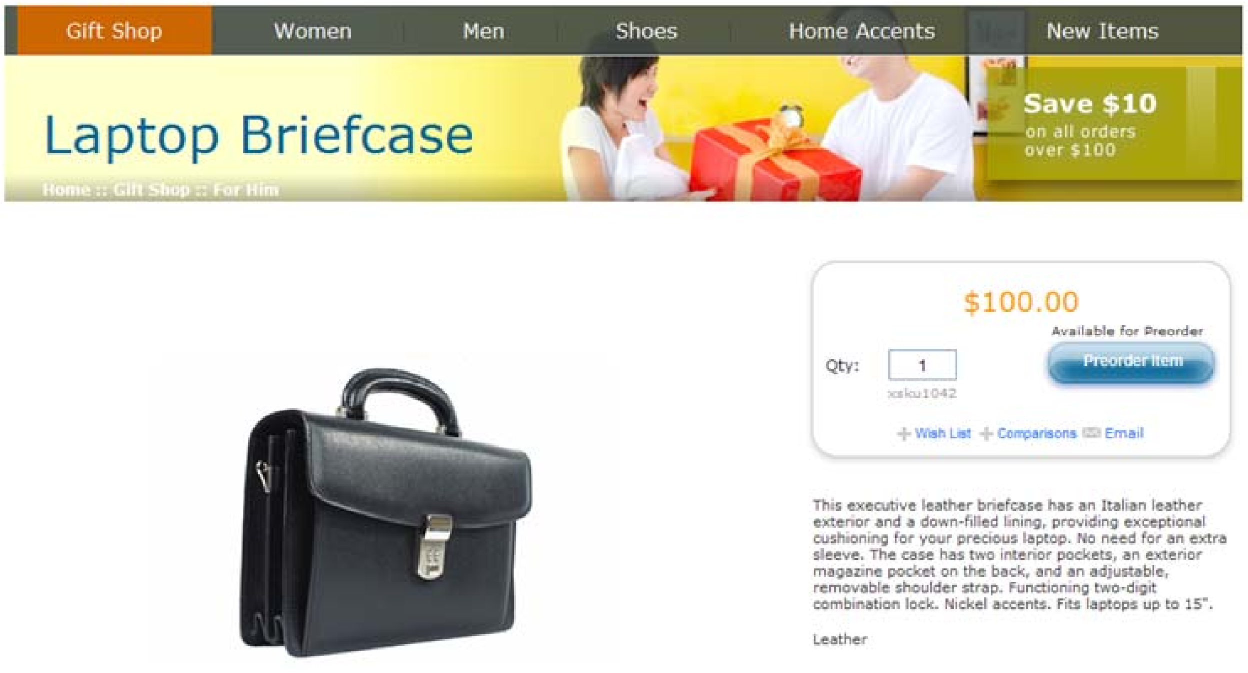
- `preorderable` is a Boolean property that indicates whether an item is preorderable or not. Customers can order preorderable items before they are stocked in inventory.
- `preorderEndDate` contains the date up until which a product may be preordered, which is generally the day the product becomes available for standard ordering. `preorderEndDate` is an exclusive value; in other words, if the date is set to September 30th, the product may be preordered through September 29th.
- `useInventoryForPreorder` is a Boolean property that controls whether a preordering limit exists for a product. When set to `false,` a product has

unlimited preordering capability. When set to `true`, a request to preorder is checked against a preorder inventory. When the preorder inventory is depleted, the product may no longer be preordered.

The `store.war/browse/gadgets/skuAvailabilityLookup.jsp` and `store.war/browse/gadgets/multiSkuAvailabilityLookup.jsp` gadgets use the `/global/gadgets/SkuAvailabilityLookup` component to determine whether a product is preorderable. `SkuAvailabilityLookup`, which is an instance of `atg.projects.store.droplet.SkuAvailabilityLookup`, returns an item's availability status based on a product's properties and the inventory levels reported by the `/atg/commerce/inventory/InventoryManager` component. `SkuAvailabilityLookup` will identify a product as preorderable if the following conditions are met:

- `preorderable` property is true.
- `preorderEndDate` has not yet passed or is null.
- `useInventoryForPreorder` property is set to false, or it is set to true and preorder inventory is still available.

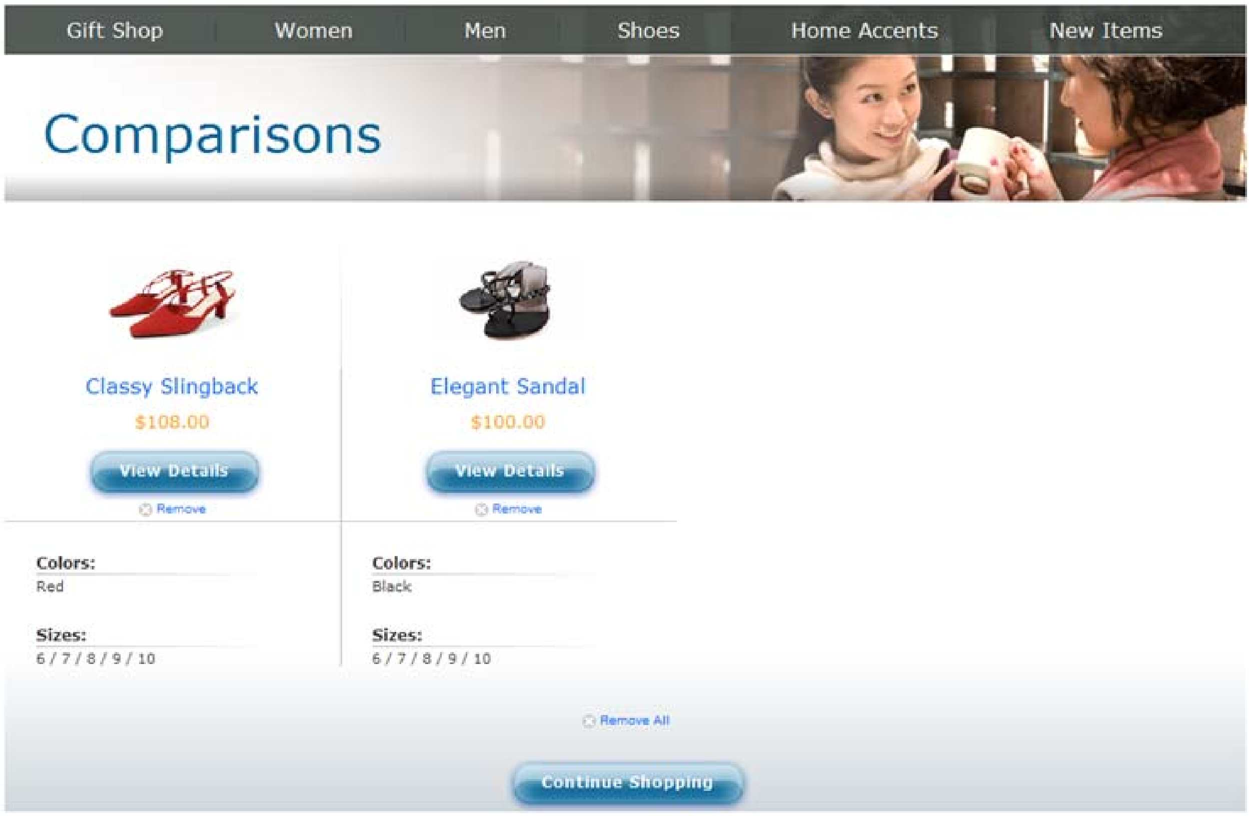
Items that are preorderable appear like this on the product detail page:



This executive leather briefcase has an Italian leather exterior and a down-filled lining, providing exceptional cushioning for your precious laptop. No need for an extra sleeve. The case has two interior pockets, an exterior magazine pocket on the back, and an adjustable, removable shoulder strap. Functioning two-digit combination lock. Nickel accents. Fits laptops up to 15".

Leather

promoImage
`promoImage` stores the product image displayed by the `store.war/browse/gadgets/productPromoImg.jsp` gadget on the product comparisons page, in the form of a `media` item:



shippableCountries and nonShippableCountries

`shippableCountries` stores a list of countries to which a product may be shipped, while `nonShippableCountries` stores a list of countries to which a product may not be shipped. Use the `shippableCountries` property in those instances where it is more efficient to list a small number of countries to which a product may be shipped. By contrast, use `nonShippableCountries` when it is more efficient to list a small number of countries to which a product may not be shipped.

The `/atg/store/droplet/ShippingRestrictionsDroplet` servlet component, which is used by a number of CRS gadgets, considers both properties when determining if a shipping address is valid for a particular item:

- If `shippableCountries` is null, all countries are shippable, except those listed in `nonShippableCountries`.
- If `shippableCountries` contains one or more countries, only those countries are shippable and `nonShippableCountries` is ignored.

titleImage

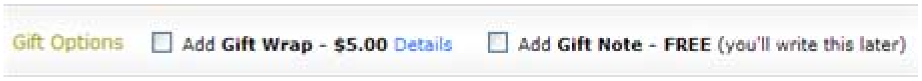
`titleImage` stores a title image for the product in the form of a `media` item. CRS does not use this property.

Extensions to the SKU Item Descriptor

This section describes additions and modifications to the SKU item descriptor.

giftWrapEligible

The `giftWrapEligible` property is a Boolean that indicates whether an item is eligible for gift wrapping. The `store.war/cart/gadgets/giftWrap.jsp` gadget invokes the `/atg/store/droplet/ShowGiftWrap` servlet bean component to determine if the gift wrap options should be shown on the shopping cart page. At least one of the hardgood items in an order must have a `giftWrapEligible` value of `true` for Add Gift Wrap option to be rendered. The Add Gift note option is always rendered, regardless of the `giftWrapEligible` property.


Gift Options
Add Gift Wrap - $5.00 Details
Add Gift Note - FREE (you'll write this later)

taxStatus

The `taxStatus` property tags items with different product codes that indicate whether a product should be taxed. CRS adds a `gift_wrapping` code because, typically, gift wrapping charges should not be included when taxes are calculated. See *Designating Tax Status of Products* in the *ATG Commerce Programming Guide* for more details on using tax status codes.

type

`type` is an enumerated property that classifies the type of SKU. `store.war/global/util/displaySkuProperties.jsp` uses this value to determine what type of SKU it is rendering and, hence, what type of information should be provided. Possible values are:

- `sku` (0)
- `clothing-sku` (1)
- `furniture-sku` (2)
- `configurableSku` (4001)

See Custom Item Descriptors in the Catalog Repository for more information on the clothing and furniture SKU sub-types.

Custom Item Descriptors in the Catalog Repository

CRS adds the following item descriptors to the catalog repository:

- `asSeenIn`
- `clothing-sku`
- `features`
- `furniture-sku`
- `promotionalContent`

This section describes the CRS-specific item types and their key properties. For a complete list of item type properties, use the Component Browser in the ATG Dynamo Server Admin.

asSeenIn Item Descriptor

The `asSeenIn` item type stores information about the location where a product may have been seen by a customer, such as a magazine article or a television show. The `asSeenin` product property is of type `asSeenIn`.

Property	Description
`date`	The date the product appeared in a publication, on a television show, etc.
`description`	A description of the product’s appearance.
`displayName`	A name for the `asSeenIn` item. This name appears in the ATG Business Control Center and may also be used on a Web site to identify the item.
`id`	A unique ID for the `asSeenIn` item.
`source`	A string that stores the source of the product appearance, for example, a publication, a television show, etc.

clothing-sku Item Descriptor

The `clothing-sku` item type is a sub-type of the `sku` item type and it adds properties that are appropriate for clothing items.

Property	Description
`color`	`color` is a string property that represents an item’s color. It is used in several ways: - By the color/size picker. See How the SKU Picker is Populated in the Pages and Navigation chapter for details. - As a faceting property. See Faceted Search in the ATG Search Integration chapter for details. - By the `store.war/global/util/displaySkuProperties.jsp` gadget to render the color of an item on the shopping cart page.
`colorSwatch`	`colorSwatch` holds an image that represents one of the color swatches that appear in the color/size picker, in the form of a `media` item.

Property	Description
`size`	`size` is a string property that represents an item's size. It is used in several ways: - By the color/size picker. See How the SKU Picker is Populated in the Pages and Navigation chapter for details. - As a faceting property. See Faceted Search in the ATG Search Integration chapter for details. - By the `store.war/global/util/displaySkuProperties.jsp` gadget to render the size of an item on the shopping cart page.

furniture-sku Item Descriptor

The `furniture-sku` item type is a sub-type of the `sku` item type and it adds properties that are appropriate for furniture items.

Property	Description
`colorSwatch`	`colorSwatch` holds an image that represents one of the wood finish swatches that appear in the wood finish picker, in the form of a `media` item.
`woodFinish`	`woodFinish` is a string property that represents an item's wood finish. It is used in several ways: - By the wood finish picker. See How the SKU Picker is Populated in the Pages and Navigation chapter for details. - As a faceting property. See Faceted Search in the ATG Search Integration chapter for details. - By the `store.war/global/util/displaySkuProperties.jsp` gadget to render the wood finish of an item on the shopping cart page.

feature Item Descriptor

The `feature` item type stores information about a specific product feature, such as type of fabric or material. The `features` product property is list of items of type `feature`.

Property	Description
`description`	A short description of the `feature` item.
`displayName`	A name for the `feature` item that appears in the ATG Business Control Center.

Property	Description
`featureName`	A string that identifies the `feature` item and may be used on a Web site to identify the item.
`id`	A unique ID for the `feature` item.

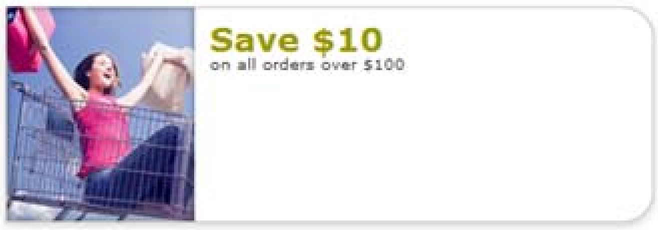
`largeImage`
`smallImage`
`titleImage`
	Three image properties, of type `media`, that allow you to associate various images with the `feature` item.

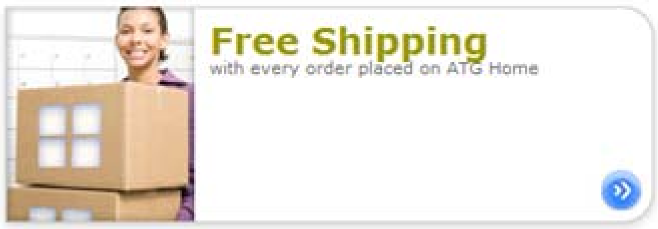
`longDescription`	A long description of the `feature` item.

promotionalContent Item Descriptor

The `promotionalContent` item type provides information about a pricing promotion or marketing campaign on ATG Store. It allows a merchant to consolidate any number of possible marketing tools into a single item: image, text, a link to a product or category, and a pricing promotion. This gives the merchant flexibility for the type of content that can be populated in a slot on a page, or be referenced from a category or product. Note that `promotionalContent` items are informational only. The actual granting of pricing promotions is handled by other mechanisms.

For the purposes of CRS, a `promotionalContent` item is essentially an image, sometimes with supporting text. This illustration shows two examples of `promotionalContent` items:





`promotionalContent` properties include:

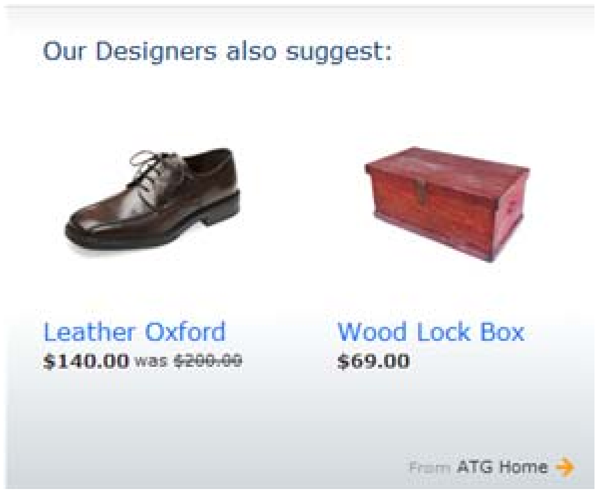
Property	Description
associatedCategory	An associated category for the promotionalContent item. The store.war/WEB-INF/tags/store/promotionalContentWrapper.tag uses this property to render a link that takes a customer to a category page that is appropriate for the promotion.
associatedProduct	An associated product for the promotionalContent item. The store.war/WEB-INF/tags/store/promotionalContentWrapper.tag uses this property to render a link that the customer can click to view a product detail page that is appropriate for the promotion.
associatedSite	An associated site for the promotionalContent item. The store.war/WEB-INF/tags/store/promotionalContentWrapper.tag uses this property to render a link that the customer can click to go from the promotion to the home page for the associated site.
creationDate	A timestamp for the date and time that the promotionalContent item was created. Defaults to the current date and time at the time of creation.
description	A short description of the promotionalContent item.
displayName	A name for the promotionalContent item.
id	A unique ID for the promotionalContent item.
image, derivedImage	The image and derivedImage properties work together to derive a URL for an image that represents the promotionalContent item. The image property contains a URL to an image. CRS uses URLs that contain {site} and {language} variables that provide the foundation for deriving a site-specific URL for the image. For example: /crsdocroot/images/atgstore/homepage_promo_{site}_{language}_fashionista.png A JSP reference to the derivedImage property calls the atg.projects.store.dp.StoreContextDerivation class and passes it the value of the image property. StoreContextDerivation derives the URL by replacing the {site} and {language} variables in the image property value with the current site's site ID and language. For example, the image property value above could result in a URL similar to the following: /crsdocroot/images/atgstore/homepage_promo_storeSiteDE_de_fashionista.png
linkText linkURL	A string and an associated URL that may be rendered on a page. Clicking the linkText takes a customer to the linkURL.
longDescription	A long description of the promotionalContent item.

Property	Description
`parentFolder`	The folder that the `promotionalContent` item is stored in.
`promotion`	A pricing promotion that is associated with the `promotionalContent` item.
`template`	A JSP template for rendering the `promotionalContent` item.

Cross-Site Related Products Catalog Data

Several ATG Store US products have their `relatedProducts` property populated with references to ATG Home US products. The function of this data is to drive traffic from ATG Store US to ATG Home.

All products in the Gift Ideas/ For Him category reference the Wood Lock Box in their `relatedProducts` property. All products in the Gift Ideas/ For Her category reference the Velvet-lined Storage Box in their `relatedProducts` property. The `store.war/browse/gadgets/RecommendedProducts.jsp` page uses this data to render product suggestions at the bottom of the product detail page, along with site indicators that notify the customer that these products come from another site.



CRS Pricing

This section describes how CRS calculates and renders prices on the product detail page.

Note: The mechanism for rendering prices depends on whether the prices are being shown on the product detail page or as part of the checkout process, at which point prices may get modified by things like promotions and discounts. This section describes the product detail page pricing mechanism. See the *Order Handling* chapter for details on pricing during the checkout process.

CRS Price Lists

As recommended in the *Multisite Roadmap* section of the *ATG Multisite Administration Guide*, CRS uses prices lists, instead of SKU-based pricing, to calculate the prices for items. There are four CRS price lists:

- `List Price` and `Sale Prices`: Includes list prices and sale prices, respectively, for any items that appear on ATG Store US and ATG Home. Because these two sites share a shopping cart, it is important that the price lists they use cover any items from either site that a customer may add to the shopping cart. `List Prices` and `Sale Prices` use dollars as their currency.
- `German List Prices` and `German Sale Prices`: Includes list prices and sale prices, respectively, for any items that appear on ATG Store Germany. Because ATG Store Germany has its own shopping cart, it also has its own price lists. `German List Prices` and `German Sale Prices` use euros as their currency.

Setting Price List Properties on the User Profile

ATG Commerce adds two properties to user profiles, `priceList` and `salePriceList`, which specify the price lists to use for a customer. ATG Commerce populates each of these properties with one of the four price lists described above, based on the current site context. For details on how this population mechanism works, see the *Multisite Overview* section of the *ATG Multisite Administration Guide* and the *ATG Commerce Programming Guide*.

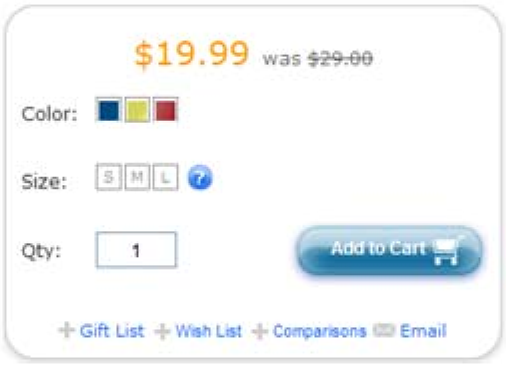
Rendering Prices on the Product Detail Page

CRS product detail pages include the `store.war/browse/gadgets/pickerPriceAttribute.jspf` page fragment. This page fragment determines how pricing information should be presented on the product detail page. There are two cases: a SKU is currently selected or no SKU is selected.

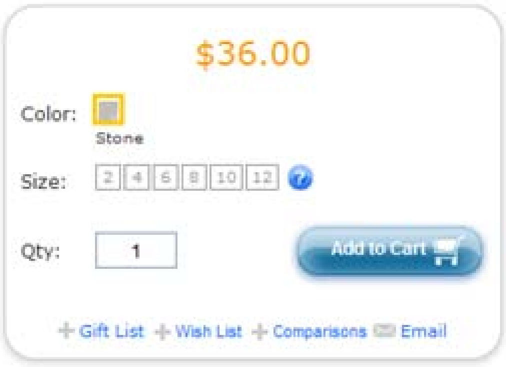
Rendering Prices for a Selected SKU

When a SKU is currently selected, the price information should reflect the price for that SKU only. In this case, `pickerPriceAttribute.jspf` calls `store.war/global/gadgets/PriceLookup.jsp` to render the SKU's price. `priceLookup.jsp` calls the `/atg/commerce/pricing/priceLists/PriceDroplet` component twice, once with the price list stored in the customer's `priceList` property and a second time with the price list stored in the customer's `salePriceList` property. These two calls allow `priceLook.jsp` to determine if the SKU is on sale and, if so, render the price information accordingly. If a sale price exists, `priceLookup.jsp` shows sale price followed by the list price in strike-through format. If a sale price doesn't exist, `priceLookup.jsp` simply renders the list price.

This illustration shows the pricing portion of the product detail page for a SKU that is on sale:

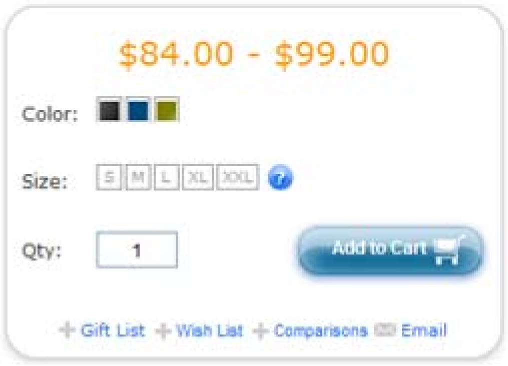


This illustration shows the pricing portion of the product detail page for a SKU that is not on sale:



Rendering Prices When No SKU is Selected

When no SKU is selected, the price information should reflect the range of prices for all of the SKUs associated with the product. In this case, `pickerPriceAttribute.jspf` calls `store.war/global/gadgets/PriceRange.jsp` to render the price range. `PriceRange.jsp` uses the `atg/commerce/pricing/PriceRangeDroplet` to calculate the highest and lowest prices for a product's constituent SKUs and then renders the results. This gadget takes sale prices into account when calculating the range, if they exist.


$84.00 - $99.00
Color:
Size:
S
M
L
XL
XXL
Qty:
1
Add to Cart
Gift List
Wish List
Comparisons
Email

6 Inventory Management

ATG Commerce Reference Store uses the standard ATG Commerce inventory management system, with a few customizations. This chapter discusses those customizations and how inventory management is implemented in the stores. It includes the following topics:

Inventory Manager

By default, inventory is managed in ATG Commerce by the `/atg/commerce/inventory/InventoryManager` component, which is of class `atg.commerce.inventory.RepositoryInventoryManager`. CRS changes this component's class to `atg.projects.store.inventory.StoreInventoryManager`, which is a CRS-specific subclass of `RepositoryInventoryManager` that adds custom logic for determining the availability status of SKUs. Inventory data is stored in the inventory repository, `/atg/commerce/inventory/InventoryRepository`. This repository is populated with inventory data about each SKU in the CRS product catalog. Most SKUs are in stock, but a few are preorderable, backorderable, or out of stock, to illustrate the inventory functionality of the site.

Availability Status

For each SKU in the CRS catalog, there is a corresponding `inventory` item in the inventory repository that tracks inventory data for the SKU. ATG Commerce includes a mechanism for tracking the current availability of a SKU using the `inventory` item's `availabilityStatus` property. This property can be set to integers that represent IN_STOCK (1000), OUT_OF_STOCK (1001), PREORDERABLE (1002), BACKORDERABLE (1003), DERIVED (1004), and DISCONTINUED (1005).

In the CRS inventory repository, every `inventory` item's `availabilityStatus` property is set to 1004 (DERIVED). This means that rather than having an explicit value, the status is derived at runtime based on the values of three other `inventory` item properties: `stockLevel`, `backorderLevel`, and `preorderLevel`. The logic for deriving the status value is:

- If `stockLevel` is not 0, the SKU is in stock.
- If `stockLevel` is 0 but `backorderLevel` is not 0, the SKU is backorderable.
- If `stockLevel` and `backorderLevel` are both 0, but `preorderLevel` is not 0, then the SKU is preorderable.
- If all three values are 0, the SKU is out of stock.

Preorderable Customizations

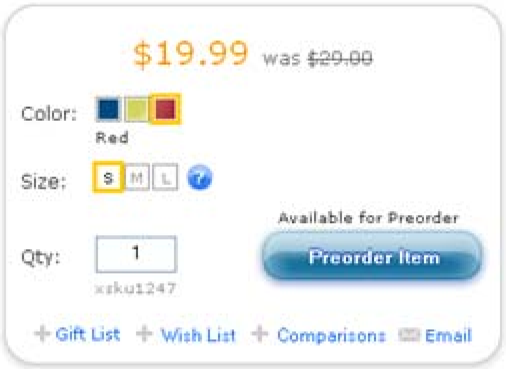
CRS uses the DERIVED availability status for determining if a SKU is in stock, out of stock, or backorderable. To determine if a SKU is preorderable, however, the `StoreInventoryManager` class implements custom logic based on properties of the SKU's `inventory` item plus CRS-specific properties of the `product` item type. These custom properties are `preorderable`, `useInventoryForPreorder`, and `preorderEndDate`.

The logic works like this:

- If `preorderable` is `false`, the product's SKUs are not preorderable.
- If `preorderable` is `true` and `useInventoryForPreorder` is `false`, the SKUs are preorderable if the product's `preorderEndDate` is in the future or null.
- If `preorderable` is `true` and `useInventoryForPreorder` is `true`, a SKU is preorderable if the product's `preorderEndDate` is in the future or null, and the properties of the SKU's `inventory` item meet these conditions:
    - `stockLevel` is 0.
    - `preorderLevel` is greater than `preorderThreshold`.

Inventory Information on Site Pages

CRS looks up inventory information for a product when the product detail page is loaded, and sends the information over to the browser, where it is available for use by browser-side JavaScript. This information is accessed at various points: when a customer selects an individual SKU or hovers the mouse over a color or size in the picker; when displaying a shopping cart; during the checkout process. For example, if a customer views the product detail page for the Long Sleeve Soft Touch Hoodie, and selects size Small, color Red, the SKU picker looks like this:



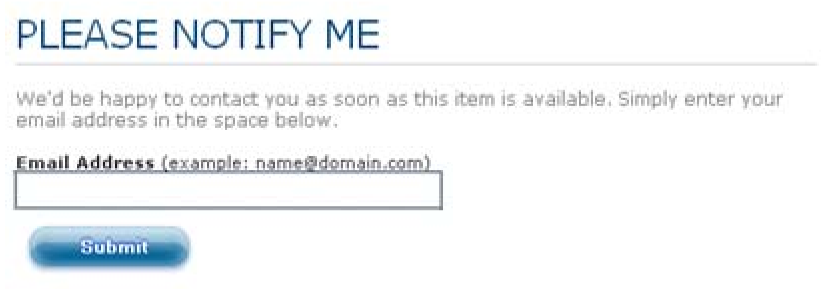
When a SKU is selected that is not in stock, the label on the Add to Cart button changes to reflect this. In the example above, the label has changed to Preorder Item because the selected SKU is preorderable.

Inventory lookup is done by the `store.war/global/gadgets/skuAvailabilityLookup.jsp` gadget. This gadget invokes the `/atg/store/droplet/SkuAvailabilityLookup` servlet bean component, which is of class `atg.project.store.droplet.SkuAvailabilityLookup`. This component is configured to use the CRS Inventory Manager to obtain inventory information from the inventory repository. If a SKU is not in stock, `SkuAvailabilityLookup` displays a message about its availability status (such as Available for Preorder, as shown in the figure above), and changes the label of the Add to Cart button (in this case, to Preorder Item).

See Appendix B: Sample Data for more information about preorderable products.

Back in Stock Emails

If a customer selects a SKU that is out of stock (i.e., `stockLevel`, `backorderLevel`, and `preorderLevel` are all 0), the label on the Add to Cart button changes to Email Me When Available. If the customer clicks this link, a popup window prompts for an e-mail address to send a notification to when the item is back in stock:


PLEASE NOTIFY ME

We'd be happy to contact you as soon as this item is available. Simply enter your email address in the space below.

Email Address (example: name@domain.com)

If the customer fills in an e-mail address and clicks the Submit button, a repository item of type `backInStockNotifyItem` is created in the profile repository. This item holds the e-mail address plus the product ID and SKU ID of the out-of-stock item.

The CRS `Store.Fulfillment` module adds a Patch Bay message sink class, `atg.projects.store.inventory.BackInStockNotificationService`, to the standard Fulfillment system. A component of this class, `/atg/store/inventory/BackInStockNotificationService`, listens for JMS messages of type `UpdateInventory`. This message type, which is typically sent by an external inventory system, informs the Fulfillment system about formerly unavailable products and SKUs that stock is now available for. (See the *ATG Programming Guide* for information about Patch Bay.)

When an `UpdateInventory` message is received, `BackInStockNotificationService` examines the `backInStockNotifyItem` repository items. If a `backInStockNotifyItem` contains the product ID or SKU ID of one of the products or SKUs listed in the `UpdateInventory` message, `BackInStockNotificationService` sends a Back in Stock e-mail message to the address stored in the `backInStockNotifyItem`, and then deletes the `backInStockNotifyItem`.

See Appendix B: Sample Data for more information about out-of-stock products.

7 Order Handling

A key aspect of any ATG Commerce implementation is the handling and processing of orders. This chapter discusses the order-handling process in ATG Commerce Reference Store. It includes the following topics:

Shopping Cart

The main shopping cart page is `store.war/cart/cart.jsp`. This page is accessed by clicking the Checkout link on the upper right part of most pages. The shopping cart page provides options for managing the contents of the cart: changing item quantities, removing items, entering promotion codes, etc.

The `cart.jsp` page uses the `/atg/store/order/purchase/CartFormHandler` to manage the contents of the cart. This component is of class `atg.projects.store.order.purchase.StoreCartFormHandler`, which is a CRS-specific subclass of the standard ATG Commerce `atg.commerce.order.purchase.CartModifierFormHandler` class.

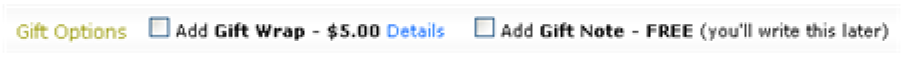
The shopping cart page includes a Checkout button that initiates the standard checkout process. The page also conditionally includes an Express Checkout button. See the Checkout Process section for more information.

ATG Store US and ATG Home share a shopping cart. A customer can put items from both sites in the same cart, and check out on either site. The shopping cart page indicates the site for each product in the cart. If a product is available on both sites, the cart page reflects the site the product was actually added to the cart from. Different discounts and promotions may be applied depending on which site the customer checks out from.

Note that in addition to the shopping cart page, CRS includes a JavaScript-based shopping cart dropdown that enables customers to see the contents of their cart without leaving the current page. (See Rich Shopping Cart Widget in the Pages and Navigation chapter.) The dropdown is primarily informational, rather than a means of managing the cart. The customer must still access the shopping cart page to modify the cart or initiate checkout.

Gift Wrap

The `store.war/cart/gadgets/giftWrap.jsp` gadget on the shopping cart page is responsible for rendering the gift wrap and gift note options. The gift note option is displayed for all orders. To determine whether to render the gift wrap option, `giftWrap.jsp` invokes the `/atg/store/droplet/ShowGiftWrap` servlet bean. If the `giftWrapEligible` property is `true` for at least one hardgood SKU in the order, the gift wrap option is displayed as well:


Gift Options
Add Gift Wrap - $5.00 Details
Add Gift Note - FREE (you'll write this later)

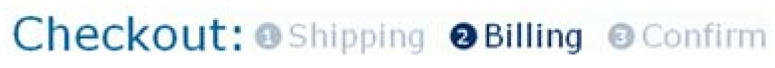
If the customer selects the gift wrap option, it applies to all of the SKUs in the order whose `giftWrapEligible` property is `true`. It is not possible to select the gift wrap option for some SKUs and not for others. The price of gift wrapping is the same regardless of the number of SKUs.

The `giftWrap.jsp` gadget includes the `/atg/store/order/purchase/CartFormHandler` component, which manages adding the gift wrap and gift note to the order. Gift Wrap is an actual SKU that is added to the order if gift wrapping is selected.

Checkout Process

The checkout process encompasses a number of steps: applying promotions, calculating taxes, specifying shipping options and payment options, etc. These steps are handled through a sequence of pages that the customer steps through. Most of the JSPs for these pages are found in the `store.war/checkout/` and `store.war/checkout/gadgets/` directories.

Each checkout page displays a progress bar that indicates the stages of the process. In the following example, the customer is in the Billing stage:



The exact set of pages the customer steps through can vary depending on context. For example, if the customer is not logged in when he or she clicks the Checkout button, a Login page (`store.war/checkout/login.jsp`) displays, with options for logging in to an existing account, creating a new account, or checking out without logging in. If the customer selected the “Add gift note” option on the shopping cart page, the Your Gift Message page (`store.war/checkout/giftMessage.jsp`) displays during checkout, with a form for creating a gift note.

Express Checkout

In addition to the standard checkout button, the shopping cart page also includes an Express Checkout button if all of the following are true:

- The customer is logged in.
- The customer has a default shipping method.

- The customer has a valid default shipping address.
- The customer has a valid default credit card.

The `store.war/cart/gadgets/actionItems.jsp` gadget invokes the `/atg/store/droplet/ExpressCheckoutOk` servlet bean, which determines whether these conditions are all true and, if so, renders the Express Checkout button. This component is a CRS-specific servlet bean of class `atg.projects.store.droplet.ExpressCheckoutOkDroplet`.

Clicking the Express Checkout button takes the customer directly to the Confirm stage, using the default shipping address and credit card. In this case, the Confirm page includes a field for entering the card's security code, since this is normally done in the Billing stage.

Shipping Options

The main shipping page is `store.war/checkout/shipping.jsp`, which uses the `/atg/commerce/order/purchase/ShippingGroupFormHandler` to create and manage shipping groups. CRS changes the class of this component to `atg.projects.store.order.purchase.ShippingInfoFormHandler`, which is a CRS-specific subclass of the ATG Commerce `atg.commerce.order.purchase.ShippingGroupFormHandler` class.

The `store.war/checkout/shipping.jsp` page conditionally includes different options, depending on the items in the order. For example, if the order includes more than one hardgood item, the page includes a link to the `store.war/checkout/shippingMultiple.jsp` page. This page includes the `store.war/checkout/gadgets/shippingMultipleForm.jsp` gadget, which creates a form that enables customers to select shipping options for each item individually.

A customer who is logged in can select previously saved shipping addresses or enter new ones. A customer who is not logged in must manually enter shipping addresses.

Pricing the Order

Order pricing in CRS is based on the ATG Commerce pricing services described in the *ATG Commerce Programming Guide*. The process is managed by the `/atg/commerce/pricing/PricingTools` component, which calls the following pricing engines in the order listed:

- `/atg/commerce/pricing/ItemPricingEngine` -- Determines the price for each item in the order.
- `/atg/commerce/pricing/OrderPricingEngine` -- Determines the order subtotal, which includes the total price of all the items, but does not include shipping or taxes.
- `/atg/commerce/pricing/ShippingPricingEngine` -- Determines the shipping price for each shipping group in the order.
- `/atg/commerce/pricing/TaxPricingEngine` -- Determines the tax charges for the order.

Each pricing engine iterates through a set of precalculators to determine an initial price. It then applies any promotions of the corresponding type (i.e., item discount, order discount, or shipping discount) that have been granted to the customer. (See the CRS Promotions section of the Personalization and

Promotions chapter for information about the available promotions.) When one engine finishes its work, the next engine is called, until the total price is determined.

So, for example, if a customer has been granted a promotion for 10% off the order total, the `OrderPricingEngine`:

1. Invokes `/atg/commerce/pricing/calculators/OrdersSubtotalCalculator`, which computes an initial subtotal by summing the prices (previously determined by `ItemPricingEngine`) of the items in the order.
2. Applies the promotion, which reduces the order price by 10%.

`PricingTools` then invokes `ShippingPricingEngine`.

ItemPricingEngine

CRS configures `ItemPricingEngine` to use precalculators that look up prices in price lists:

```
precalculators=\
   calculators/ItemPriceListCalculator,\
   calculators/ItemPriceListSaleCalculator,\
   calculators/ConfigurableItemPriceListCalculator,\
   calculators/ConfigurableItemPriceListSaleCalculator
```

OrderPricingEngine

CRS uses `OrderPricingEngine` without any additional configuration.

ShippingPricingEngine

CRS configures `ShippingPricingEngine` to use three precalculators: `/atg/commerce/pricing/shipping/Ground`, `/atg/commerce/pricing/shipping/TwoDay`, and `/atg/commerce/pricing/shipping/NextDay`. The calculator applied to an individual shipping group is determined by the shipping method specified for that shipping group.

The `TwoDay` and `NextDay` calculators are fixed-price calculators (class `atg.commerce.pricing.FixedPriceShippingCalculator`), applying prices of $9.50 and $18.95 to the shipping group, respectively. The `Ground` calculator (class `atg.commerce.pricing.PriceRangeShippingCalculator`) determines the shipping price based on the total cost of the items in the shipping group. The shipping prices are set through the `ranges` property of the calculator:

```
ranges=00.00:14.99:4.75,\
       15.00:49.99:5.95,\
       50.00:MAX_VALUE:6.50
```

This setting means that if the total is $14.99 or less, shipping is $4.75; if the total is between $15.00 and 49.99, shipping is $5.95; and if the total is $50.00 or more, shipping is $6.50.

TaxPricingEngine

`TaxPricingEngine` is configured in ATG Commerce to use `/atg/commerce/pricing/calculators/TaxProcessorTaxCalculator` as its precalculator. This component (of class `atg.commerce.pricing.AddressVerificationTaxProcessorTaxCalculator`), has a `taxProcessor` property that is set to the component that performs or manages the actual tax calculations.

By default, this property is set to `/atg/commerce/payment/DummyTaxProcessor`, which always returns $0.00. If you run ATG Commerce with the `Cybersource` module, `taxProcessor` is instead set to `/atg/commerce/payment/CyberSourceTax`, which is of class `atg.integrations.cybersource.CyberSourceTax`. This component makes tax-calculation calls to CyberSource and parses the response. Tax is calculated for each shipping group as a whole.

The CRS-specific `Store.Cybersource` module changes the class of `/atg/commerce/payment/CyberSourceTax` to `atg.projects.store.integrations.cybersource.StoreCyberSourceTax`, and configures it to calculate tax for each item separately. This module also changes the class of `/atg/commerce/pricing/calculators/TaxProcessorTaxCalculator` to `atg.projects.store.pricing.StoreTaxProcessorTaxCalculator`, which processes the item-level tax information.

Note that to use CyberSource, you must have a valid account with CyberSource Corporation and configure your application appropriately. See the *Integrating CyberSource with ATG Commerce* section of the *ATG Commerce Programming Guide* for more information.

Payment Options

CRS supports two payment options: store credits and a single credit card. (CRS does not support using multiple credit cards to pay for an individual order.)

If a customer is logged in and has one or more store credits, the value of the credits is automatically applied toward the price of the order. If the credits are less than the total price of the order, the customer must use a credit card to pay for the remainder of the order. If the credits are more than the total price of the order, no further payment is required, and the difference between the order price and the credits is retained for use on future orders.

A customer who is not logged in can use a credit card entered during checkout for payment. A customer who is logged in also has the option of choosing a saved credit card.

The billing page, `store.war/checkout/billing.jsp`, includes the `store.war/checkout/gadget/billingForm.jsp` gadget, which contains the form used for specifying payment options. This gadget uses the `/atg/store/order/purchase/BillingFormHandler` to create the form and apply the payment options the customer specifies. The `BillingFormHandler` is of class `atg.projects.store.order.purchase.BillingInfoFormHandler`, which extends the `atg.commerce.order.purchase.PurchaseProcessFormHandler` abstract class.

Fulfillment

CRS uses the standard ATG Commerce Fulfillment system with a few extensions. (See the *Configuring the Order Fulfillment Framework* chapter of the *ATG Commerce Programming Guide*.) To run the Fulfillment system with the CRS extensions, include the `Store.Fulfillment` module in your storefront EAR file.

Simulating Normal Order-Processing Behavior

When you run CRS for evaluation or development purposes, you can simulate customer activity by browsing the site, adding items to a shopping cart, and proceeding through the checkout process. After the order is submitted, though, nothing further will happen, because no actual fulfillment takes place.

To enable order processing to continue (e.g., sending an e-mail that the order has shipped, adjusting inventory, etc.), hardgood shipping groups must be marked as fulfilled. There are two ways to make this happen:

- Mark individual shipping groups as fulfilled in the ATG Dynamo Server Admin.
- Configure the `/atg/commerce/fulfillment/HardgoodShipper` component to periodically mark all shipping groups that are pending shipment as fulfilled.

Marking an Individual Shipping Group as Fulfilled

To mark an individual shipping group as fulfilled:

1. Start the ATG Dynamo Server Admin, and click the Commerce Administration link.
2. On the Dynamo Commerce Administration page, click the Fulfillment Administration link.
3. On the Fulfillment Administration page, under List Shippable Groups, click the link to display the IDs of all the orders and shipping groups that are ready to be shipped.
4. Under Notify Fulfillment of Shipment, enter the `OrderId` and the `ShippingGroupId` for the shipping group you want to mark as fulfilled. Click Ship.

Configuring the HardgoodShipper Component

The `/atg/commerce/fulfillment/HargoodShipper` component is of class `atg.commerce.fulfillment.HardgoodShipper`, which implements the `Schedulable` interface. This component periodically checks for shipping groups whose status is PENDING_SHIPMENT and marks them as shipped.

The `HardgoodShipper` component comes preconfigured in the ATG Commerce `Fulfillment` module, but it is not enabled by default. You can enable it on a currently running ATG instance through the ATG Control Center by opening the component and clicking the Start button. Or you can configure it to start up automatically when your application starts up by adding it to the `initialServices` property of the `/Initial` component:

```
initialServices+=/atg/commerce/fulfillment/HargoodShipper
```

By default, `HardgoodShipper` is configured to run every six hours. You can change the frequency by setting the `schedule` property to a different value. See the *Core Dynamo Services* chapter of the *ATG Programming Guide* for information about setting the `schedule` property of a schedulable service.

Important: Be sure to disable this component on your production environment.

Adjusting Inventory

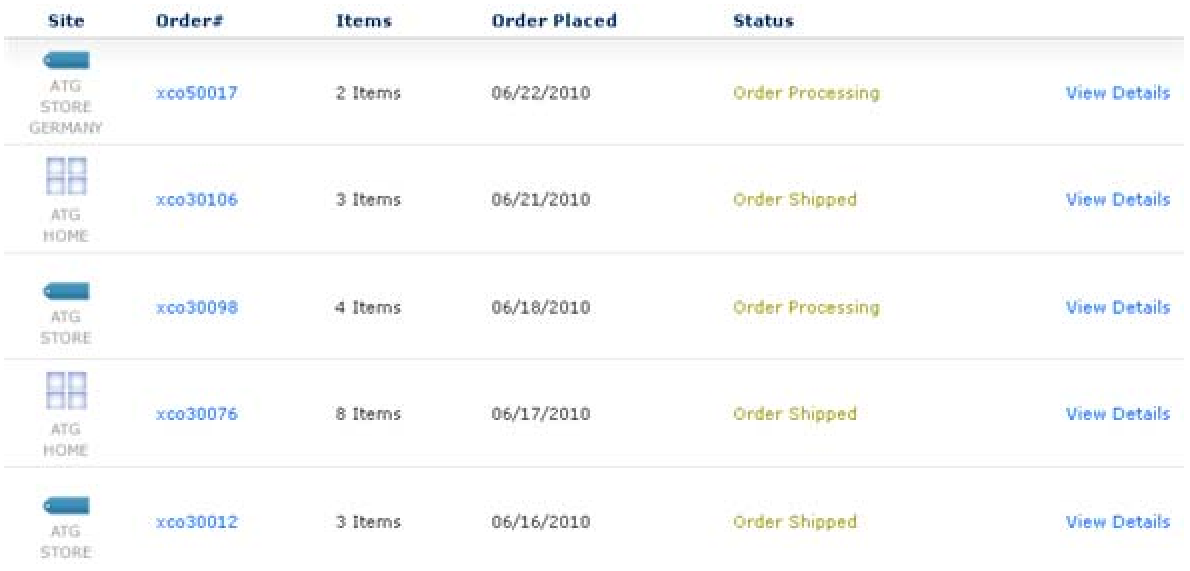
The ATG Fulfillment system adjusts inventory when items are fulfilled. If you are using a different fulfillment system that does not adjust inventory, you can configure the CRS order manager to adjust inventory on checkout.

By default, the ATG Commerce order manager, `/atg/commerce/order/OrderManager`, is of class `atg.commerce.order.SimpleOrderManager`. CRS changes the class of this component to `atg.projects.store.order.StoreOrderManager`, which is a CRS-specific subclass of `SimpleOrderManager`. `StoreOrderManager`, adds a Boolean property, `adjustInventoryOnCheckout`, which determines whether it should adjust inventory on checkout. By default, this property is set to `false`, because ATG Fulfillment handles inventory adjustment. If you are using a fulfillment system that does not adjust inventory, set this value to `true`:

```
adjustInventoryOnCheckout=true
```

Order History

Order history (available only to customers who are logged in) can be displayed by clicking the Orders link on the upper right of most pages. The Orders page, `store.war/myaccount/myOrders.jsp`, lists all orders placed by the customer, regardless of what site each order was placed on. For example:



Site	Order#	Items	Order Placed	Status	
ATG STORE GERMANY	xco50017	2 Items	06/22/2010	Order Processing	View Details
ATG HOME	xco30106	3 Items	06/21/2010	Order Shipped	View Details
ATG STORE	xco30098	4 Items	06/18/2010	Order Processing	View Details
ATG HOME	xco30076	8 Items	06/17/2010	Order Shipped	View Details
ATG STORE	xco30012	3 Items	06/16/2010	Order Shipped	View Details

The `store.war/myaccount/myOrders.jsp` page includes the `store.war/myaccount/gadgets/MyOrders.jsp` gadget, which uses the `/atg/commerce/order/OrderLookup` servlet bean to retrieve the list of orders.

The View Details links on the Order History page take the customer to the Order Details page, `store.war/myaccount/orderDetail.jsp`. Each link specifies the order to display using the `orderId` query parameter.

The Order Details page displays detailed information about the order, including a list of the items in the order, the order status, the shipping and billing information, and the site the order was placed on.

See Appendix B: Sample Data for a list of sample customers who have orders with various statuses.

8 Search Engine Optimization

The ATG platform provides support for a number of Search Engine Optimization (SEO) techniques for making pages more accessible to web spiders (also known as robots), the scripts used by Internet search engines to crawl the Web to find pages for indexing. The goal of SEO is to increase the ranking of the indexed pages in search results.

ATG Commerce Reference Store includes implementations of the SEO techniques that the ATG platform supports. These techniques include:

URL Recoding

Canonical URLs

Sitemaps

SEO Tagging

The platform support for these features is described in the *Search Engine Optimization* chapter of the *ATG Programming Guide*. This chapter describes the specific implementations in CRS, and assumes you are familiar with the discussion in the *ATG Programming Guide*.

URL Recoding

The URL recoding feature detects whether the site visitor is a web spider, and if it is, generates static URLs for links to pages, rather than the dynamic URLs that the ATG platform typically uses. A static URL can include the product and category name, which may result in higher ranking for the page in searches for that product or category. For example, the static URL (minus the host name and context root) for a product detail page might look similar to this:

```
/jump/Crystal-Decanter/productDetail/Glassware/xprod2081/cat10024
```

This page will have a higher ranking for a search term like "Crystal Decanter" than the equivalent dynamic URL, which looks similar to this:

```
/browse/productDetailSingleSku.jsp?productId=xprod2081&categoryId=cat10024
```

To implement the URL recoding feature, CRS includes components that:

- detect whether a site visitor is a spider or not
- render either static URLs or dynamic URLs, depending on the type of visitor

- specify how to translate a dynamic URL to its static equivalent

In addition, it includes components that translate static URLs in incoming requests (issued when the customer clicks a link returned by a Web search engine) to the dynamic URLs that the ATG platform understands.

These components are described below.

Rendering URLs

The `atg.repository.seo.ItemLink` servlet bean takes a repository item as input and uses a URL template to construct a static or dynamic link to that item, depending on the value of the HTTP request's `User-Agent` property.

CRS includes two `ItemLink` components, `CatalogItemLink` and `ProductLookupItemLink`, and their associated template mapper and template components. All of these components are found in the `/atg/repository/seo/` Nucleus folder.

CRS also uses the `atg.repository.seo.BrowserTyperDroplet` servlet bean to determine if a site visitor is a web spider. CRS includes a component of this class, `/atg/repository/seo/BrowserTyperDroplet`, whose `browserTypes` property is set to `robot`. The servlet bean examines the current request, and if it determines that the source of the request is a robot or spider, it sets its `browserType` output parameter to `robot`; otherwise it sets it to `other`. This information is used on the page when caching page content and adding parameters to links.

CatalogItemLink

The `CatalogItemLink` component is the main instance of the `ItemLink` servlet bean used by CRS. This component is configured as follows:

```
itemDescriptorNameToMapperMap=\
        product=/atg/repository/seo/ProductTemplateMapper,\
        category=/atg/repository/seo/CategoryTemplateMapper

defaultRepository=/atg/commerce/catalog/ProductCatalog
defaultItemDescriptorName=product

siteManager=/atg/multisite/SiteManager
```

The `itemDescriptorNameToMapperMap` property determines the template mapper to use, based on the type of repository item passed to the servlet bean. The item type is either `product` (for a link to a product detail page) or `category` (for a link to a category page).

Each template mapper component has a `templates` property that specifies one or more templates to use for rendering static URLs, and a `defaultTemplate` property that specifies the template to use for rendering dynamic URLs. The `templates` property of the `ProductTemplateMapper` component is set to `/atg/repository/seo/ProductIndirectTemplate`. This template is used to construct static URLs for product detail pages if the site visitor is a spider. Similarly, the `templates` property of the

`CategoryTemplateMapper` is set to `/atg/repository/seo/CategoryIndirectTemplate`, which is used to construct static URLs for category pages.

The `urlTemplateFormat` property of `ProductIndirectTemplate`, which specifies the format of static URLs for product detail pages, is set to:

```
/jump/{item.displayName}/productDetail/{item.parentCategory.displayName}/
{item.id}/{item.parentCategory.id}
```

The resulting URL thus includes the display name of the product and its default parent category, so the Web search engine can index these terms. In addition, it includes the repository IDs of the product and category. These IDs, along with the `/jump` prefix, enable the SEO jump servlet to reconstruct the dynamic URL, as discussed in Translating Static URLs.

ProductLookupItemLink

When the `CatalogItemLink` component constructs a static URL for a product detail page, it determines the category to use in the URL by looking up the product's default parent category. But if the product has multiple parent categories, and is being accessed in a category that is not the default parent, using the default parent category in the URL may not be desirable.

To handle this situation, CRS includes another instance of the `ItemLink` servlet bean, `ProductLookupItemLink`. This component is used for generating links to product detail pages in cases where a product is not being viewed in its default category. It uses the `ProductLookupIndirectTemplate` to generate static URLs. The `urlTemplateFormat` property of this template is set to:

```
/jump/{item.displayName}/productDetail/{item.parentCategory.displayName}/
{item.id}/{categoryId}
```

The resulting URL includes the display name of the default parent category, but rather than the default parent category's ID, it instead includes the ID of the category that the product is being accessed from. This enables the Web search engine to associate the product with its default category in the index, while still allowing the SEO jump servlet to reconstruct the dynamic URL specifying the (non-default) category that the product was accessed from. The `ProductLookupIndirectTemplate` can obtain the category ID of the non-default parent category from a page parameter, or it can be explicitly set on the page.

Translating Static URLs

When a spider visits an ATG Commerce site that uses URL recoding, the site typically renders static URLs for pages accessed by the spider. These static URLs are used as links in results returned by the Web search engine, but they are not actually valid URLs on ATG Commerce sites. So when a user clicks one of these links, the site must examine the request and translate it into a dynamic URL that can be interpreted by the ATG request-handling components. This translation is done by the SEO jump servlet, which is declared in the `store.war/WEB-INF/web.xml` file as a URI-mapped servlet:

```
<servlet>
  <servlet-name>MappedJumpServlet</servlet-name>
  <servlet-class>
```

```
    atg.repository.seo.MappedJumpServlet
  </servlet-class>
  <init-param>
    <param-name>loggingDebug</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>jumpServlet</param-name>
    <param-value>
      ctx:dynamo:/atg/dynamo/servlet/dafpipeline/JumpServlet
    </param-value>
  </init-param>
</servlet>

...

<servlet-mapping>
  <servlet-name>MappedJumpServlet</servlet-name>
  <url-pattern>/jump/*</url-pattern>
</servlet-mapping>
```

The servlet’s `templates` property, which lists the indirect URL templates to examine to determine how to translate the URL, is set to:

```
templates=\
  /atg/repository/seo/CategoryIndirectTemplate,\
  /atg/repository/seo/ProductIndirectTemplate
```

Canonical URLs

CRS is configured to render a `link` tag on each page that specifies the canonical URL for the page. When spiders for certain Web search engines crawl a page, they record the page’s URL as the value specified in the tag, rather than the actual URL that was used to access the page.

The `link` tag for a page looks similar to this:

```
<link rel="canonical" ref="http://www.example.com:80/crs/storeus/jump/
Dotted-Repp-Tie/productDetail/For-Him/xprod1001/cat50067 " />
```

To code your JSPs to render canonical URLs, use the URL recoding feature described in URL Recoding. The canonical URL generated for a page is similar to the static URL rendered for web spiders by an indirect URL template. The canonical URL is always static, regardless of whether the page is accessed by a spider or a human user. That way, if a spider happens to access a page using a dynamic URL (e.g., by following a link from another page), it will still see (and record) the static URL it finds in the `link` tag. As with the URL recoding feature, when a user accesses a page via a static URL, the SEO jump servlet translates it back to its dynamic equivalent for processing.

To render canonical URLs, you use the `atg.repository.seo.CanonicalItemLink` servlet bean. CRS includes a component of this class, `/atg/repository/seo/CanonicalItemLink`, which is configured as follows:

```
itemDescriptorNameToUrlTemplateMap=\
        product=/atg/repository/seo/ProductIndirectTemplate,\
        category=/atg/repository/seo/CategoryIndirectTemplate

defaultRepository=/atg/commerce/catalog/ProductCatalog
defaultItemDescriptorName=product

siteManager=/atg/multisite/SiteManager
```

The `store.war/global/gadgets/canonicalTag.jsp` gadget invokes the `CanonicalItemLink` servlet bean to render the canonical URL for a product detail page:

```
<dsp:droplet name="/atg/repository/seo/CanonicalItemLink">
  <dsp:param name="id" param="productId" />
  <dsp:param name="itemDescriptorName" value="product" />
  <dsp:param name="repositoryName" value="/atg/commerce/catalog/ProductCatalog" />
  <dsp:oparam name="output">
    <dsp:getvalueof var="pageUrl" param="url" vartype="java.lang.String" />
    <link rel="canonical" href="${httpLink}${pageUrl}" />
  </dsp:oparam>
</dsp:droplet>
```

Sitemaps

CRS includes components for creating sitemaps to help web spiders access site pages. These components create a sitemap index file named `siteindex.xml` with references to three sitemap files:

- `categorySitemap.xml` is the sitemap for category pages.
- `productSitemap.xml` is the sitemap for product detail pages.
- `staticSitemap.xml` is the sitemap for static pages, such as the About Us page.

To generate `categorySitemap.xml` and `productSitemap.xml`, CRS includes two components of class `atg.commerce.sitemap.CatalogSitemapGenerator`, `/atg/sitemap/CategorySitemapGenerator` and `/atg/sitemap/ProductSitemapGenerator`. (`CatalogSitemapGenerator` is a subclass of `atg.sitemap.DynamicSitemapGenerator` that adds logic for determining the catalog to use for a site.) `CategorySitemapGenerator` generates the entries in the `SitemapRepository` that are used to create the `categorySitemap.xml` file. `ProductSitemapGenerator` generates the entries in the `SitemapRepository` that are used to create the `productSitemap.xml` file.

In addition to the `CatalogSitemapGenerator` components, CRS includes the following sitemap-generation components, which are also found in the `/atg/sitemap/` Nucleus folder:

- `StaticSitemapGenerator` -- This component is of class `atg.sitemap.StaticSitemapGenerator`. It generates the entries in the `SitemapRepository` that are used to create the `staticSitemap.xml` file.
- `SitemapIndexGenerator` -- This component is of class `atg.sitemap.SitemapIndexGenerator`. It generates the entries in the `SitemapRepository` that are used to create the `siteindex.xml` file.
- `SitemapGeneratorService` -- This component is of class `atg.sitemap.SitemapGeneratorService`. It manages the process of scheduling and invoking the sitemap generator components.
- `SitemapWriterService` -- This component is of class `atg.sitemap.SitemapWriterService`. It writes out the entries from the `SitemapRepository` to create the sitemap and sitemap index XML files.

Enabling Sitemap Generation

The components described above come preconfigured with most of the settings they need to generate and write out sitemaps for CRS. To enable sitemap generation, you need to set the `warDir` property of the `SitemapGeneratorService` component to the operating-system pathname of the deployed WAR file that the sitemap is generated for. For example:

```
warDir=\
  C:\jboss-eap-5.0\jboss-as\server\atg\deploy\ATG.ear\store.war
```

Once you set this value, you can manually invoke sitemap generation in the ATG Dynamo Server Admin. If you want sitemap generation to run automatically on a regular schedule, you must also set the `schedule` property of the `SitemapGeneratorService` and `SitemapWriterService` components. See the *Core Dynamo Services* chapter of the *ATG Programming Guide* for information about setting the `schedule` property of a schedulable service.

SEO Tagging

Web search engines base their rankings of pages partly on the words that appear in certain HTML tags, particularly `<meta>` tags and the `<title>` tag. A common SEO technique is to list key search terms in those tags, to raise the ranking of the pages for those terms.

CRS pages include logic for rendering `<title>`, `<meta name=description ...>`, and `<meta name=keywords ...>` tags. This logic is found in the `store.war/global/gadgets/metaDetails.jsp` gadget, which is invoked through the `<crs:pageContainer>` tag. For example:

```
<crs:pageContainer>
    <jsp:attribute name="SEOTagRenderer">
```

```
      <dsp:include page="/global/gadgets/metaDetails.jsp" flush="true">
        <dsp:param name="catalogItem" param="product" />
      </dsp:include>
    </jsp:attribute>
</crs:pageContainer>
```

Depending on the context, the gadget can be invoked either with or without the `catalogItem` parameter. If the `catalogItem` parameter is omitted, `metaDetails.jsp` renders the tags based only on values stored in the `SEORepository`. Using the servlet path of the current request to look up the `key` property, and the site ID of the current site to look up the `sites` property, it finds the correct `SEOTags` item. It then uses the values of this item's `description`, `keywords`, and `title` properties to construct the `<meta>` tags and the `<title>` tag.

If the `catalogItem` parameter is included, `metaDetails.jsp` renders the tags by combining the values stored in the repository with values taken from the item's properties. For example, if the item is a product, the `<meta name=description ...>` tag is constructed by concatenating the `description` property of the `SEOTags` item with the `longDescription` property of the product.

9 ATG Search Integration

The ability to search for products and SKUs is an essential feature of almost every commerce site. Searching enables customers to quickly track down specific items without having to navigate through the site's category structure. More sophisticated search techniques can be used to find products based on criteria such as price, color, or manufacturer.

Search in ATG Commerce Reference Store is implemented through integration with ATG Search. ATG Search supports a number of advanced search techniques, including natural language queries, faceted navigation, and the ability to tailor results to individual users (Search Merchandising).

If ATG Search is not present, CRS determines this and falls back to using ATG's basic search facility. In this case, search queries are limited to keywords and simple search strings, and facets and search merchandising are not available.

CRS determines if ATG Search is available based on the value of the `atgSearchInstalled` property of the `/atg/store/StoreConfiguration` component. By default, this property is set to `false`, meaning ATG Search is not available. The `Store.Search.Query` module sets the value of this property to `true`.

This chapter includes the following topics:

Indexing the Product Catalog

The indexing process for Commerce sites is configured through the `/atg/commerce/search/ProductCatalogOutputConfig` component. This component's XML definition file is configured to create an XHTML document for each product in the product catalog.

The XML definition file used in CRS is an XML combination of the definition file included in the `DCS.Search.Index` module and CRS-specific extensions added in the `Store.Search.Index` module. The CRS extensions add CRS-specific catalog properties to the index, and configure the index to get price values from price lists, as described in Indexing and Querying Price Data. For internationalized sites, the

`Store.Estore.International.Search.Index` also adds configuration for including content in multiple languages.

Each product in the CRS catalog repository has a `siteIds` property whose value is a comma-separated list of the sites the product appears on. Certain products appear only on ATG Store US and ATG Store Germany, other products appear only on ATG Home, and some products appear on all three sites.

The XHTML document generated for a product includes a special `$siteId` metadata property whose value is based on the value of the product's `siteIds` property. The document includes a separate tag for each site listed in the `siteIds` property. For example, the XHTML document for a product found on ATG Store US and ATG Store Germany includes:

```
<meta name="atg:string,index:$siteId" content="storeSiteUS" />
<meta name="atg:string,index:$siteId" content="storeSiteDE" />
```

This information is used to determine which results to return when a query is issued from a site, as discussed below.

Issuing Search Queries

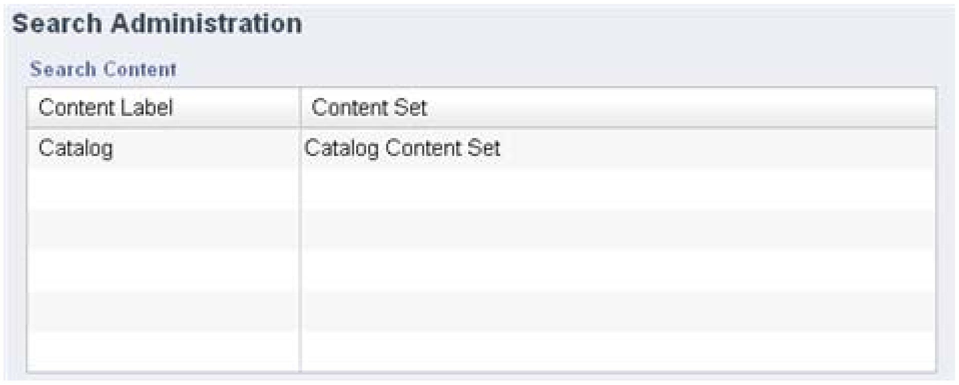
For issuing search queries, CRS uses the `/atg/commerce/search/catalog/QueryFormHandler` and `/atg/commerce/search/catalog/QueryRequest` components. The `Store.Search.Query` module modifies the configuration of `QueryRequest` to add constraints that do the following:

- Restrict search results to products in a catalog's root navigation category, so products that are not part of the navigational hierarchy (such as Gift Wrap) are not returned.
- Restrict search results to products whose `startDate` is either in the past or is undefined, and whose `endDate` is either in the future or is undefined.

Querying in a Multisite Environment

CRS is configured with a single ATG Search project, which has a content label of Catalog. This content label is associated with one content set, Catalog Content Set, which represents the Commerce catalog repository.

In the Site Administration utility, Catalog Content Set is associated with each of the CRS sites. For example, the configuration for each site looks like this:



Search Administration

Search Content

Content Label	Content Set
Catalog	Catalog Content Set

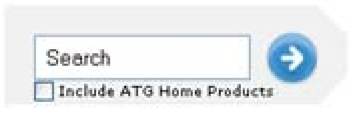
The content label and target type for queries are set using the `contentLabels` and `targetType` properties of the `/atg/commerce/search/catalog/QueryRequest` component. CRS uses the default setting for these properties:

```
contentLabels=Catalog
targetType=Production
```

The `/atg/search/routing/DynamicTargetGenerator` component determines the search environment, based on the values of `contentLabels` and `targetType`. In addition, it uses the `/atg/search/routing/command/search/DynamicTargetSpecifier` component to determine the sites that the query should apply to, and uses this information to configure the `/atg/search/routing/command/search/MultisiteConstraint` component with the appropriate site constraint. The constraint ensures that the query returns only those XHTML documents that contain `meta` tags for the appropriate sites.

Implementing a Multisite Search Control

CRS includes a multisite search control that allows customers to search only the current site or to search the sites that share a shopping cart with the current site. For example:



The following code in `store.war/atgsearch/gadgets/atgSearch.jsp` creates this search control. Based on what the user selects, the code sets the value of the `searchSiteIds` property of the `/atg/store/profile/SessionBean` component. The `DynamicTargetSpecifier.siteIdsArray` property, which actually specifies the sites to search, is linked to the value of `SessionBean.searchSiteIds`:

```
siteIdsArray^=/atg/store/profile/SessionBean.searchSiteIds
```

```
<dsp:droplet name="SharingSitesDroplet">
  <dsp:param name="shareableTypeId" value="atg.ShoppingCart" />
  <dsp:param name="excludeInputSite" value="true" />

  <dsp:oparam name="output">
    <%-- Sort sites --%>
    <dsp:getvalueof var="sites" param="sites" />
    <dsp:getvalueof var="size" value="${fn:length(sites)}" />

    <dsp:droplet name="ForEach">
      <dsp:param name="array" param="sites" />
      <dsp:setvalue param="site" paramvalue="element" />

      <%-- current site, selected and disabled --%>
      <dsp:oparam name="outputStart">

        <dsp:input bean="SessionBean.searchSiteIds" type="hidden"
           value="${currentSiteId}" priority="10" />
      </dsp:oparam>

      <%-- other sites --%>
      <dsp:oparam name="output">
        <dsp:getvalueof var="siteId" param="site.id" />
        <div>
          <dsp:input bean="SessionBean.searchSiteIds" type="checkbox"
             value="${siteId}" priority="10" id="otherStore" checked="false" />
          <label for="otherStore">
            <fmt:message key="search.otherStoresLabel">
              <fmt:param>
                <dsp:valueof param="site.name" />
              </fmt:param>
            </fmt:message>
          </label>
        </div>
      </dsp:oparam>
    </dsp:droplet>
  </dsp:oparam>

  <dsp:oparam name="empty">
    <dsp:input bean="SessionBean.searchSiteIds" type="hidden"
       value="${currentSiteId}" priority="10" />
  </dsp:oparam>
</dsp:droplet>
```

Indexing and Querying Price Data

The CRS catalog uses separate price lists for list prices and sale prices. Every product has a list price, which is used if the product does not have a sale price. If a product has a sale price, the sale price overrides the list price. These price lists are assigned to a user profile using the profile's `priceList` and `salePriceList` properties. These properties are typically set as follows:

```
priceList=listPrices
salePriceList=salePrices
```

As discussed in the *ATG Search Administration Guide*, there are three key aspects to working with price lists in ATG Search:

- Because prices in price lists are not stored in properties of the product or SKU, CRS uses the `price` metadata property created by the ATG Commerce component `/atg/commerce/search/PriceListPropertyProvider` to represent the price data stored in price lists.
- The XHTML document created for each product includes the list price and, if there is one, the sale price.
- The search client determines the logic for selecting which price to use, and specifies this logic in the query.

Indexing Price List Data

When the CRS catalog is indexed, the item prices are read from the price lists and used to construct `meta` tags in the XHTML documents. A separate `meta` tag is created for each price list, and the property name in the tag identifies the price list the tag is associated with. The ATG Commerce property accessor `/atg/commerce/search/PriceListMapPropertyAccessor` is used to read the prices from the price lists, to get the values to include in the `meta` tags.

The `Store.Search.Index` module specifies this property accessor by adding the following line to the XML definition file of the `/atg/commerce/search/ProductCatalogOutputConfig` component:

```
<property name="price" type="float"
   property-accessor="pricePropertyAccessor"
      is-non-repository-property="true" />
```

The XHTML output for a product might look like this:

```
<meta name="atg:float:childSKUs.price@salePrices" content="5.99" />
<meta name="atg:float:childSKUs.price@listPrices" content="7.99" />
```

Specifying the Price List in a Search Request

The Price facet in CRS is based on the `price` property created by the `PriceListPropertyProvider`. For search requests that return this facet, the search client must determine which `meta` tag in the XHTML output to use to obtain the value for `price`.

To do this, the /atg/commerce/search/PriceListPropertyMapping component checks the user's profile to determine which price lists are assigned to it. It then creates an XML tag for mapping the price property in the query to the XHTML meta tags associated with those price lists. The XML tag is stored in the PriceListPropertyMapping component's priceMapping property. The value of this property is typically:

```
<propertyMapping>
  price,childSKUs.price@salePrices,childSKUs.price@listPrices
</propertyMapping>
```

This tag tells ATG Search how to get the value for the price property:

- If an XHTML document includes a meta tag associated with the salePrices price list, use the value from that tag.
- Otherwise, use the value from the meta tag associated with the listPrices price list.

To include the XML tag in the search request, the propertyMappings property of the /atg/commerce/search/ProductCatalogParserOptions component is linked to the PriceListPropertyMapping.priceMapping property:

```
propertyMappings^=\
   /atg/commerce/search/PriceListPropertyMapping.priceMapping
```

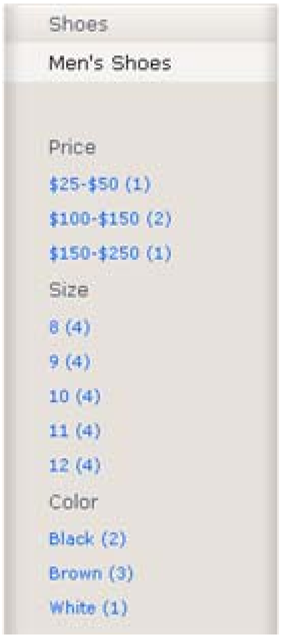
Faceted Search

CRS defines six facets, all of which are global facets:

- Size -- This facet has childSKUs.size as its faceting property.
- Color -- This facet has childSKUs.color as its faceting property.
- Wood Finish -- This facet has childSKUs.woodFinish as its faceting property.
- Features -- This facet has features.displayName as its faceting property, where features is an array of feature items.
- Price -- This facet gets its value from the price property created by PriceListPropertyProvider. See Indexing and Querying Price Data for more information about this property.
- Category -- This facet has ancestorCategories.$repositoryId as its faceting property. This Category facet itself is not displayed on the site (it is explicitly hidden by the JSP code that displays the facets), but it is used in determining which other facets to display as a customer navigates the catalog.

Determining the Facets and Results to Display

As a customer navigates the catalog, CRS displays facet values on the left side of the page. For example:



Note that even though all of the facets are global, some of them may not display for certain products. For example, the Wood Finish facet applies only to products whose SKUs are of the `furniture-sku` subtype, because only those SKUs have a `woodFinish` property.

ATG Search determines which facets and results to return based on the facet trail submitted with the faceted search request. When a customer enters a free-text search query, a new facet trail is created containing the SRCH facet. So, for example, if the customer searches for “feather,” the facet trail is:

```
SRCH:feather
```

ATG Search returns results from the free-text search and displays the facets associated with the lowest-level ancestor category that is common to all of the returned items. (“Lowest-level” means furthest from the root category, which is considered the top of the catalog hierarchy.) This behavior is enabled by the following properties of the `/atg/commerce/search/catalog/QueryRequest` component:

```
refineConfig=$map
refineConfigMapProperty=ancestorCategories.catalogSpecificId
```

Setting the `refineConfig` property to `$map` instructs ATG Search to select the refinement configuration by finding a metadata property value common to all of the results; `refineConfigMapProperty` specifies which metadata property to use.

As the customer makes facet selections, they are appended to the facet trail, so that the results returned are only those products that satisfy both the text query and the facet selections. So, if the customer searches for “skirt” and then selects the Green value for the Color facet, the facet trail looks something like this:

```
SRCH:skirt:1002:Green
```

The facet trail is reset if the customer uses category navigation (e.g., selects Home Accents -> Clocks from the category bar near the top of the page), or enters a new free-text search.

Search Merchandising

CRS does not come preconfigured with any search configurations for Search Merchandising. For information about creating search configurations, see the *ATG Merchandising Administration Guide* and the *ATG Merchandising Guide for Business Users*.

10 ATG Commerce Service Center Integration

This chapter describes how to integrate ATG Commerce Reference Store with ATG Commerce Service Center (CSC), a customer-service application used by call center agents to assist customers on ATG Commerce sites. It discusses how to set up CSC so agents can perform support tasks for CRS storefronts. In addition, it describes how to configure CRS to work with the CSC/Click to Call integration, which enables CRS customers to initiate phone calls with CSC agents.

This chapter includes the following sections:

Running ATG Commerce Service Center with CRS

To run ATG Commerce Service Center with CRS, follow the instructions in the *ATG Commerce Service Center Installation and Programming Guide* for configuring CSC. When you assemble your CSC EAR file, be sure to include the `Store.DCS-CSR` module from the CRS distribution. This module adds configuration to CSC for working with CRS.

For example, the command to assemble the CSC EAR file might be:

```
runAssembler DCSCSR.ear –m DafEar.Admin Store.Storefront DCS-CSR
  Store.DCS-CSR Fullfillment
```

Integrating CRS with ATG Click to Call

CRS pages can incorporate links to ATG Click to Call. When a customer clicks a Click to Call link, a popup window prompts for the customer’s phone number. When the phone number is submitted, Click to Call initiates a call between a CSC agent and the customer, and displays the customer’s order information and profile on the agent’s desktop.

To use CRS with ATG Click to Call:

- Configure the links using the ATG Optimization Console. For information about how to do this, see the *ATG Commerce Reference Store Technical Community*.
- Include the `DCS.ClickToConnect` module in your storefront EAR file. For example, the command to assemble the storefront EAR file might be:

```
runAssembler ATGStore.ear –m Store.Storefront DCS.ClickToConnect
  Fullfillment
```

Invoking the ADC Pipeline

To enable the ADC processor pipeline, which automatically incorporates the Click to Call links in pages, the `web.xml` file for `store.war` includes the following:

```
<filter>
  <filter-name>ADCDataInsertFilter</filter-name>
  <filter-class>
    atg.filter.ConditionalDelegatingFilter
  </filter-class>
  <init-param>
    <param-name>targetFilter</param-name>
    <param-value>
      atg.adc.filter.ADCDataInsertFilter
    </param-value>
  </init-param>
  <init-param>
    <param-name>mimeTypes</param-name>
    <param-value>text/html</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>ADCDataInsertFilter</filter-name>
  <url-pattern>*.jsp</url-pattern>
</filter-mapping>
```

These settings specify that the `atg.filter.ConditionalDelegatingFilter` should run the `atg.adc.filter.ADCDataInsertFilter` if it is present. The `ADCDataInsertFilter` invokes the processor pipeline that inserts the Click to Call links. The filter is part of the `ADC` module, which is automatically included in the assembled EAR file when you specify the `DCS.ClickToConnect` module.

Rendering the Links

Click to Call links appear on many pages, including category pages, product details pages, the order history page, the shopping cart page, and the Contact Us page. Different pages use different images as link icons; the image used for a specific link is configured in the ATG Optimization Console.

Links are inserted in CRS pages by the `store.war/navigation/gadgets/clickToCallLink.jsp` gadget:

```
<dsp:page>
  <%-- If the click to call feature is disabled, this entire div can be safely
       bypassed. --%>
  <dsp:importbean var="c2cConfig" bean="/atg/clicktoconnect/Configuration" />
  <c:if test="${not empty c2cConfig}">
    <%-- Add c2c scripts etc to output. --%>
    <dsp:droplet name="/atg/adc/droplet/InsertTag">
      <dsp:param name="location" value="body" />
      <dsp:oparam name="output">
        <dsp:valueof param="data" converter="valueishtml" />
      </dsp:oparam>
    </dsp:droplet>
    <%-- And display the base div. --%>
    <dsp:getvalueof var="pageName" param="pageName" />
    <div id="atg_store_c2c_${pageName}">&nbsp;</div>
  </c:if>
</dsp:page>
```

This gadget checks for the existence of the `/atg/clicktoconnect/Configuration` component. This component is part of the `ClickToConnect` module, which is automatically included in the assembled EAR file when you specify the `DCS.ClickToConnect` module. If the component exists, the gadget then:

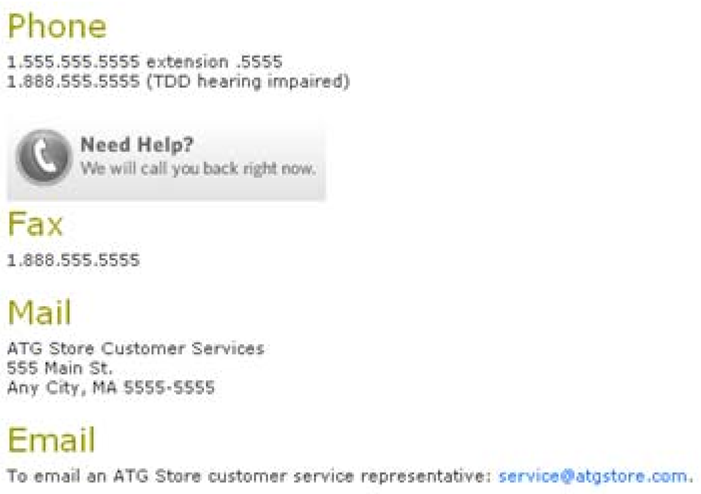
- Invokes the `/atg/adc/droplet/InsertTag` droplet, which renders a link to a JavaScript function for initiating a call.
- Renders the `<div>` element that the link icon is associated with.

For example, on the Contact Us page, this code renders:

```
<script type="text/javascript" src="/clicktoconnect/scripts/c2c.js"></script>

<div id="atg_store_c2c_contactUs">&nbsp;</div>
```

The link appears on the Contact Us page like this:



Phone

1.555.555.5555 extension .5555
1.888.555.5555 (TDD hearing impaired)

Need Help?
We will call you back right now.

Fax

1.888.555.5555

Mail

ATG Store Customer Services
555 Main St.
Any City, MA 5555-5555

Email

To email an ATG Store customer service representative: service@atgstore.com.

11 Internationalization

This chapter discusses how ATG Commerce Reference Store (CRS) implements internationalization features for its three sites. It includes the following topics:

Running the International Module

Internationalization functionality is implemented in CRS through the `Store.EStore.International` module, located in `<ATG10dir>/CommerceReferenceStore`. If you choose the Internationalization add-on when configuring CRS through CIM, this module is included for you in the resulting EAR files, along with the required data and database schema changes. The `International` module is necessary for CRS sites that support multiple languages or multiple countries. If you do not install the `International` module, your production instance of CRS will include the English versions of ATG Store US and ATG Home only. You will not see ATG Store Germany or the Spanish translations for ATG Store US and ATG Home.

Working with Internationalized Stores

CRS incorporates internationalization features in two ways:

- It includes two related regional stores, ATG Store US and ATG Store Germany, that share a catalog.
- It includes translations for multiple languages. ATG Store US and ATG Home have English and Spanish translations while ATG Store Germany has English and German translations.

The related regional stores described in the first bullet above are implemented using ATG's multisite functionality. Detailed information on this implementation is included in the Multisite Features chapter. ATG recommends that you read the *Multisite Features* chapter first, before continuing with this chapter. The remainder of this chapter, along with Appendix A: Implementing a Related Regional Store, focus on how translations work in CRS and how to create a new related regional store.

Note: Related regional stores are also called country stores.

Determining a Customer's Locale

A customer's locale determines the language that the customer sees when viewing a site in CRS. This section describes how a customer's locale is determined for each request.

Setting a customer's locale is implemented through a series of includes. To start, an `include-prelude` statement in the `/WEB-INF/web.xml` file specifies that the `store.war/includes/prelude.jspf` page fragment should be included for all CRS JSP pages. `prelude.jspf`, in turn, includes the `store.war/includes/context.jspf` page fragment. This fragment contains code for identifying and setting the customer's locale and it looks like this:

```
<%-- Identify and set customer's locale --%>
<dsp:importbean var="requestLocale" bean="/atg/dynamo/servlet/RequestLocale" />
<fmt:setLocale value="${requestLocale.locale}"/>
```

The `context.jspf` fragment invokes the `/atg/dynamo/servlet/RequestLocale` component included in the `Store.EStore.International` module. `RequestLocale` is of class `atg.projects.store.servlet.StoreRequestLocale`, which extends the `atg.userprofiling.ProfileRequestLocale` class with the following logic that determines a user's locale:

- First, `RequestLocale` determines if a `locale` parameter is included in the HTTP request. The `locale` parameter is embedded in the links associated with the language picker on the `store.war/navigation/gadgets/languages.jsp` gadget. When a customer clicks a language in the language picker, an appropriate `locale` is sent in the ensuing HTTP request. (See Rendering the Language Picker in the Multisite Features chapter for more details on the requests generated by the language picker.)
- If a `locale` is not present in the HTTP request, `RequestLocale` looks to see if a `siteId` is included in the request. If a `siteId` is included, `RequestLocale` uses the default locale for that site, as defined by the `defaultLanguage` property in the site's configuration.
- If neither a `locale` nor a `siteId` parameter is present in the HTTP request, `RequestLocale` retrieves the `locale` from the customer's profile.
- If a `locale` is not present in the customer's profile, `RequestLocale` retrieves the `locale` from the browser's `userPrefLanguage` cookie.

- If the `userPrefLanguage` cookie is not present, `RequestLocale` uses the browser's default locale.
- If the browser does not have a default locale, `RequestLocale` uses its own default setting, as specified by the optional `defaultRequestLocale` property in the `RequestLocale.properties` file.
- Finally, if `RequestLocale` doesn't have a default setting, it uses the JVM's locale default.

Once a locale is determined, `RequestLocale` sets the `locale` property in the customer's profile and writes a `userPrefLanguage` cookie to the browser that specifies the customer's locale.

The `context.jspf` page fragment also calls the `fmt:setLocale` tag from the JavaServer Pages Standard Tag Library (JSTL). This tag sets the locale for any additional `fmt` tags called for the remainder of the request, such as the `fmt:message` tags that are used for displaying localized strings (see Displayable String Translations for more information).

Note: For more information on JTSL tags, see http://java.sun.com/products/jsp/jstl/1.1/docs/tlddocs/index.html.

Displaying Translations

CRS has three mechanisms for displaying translated data, one each for:

- Strings that are not stored in repository items
- Error messages
- Repository items that have properties that require translation

The following section describes these mechanisms in detail.

Displayable String Translations

CRS uses a set of resource files to store translations for any displayable strings that are not stored in repository items. The resource files, located in `<ATG10dir>/CommerceReferenceStore/store/EStore/lib/classes.jar`, include:

- `/atg/projects/store/web/WebAppResources.properties` for English translations
- `/atg/projects/store/web/WebAppResources_de.properties` for German translations
- `/atg/projects/store/web/WebAppResources_es.properties` for Spanish translations

The resource bundle that is used is determined by a customer's current locale.

Determining Which Resource Bundle to Use

To understand which resource bundle gets used, we return to the `/includes/context.jspf` page fragment:

```
<dsp:importbean var="requestLocale" bean="/atg/dynamo/servlet/RequestLocale" />
<fmt:setLocale value="${requestLocale.locale}"/>

<dsp:importbean bean="/atg/multisite/SiteContext"/>
<dsp:getvalueof var="resourceBundle" bean="SiteContext.site.resourceBundle" />

<dsp:importbean bean="/atg/store/StoreConfiguration" />
<dsp:getvalueof var="defaultResourceBundle"
                bean="StoreConfiguration.defaultResourceBundle" />

<%-- Set the default resource bundle location from the site context. If it
doesn't exist in the site context, get it from StoreConfiguration.properties file.
--%>
<c:choose>
  <c:when test="${not empty resourceBundle}">
    <fmt:setBundle basename="${resourceBundle}"/>
  </c:when>
  <c:otherwise>
    <fmt:setBundle basename="${defaultResourceBundle}"/>
  </c:otherwise>
</c:choose>
```

After the `RequestLocale` component determines the correct `locale` for a customer (as described in Determining a Customer's Locale), `context.jspf` invokes two components:

- The `/atg/multisite/Sitecontext` component, from which `context.jspf` retrieves the current site's `resourceBundle` property. If a value for this property exists, it becomes the default resource bundle.
- The `/atg/store/StoreConfiguration` component, from which `context.jspf` retrieves the `defaultResourceBundle` property. This value becomes the default resource bundle if the current site's `resourceBundle` property is null.

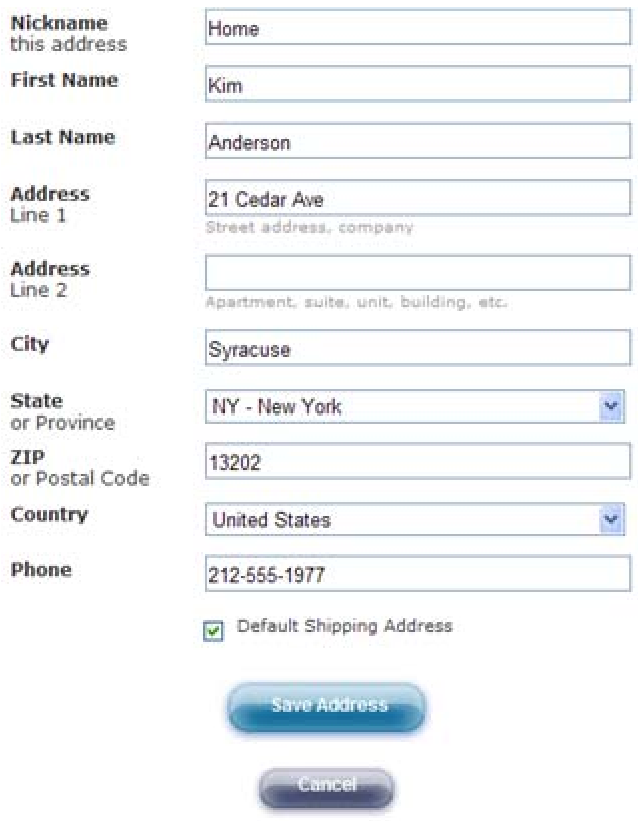
All three sites in CRS have their `resourceBundle` property configured to `atg.projects.store.web.WebAppResources`. CRS also configures the `StoreConfiguration.defaultResourceBundle` property to `atg.projects.store.web.WebAppResources`, so, regardless of where the value comes from, the default resource bundle for CRS is `atg.projects.store.web.WebAppResources`.

The default resource bundle is tied to any locale-specific sub-bundles through a naming convention. Specifically, the default bundle `WebAppResources` is tied to two sub-bundles, `WebAppResources_`**de** for the German translations and `WebAppResources_`**es** for the Spanish translations. The name extensions for a sub-bundle must include an underscore and a two-letter, lower-case language code as defined by ISO 639.

The `fmt:setBundle` tag uses both the customer's current `locale` and the default resource bundle to determine which resource bundle to set for the JSP page. `fmt:setBundle` has internal logic that allows it to compare the language portion of the current `locale` to the filename extensions on any sub-bundles that share the default resource bundle's filename. For example, the German locale `de` is matched to the `WebAppResources_de` resource bundle. This logic allows the `fmt:setBundle` tag to choose from among a collection of locale-specific sub-bundles. If it cannot find a matching bundle, `fmt:setBundle` uses the default resource bundle which, for CRS, contains the English translations.

Displaying the Strings

CRS JSP pages use the `fmt:message` tag to display strings from the current resource bundle. For example, for a customer viewing ATG Store US or ATG Home in English, a Nickname label appears on the `store.war/myaccount/gadgets/addressEdit.jsp` gadget, shown below:



Nickname this address	Home
First Name	Kim
Last Name	Anderson
Address Line 1	21 Cedar Ave Street address, company
Address Line 2	 Apartment, suite, unit, building, etc.
City	Syracuse
State or Province	NY - New York
ZIP or Postal Code	13202
Country	United States
Phone	212-555-1977

[✓] Default Shipping Address

This label is rendered using the following `fmt:message` statement:

```
<fmt:message key="common.nickNameThisAddress"/>
```

The `fmt:message` statement retrieves the "Nickname this address" text from the `WebAppResources` resource bundle, where the English translations are stored:

```
common.nicknameThisAddress=Nickname <span>this address</span>
```

Error Message Translations

Error message translations are handled in a similar way to displayable string translations. Each Java class that can produce error messages determines the customer's current locale and chooses the correct error message resource bundles accordingly.

Default Translations

CRS error messages for the default locale (English) are defined in a number of locations. For the English translations, CRS uses a layered resource bundle model, where some of the error messages come from the ATG platform and ATG Commerce, and then CRS layers on additional error messages in additional resource bundle files as necessary. For example, some gift list error messages are stored in the `/atg/commerce/gifts/UserMessages.properties` file in `<ATG10dir>/DCS/lib/resources.jar` while additional gift list messages are stored in the `/atg/commerce/gifts/UserMessages.properties` file in `<ATG10dir>/CommerceReferenceStore/Store/EStore/lib/classes.jar`. It is the aggregate of these two `/atg/commerce/gifts/UserMessage.properties` files that represents the total collection of gift list error messages.

CRS uses this resource bundle from the ATG platform. It is stored in `<ATG10dir>/DPS/lib/resources.jar`.

- `/atg/userprofiling/ProfileUserResources.properties`

CRS uses these resource bundles from ATG Commerce. They are stored in `<ATG10dir>/DCS/lib/resources.jar`.

- `/atg/commerce/catalog/comparison/UserMessages.properties`
- `/atg/commerce/gifts/UserMessages.properties`
- `/atg/commerce/order/UserMessages.properties`
- `/atg/commerce/order/purchase/PurchaseProcessResources.properties`
- `/atg/commerce/order/purchase/UserMessages.properties`
- `/atg/commerce/profile/UserMessages.properties`
- `/atg/commerce/promotion/PromotionResources.properties`
- `/atg/commerce/util/CountryStateResources.properties`
- `/atg/payment/creditcard/CreditCardResources.properties`

CRS layers these resource bundles on top of bundles that exist in ATG Commerce. They are stored in `<ATG10dir>/CommerceReferenceStore/Store/EStore/lib/classes.jar`:

- `/atg/commerce/catalog/comparison/UserMessages.properties`
- `/atg/commerce/gifts/UserMessages.properties`
- `/atg/commerce/order/UserMessages.properties`
- `/atg/commerce/order/purchase/PurchaseProcessResources.properties`
- `/atg/commerce/order/purchase/UserMessages.properties`

- `/atg/commerce/profile/UserMessages.properties`
- `/atg/commerce/promotion/PromotionResources.properties`

Finally, CRS adds these new resource bundles to the `<ATG10dir>/CommerceReferenceStore/Store/EStore/lib/classes.jar` file:

- `/atg/projects/store/catalog/EmailAFriendResources.properties`
- `/atg/projects/store/inventory/UserMessage.properties`
- `/atg/service/email/UserMessages.properties`

Creating Error Message Resource Bundles for Other Languages

To create localized error messages, you must create a set of companion resource bundles for each language. These files must:

- Reside in the same Nucleus path locations as their default locale counterparts.
- Follow the name extension rule of an underscore and a two-letter, lower-case language code as defined by ISO 639.
- Take into account any resource bundle layering. For example, `/atg/commerce/gifts/UserMessages_de.properties` contains German translations for the messages contained in both the `/atg/commerce/gifts/UserMessages.properties` file in `<ATG10dir>/DCS/lib/resources.jar` and the `/atg/commerce/gifts/UserMessages.properties` file in `<ATG10dir>/CommerceReferenceStore/Store/EStore/lib/classes.jar`.

CRS includes error message bundles for both German and Spanish. The German error messages are contained in the following files in `<ATG10dir>/CommerceReferenceStore/Store/EStore/lib/classes.jar`:

- `/atg/commerce/catalog/comparison/UserMessages_de.properties`
- `/atg/commerce/gifts/UserMessages_de.properties`
- `/atg/commerce/order/UserMessages_de.properties`
- `/atg/commerce/order/purchase/PurchaseProcessResources_de.properties`
- `/atg/commerce/order/purchase/UserMessages_de.properties`
- `/atg/commerce/profile/UserMessages_de.properties`
- `/atg/commerce/promotion/PromotionResources_de.properties`
- `/atg/commerce/util/CountryStateResources_de.properties`
- `/atg/payment/creditcard/CreditCardResources_de.properties`
- `/atg/projects/store/catalog/EmailAFriendResources_de.properties`
- `/atg/projects/store/inventory/UserMessage_de.properties`
- `/atg/service/email/UserMessages_de.properties`
- `/atg/userprofiling/ProfileUserResources_de.properties`

The Spanish error messages are contained in a similar set of files that use the _es extension in place of _de.

ATG recommends that you make copies of the error message resource bundles and edit them to include your localized text. Because they contain the aggregate of all English error messages, it is easiest to copy either the German or Spanish resource bundle files when creating resource bundles for a new language in CRS.

Repository Item Translations

The EStore.International module extends the repository definitions for existing item types that have properties that require translation (category, product, SKU, etc.). It also adds several new helper item types that are used to store translated content for the existing item types. These two mechanisms work together to replace properties that require translation with derived properties that obtain their values based on the customer's current locale. The following section describes this process in detail.

Extending the Repository Definitions

The International module makes the following modifications to each item type that has properties that require translation:

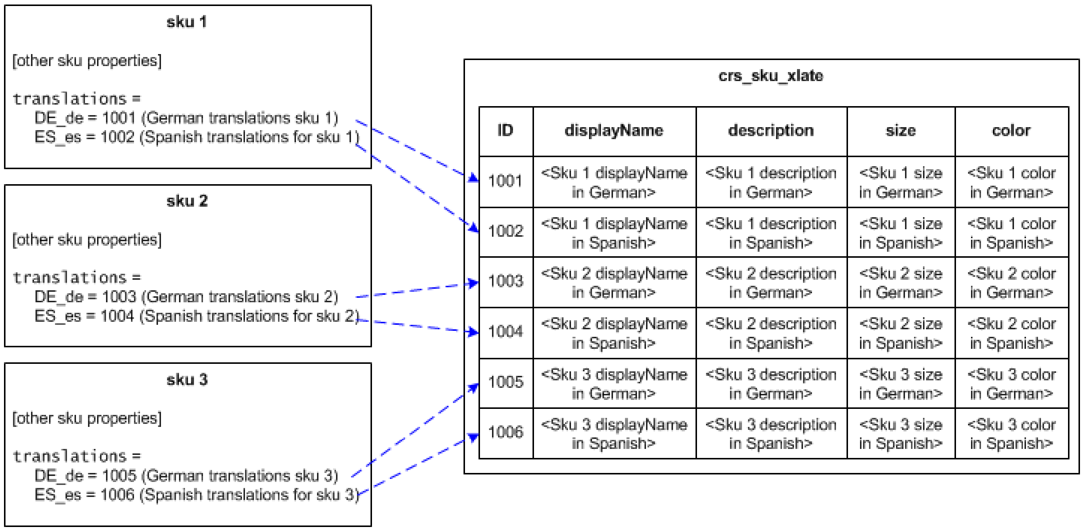
- New properties are created that correspond to the item type's translatable properties. For example, if an item type has three translatable properties, three new properties are created. The new properties are named using the convention <*translatableProperty*>Default. They are tied to the original properties' database columns and represent the default translations for the properties (thereby allowing us to redefine the original properties as derived properties). For example, the SKU item descriptor has four properties that require translation: displayName, description, size, and color. The International module adds four new properties — displayNameDefault, descriptionDefault, sizeDefault, and colorDefault — to the SKU item descriptor. These four properties are tied to the display_name, description, sku_size, and color columns, respectively, where the default translations for the content (English for CRS) are stored.
- A translations property is added to the item type. The translations property is a map whose key is a locale and whose value is an item of type <*baseType*>Translation, described below. Note that the locale key does not have to be a fully qualified locale. In fact, CRS only uses the language code portion of the locale. We'll cover locale in more detail below as well.

The International module defines a set of helper item types that use the naming convention <*baseType*>Translation, where <*baseType*> refers to an existing item type. A corresponding <*baseType*>Translation item type is defined for all existing item types that have translatable properties (for example, a skuTranslation item type is created to correspond with the sku item type, a productTranslation item type is created to correspond with the product item type, and so on). <*baseType*>Translation items function as containers for locale-specific content. As such, each <*baseType*>Translation item type has properties that correspond to the translatable properties of its base item type. For example, the sku item type has four properties that require translation — displayName, description, size, and color — therefore, the skuTranslation item also has four properties for displayname, description, size, and color.

Each <baseType>`Translation` item type has its own table in the database, where each row represents a single <basetype>`Translation` item with a unique ID. For example, the `crs_sku_xlate` table contains all the `skuTranslation` items, the `crs_prd_xlate` table contains all the `productTranslation` items, and so on.

Note: For a detailed list of <*baseType*>`Translation` item types created by the `International` module, see Translation Items Created by the International Module.

Every base item (in other words, every SKU, every product, every category, and so on) is tied, through its `translations` property, to one or more <*baseType*>`Translation` items. For each base item, a separate <*baseType*>`Translation` item exists for each locale (with the exception of the default locale, which we will get to momentarily). The following example shows three `sku` items and their six corresponding `skuTranslation` items which contain translated content for two locales, Spanish and German:


sku 1
[other sku properties]
translations =
DE_de = 1001 (German translations sku 1)
ES_es = 1002 (Spanish translations for sku 1)
sku 2
[other sku properties]
translations =
DE_de = 1003 (German translations sku 2)
ES_es = 1004 (Spanish translations for sku 2)
sku 3
[other sku properties]
translations =
DE_de = 1005 (German translations sku 3)
ES_es = 1006 (Spanish translations for sku 3)
crs_sku_xlate

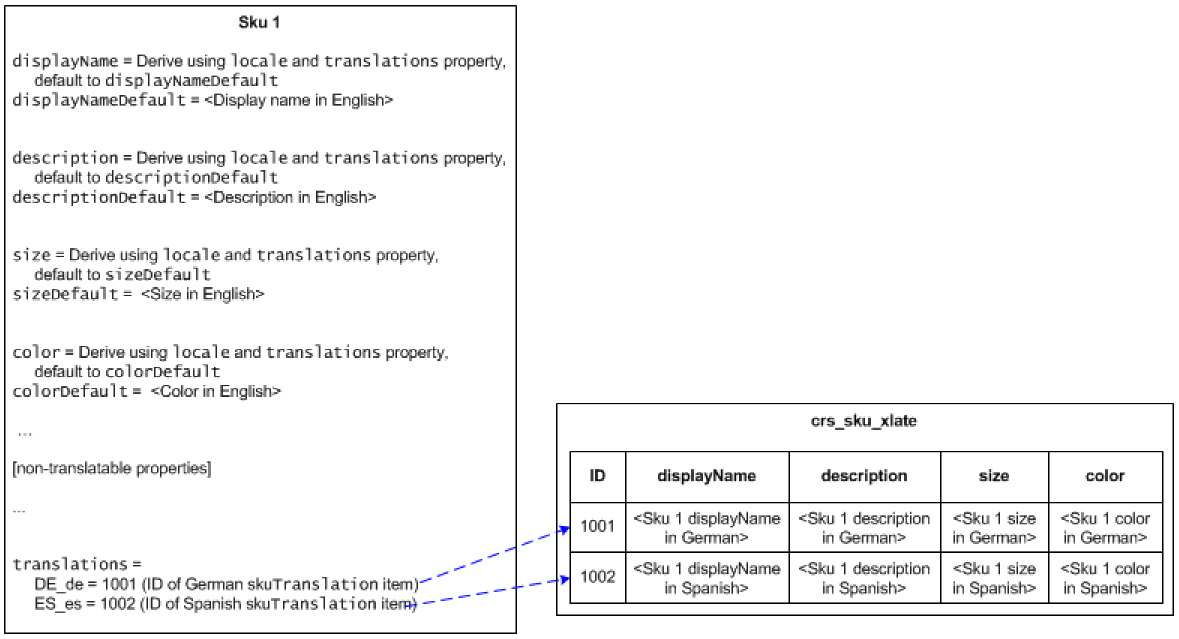
ID	displayName	description	size	color
1001	<Sku 1 displayName in German>	<Sku 1 description in German>	<Sku 1 size in German>	<Sku 1 color in German>
1002	<Sku 1 displayName in Spanish>	<Sku 1 description in Spanish>	<Sku 1 size in Spanish>	<Sku 1 color in Spanish>
1003	<Sku 2 displayName in German>	<Sku 2 description in German>	<Sku 2 size in German>	<Sku 2 color in German>
1004	<Sku 2 displayName in Spanish>	<Sku 2 description in Spanish>	<Sku 2 size in Spanish>	<Sku 2 color in Spanish>
1005	<Sku 3 displayName in German>	<Sku 3 description in German>	<Sku 3 size in German>	<Sku 3 color in German>
1006	<Sku 3 displayName in Spanish>	<Sku 3 description in Spanish>	<Sku 3 size in Spanish>	<Sku 3 color in Spanish>

To create the relationships that connect a base item to its <*baseType*>`Translation` items, the `International` module changes the definitions of the translatable properties in the existing item types. The new definitions specify that each translatable property is a derived property whose value is determined as follows:

- Use the current locale to look up a corresponding <*baseType*>`Translation` item in the `translations` property map. The property derivation attempts to find a best match. First, it searches the `locale` keys for a match on the entire locale with a variant, then it searches for a match on the locale without a variant, and finally it searches on just the language code.
- If a <*baseType*>`Translation` item exists for the current locale, use its value for the property.
- If a <*baseType*>`Translation` item doesn't exist for the current locale, or its value for the property is null, use the <*translatableProperty*>`Default` value instead.

The following illustration is a graphic representation of the repository changes made by the `International` module. It shows a `sku` base item with two supporting `skuTranslation` items, one for

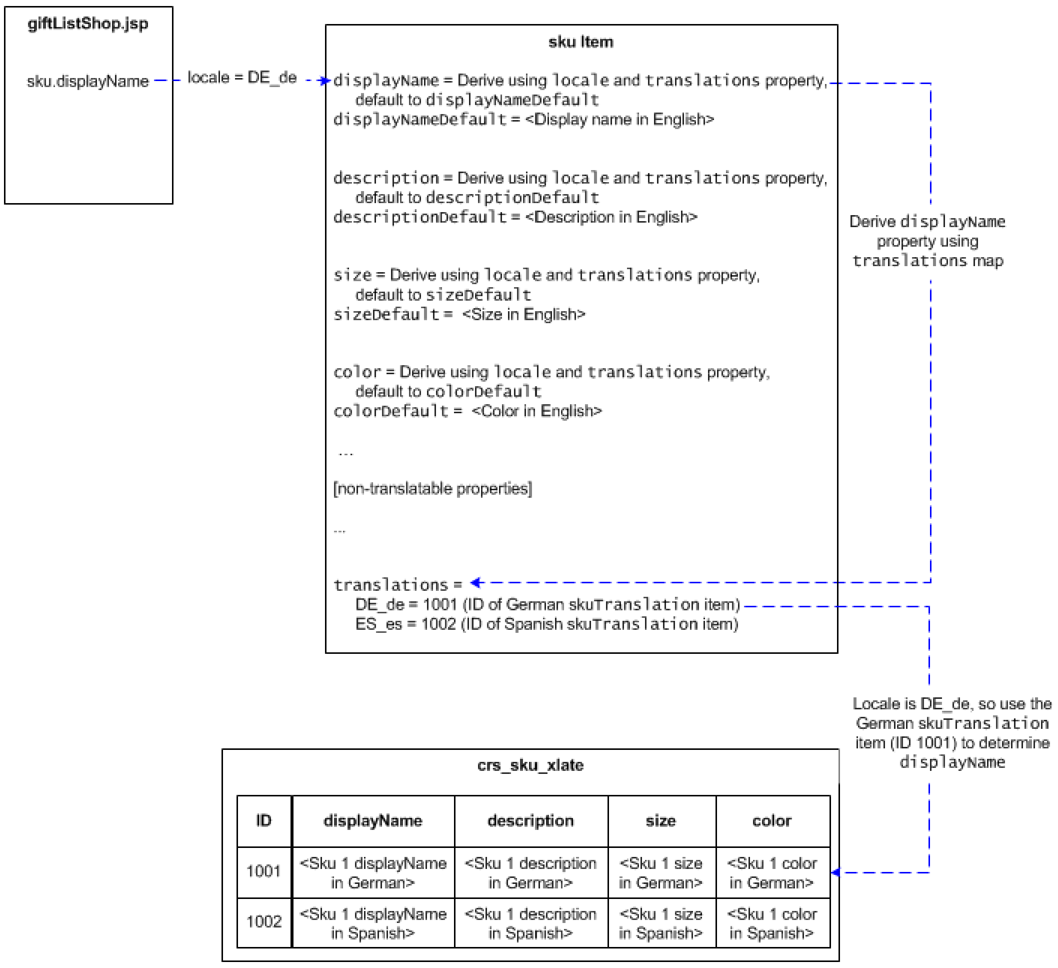
German and the other for Spanish. The `sku` base item properties have been modified to add the <*translatableProperty*>`Default` and `translations` properties, and to change the translatable properties to derived properties.


Sku 1
displayName = Derive using locale and translations property, default to displayNameDefault
displayNameDefault = <Display name in English>
description = Derive using locale and translations property, default to descriptionDefault
descriptionDefault = <Description in English>
size = Derive using locale and translations property, default to sizeDefault
sizeDefault = <Size in English>
color = Derive using locale and translations property, default to colorDefault
colorDefault = <Color in English>
...
[non-translatable properties]
...
translations =
DE_de = 1001 (ID of German skuTranslation item)
ES_es = 1002 (ID of Spanish skuTranslation item)
crs_sku_xlate

ID	displayName	description	size	color
1001	<Sku 1 displayName in German>	<Sku 1 description in German>	<Sku 1 size in German>	<Sku 1 color in German>
1002	<Sku 1 displayName in Spanish>	<Sku 1 description in Spanish>	<Sku 1 size in Spanish>	<Sku 1 color in Spanish>

Repository Item Translation Examples

The following illustration shows how the `sku.displayName` property is derived for a store that has English (default), German, and Spanish translations:


giftListShop.jsp
sku.displayName
locale = DE_de
sku Item
displayName = Derive using locale and translations property, default to displayNameDefault
displayNameDefault = <Display name in English>
description = Derive using locale and translations property, default to descriptionDefault
descriptionDefault = <Description in English>
size = Derive using locale and translations property, default to sizeDefault
sizeDefault = <Size in English>
color = Derive using locale and translations property, default to colorDefault
colorDefault = <Color in English>
...
[non-translatable properties]
...
translations =
DE_de = 1001 (ID of German skuTranslation item)
ES_es = 1002 (ID of Spanish skuTranslation item)
Derive displayName property using translations map
Locale is DE_de, so use the German skuTranslation item (ID 1001) to determine displayName
crs_sku_xlate
ID
displayName
description
size
color
1001
<Sku 1 displayName in German>
<Sku 1 description in German>
<Sku 1 size in German>
<Sku 1 color in German>
1002
<Sku 1 displayName in Spanish>
<Sku 1 description in Spanish>
<Sku 1 size in Spanish>
<Sku 1 color in Spanish>

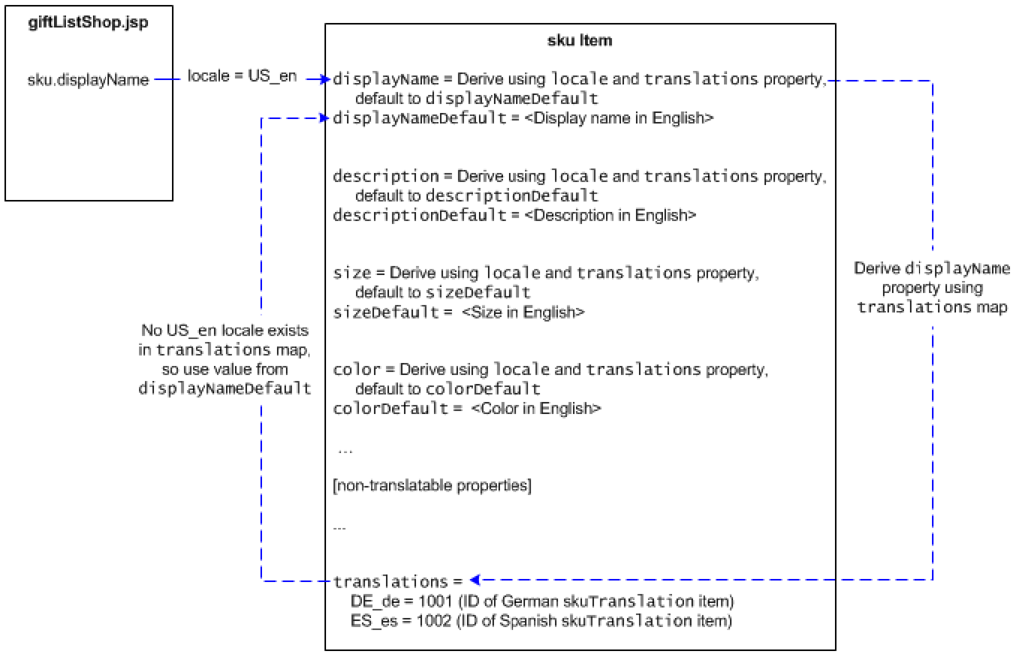
In this illustration, the following happens:

1. `giftListShop.jsp` requests the `sku.displayName` property for a SKU. The locale for the request is determined according to the rules described in Determining a Customer's Locale and, in this example, is `DE_de`.

2. The catalog repository finds the corresponding `skuTranslation` item using the `translations` property map. The repository determines that, for a `DE_de` locale, the German `skuTranslation` item should be referenced.

3. The catalog repository returns the `displayName` property from the German `skuTranslation` item.

This next illustration shows what happens when a locale is specified that doesn't exist in the `translations` property map.


giftListShop.jsp
sku.displayName
locale = US_en
sku Item
displayName = Derive using locale and translations property, default to displayNameDefault
displayNameDefault = <Display name in English>
description = Derive using locale and translations property, default to descriptionDefault
descriptionDefault = <Description in English>
size = Derive using locale and translations property, default to sizeDefault
sizeDefault = <Size in English>
color = Derive using locale and translations property, default to colorDefault
colorDefault = <Color in English>
...
[non-translatable properties]
...
translations =
DE_de = 1001 (ID of German skuTranslation item)
ES_es = 1002 (ID of Spanish skuTranslation item)
No US_en locale exists in translations map, so use value from displayNameDefault
Derive displayName property using translations map

In this illustration, the following happens:

1. `giftListShop.jsp` requests the `sku.displayName` property for a SKU. The locale for the request is determined according to the rules described in Determining a Customer's Locale and, in this example, is `US_en`.
2. The catalog repository derives the display name for the SKU using the `translations` property map. The repository determines that no `skuTranslation` item exists for the `US_en` locale, so it returns the value from `displayNameDefault`, which for CRS is the English translation.

Repository Item Translation Benefits

The approach CRS takes to handling the translation of repository items offers several significant benefits over other internationalization approaches:

- An application may switch between international and non-international modes without requiring any JSP page changes. The same property names are used in the JSP page code and each repository derives the appropriate language as necessary.
- Adding another language is accomplished by adding more <*baseType*>`Translation` items, referred to by their base items through the appropriate `locale` key. No database schema changes are required to add additional languages.

Translation Items Created by the International Module

Several repository definition files in the `International` module's `config.jar` file (<ATG10dir>`/CommerceReferenceStore/Store/EStore/International/config/config.jar`) define the <*baseType*>`Translation` items for CRS.

The /atg/commerce/catalog/custom/customCatalog.xml file defines the following <*baseType*>Translation items for the catalog repository:

- categoryTranslation (stored in the crs_cat_xlate table)
- productTranslation (stored in the crs_prd_xlate table)
- skuTranslation (stored in the crs_sku_xlate table)
- featureTranslation (stored in the crs_fea_xlate table)
- asSeenInTranslation (stored in the crs_asi_xlate table)
- promotionalContentTranslation (stored in the crs_prmcnt_xlate table)

The /atg/commerce/pricing/pricingModels.xml file defines the following <*baseType*>Translation items for the promotions repository:

- promotionTranslation (stored in the crs_prm_xlate table)
- closenessQualifierTranslation (stored crs_cq_xlate table)

The /atg/seo/SEORepository.xml file defines the following <*baseType*>Translation items for the SEO repository:

- SEOTranslation (stored in the crs_seo_xlate table)

The /atg/store/stores/storeText.xml file defines the following <*baseType*>Translation items for the store text items held in the StoreRepository:

- storeTextTranslation (stored in the crs_txt_xlate table)
- storeLongTextTranslation (stored in the crs_txt_long_xlate table)

A Note About the StoreRepository

The /atg/store/stores/StoreRepository holds store items as well as storeText and storeLongText items. store items define properties for store locations such as address, phone, and fax numbers. storeText and storeLongText items hold text-based content that appears on the static information pages such as About Us, Corporate Site, Careers, and so on (storeText holds content that is 256 characters or less; storeLongText holds content that is greater than 256 characters).

To view storeText and storeLongText items:

1. Start the ATG Business Control Center.
2. Expand Merchandising, then click Browse.
3. Click the arrow next to Store Text.

Site-Specific Cascading Stylesheets

In addition to the two main cascading stylesheets, store.war/css/site/store.css and store.war/css/site/home.css, CRS includes a number of locale-specific stylesheets:

- `store_de.css` -- Used by ATG Store Germany when German is the current language.
- `store_de_ie.css` -- Used by ATG Store Germany when German is the current language; includes additional settings for Internet Explorer.
- `store_es.css` -- Used by ATG Store US when Spanish is the current language.
- `store_es_ie.css` -- Used by ATG Store US when Spanish is the current language; includes additional settings for Internet Explorer.
- `home_es.css` -- Used by ATG Home when Spanish is the current language.
- `home_es_ie.css` -- Used by ATG Home when Spanish is the current language; includes additional settings for Internet Explorer.

These stylesheets are primarily for handling languages that have longer word lengths than English. If the related regional store you are adding uses a language that requires this type of customization, you should add new stylesheets as necessary. See Adding a Locale-specific CSS File in this chapter and Cascading Stylesheets (CSS) in the Pages and Navigation chapter for more details.

Displaying Site and Locale-specific Images

CRS includes a number of images that are site and/or locale-specific. This section describes how they are managed.

Images Associated with promotionalContent Items

A number of `promotionalContent` items are associated with images that are site and/or locale-specific. These `promotionalContent` items have two properties, `image` and `derivedImage`, that work together to derive an image URL that is appropriate for the current site context.

A `promotionalContent` item's `image` property contains a URL that uses `{site}` and `{language}` variables. For example, the Home Page Middle Banner `promotionalContent` item's `image` property has the following value:

```
/crsdocroot/images/atgstore/homepage_promo_{site}_{language}_anonymous.png
```

This URL provides the foundation for deriving a site and locale-specific URL for the image. A JSP reference to a `promotionalContent` item's `derivedImage` property calls the `atg.projects.store.dp.StoreContextDerivation` class and passes it the value of the item's `image` property. `StoreContextDerivation` derives the URL by replacing the `{site}` and `{language}` variables in the `image` property value with the current site's site ID and language. For example, the `image` property value above could result in a derived URL that is similar to the following:

```
/crsdocroot/images/atgstore/homepage_promo_storeSiteUS_en_anonymous.png
```

This URL renders the hero image on the ATG Store US home page when the site is being viewed in English by an anonymous customer.

If you look in the `storedocroot.war/images/atghome`, `/atgstore`, and `/email` directories, you will see a number of images that are managed using this variable replacement mechanism.

Images Associated with User Interface Elements

For each request, CRS determines which cascading stylesheets to use, as described in the Cascading Stylesheets (CSS) section of the Pages and Navigation chapter. CRS ships with two main cascading stylesheets:

- `store.war/css/site/home.css` for ATG Home
- `store.war/css/site/store.css` for ATG Store US and ATG Store Germany

These two stylesheets define styles for user interface items such as buttons, icons, borders, and so on. The `home.css` stylesheet references images in `storedocroot.war/images/atghome` while the `store.css` stylesheet references images in `storedocroot.war/images/atgstore`. These directories hold site-specific images for user interface elements for ATG Home and ATG Store, respectively. If your new related regional store requires a different look and feel, you may need to create additional stylesheets and images to support the new look, as well as update the CSS linking logic described in the Cascading Stylesheets (CSS) section mentioned above.

Pricing and the Locale Property

The method used to localize prices and currency depends on whether the prices are being viewed before or after an item has been added to an order. Both methods rely on the `locale` property that is manually defined for each price list in the ATG Business Control Center.

Localizing Prices for Items That Are Not Part of an Order

Pages that need to display prices for items that are not part of an order (for example, the product detail page) include the `store.war/global/gadgets/formattedPrice.jsp` gadget. The `formattedPrice.jsp` gadget implements the `/atg/commerce/pricing/CurrencyCodeDroplet`, which uses the price list's `locale` to determine which currency to use when displaying a price.

Localizing Prices within an Order

As described in the Pricing the Order section of the Order Handling chapter, during the checkout phase, an order goes through a series of pricing engines to calculate the order's price at various stages (item-level prices, order total price, order with shipping price, order with tax price). As part of its pricing operation, each engine retrieves the `locale` from the appropriate price list, translates the `locale` into a currency code, and stores the code in the `currencyCode` property of the resulting price info object. Each engine generates its own class of price info object:

Pricing Engine	**Price Info Class**
`ItemPricingEngine`	`atg.commerce.pricing.ItemPriceInfo`

`OrderPricingEngine`	`atg.commerce.pricing.OrderPriceInfo`
`ShippingPricingEngine`	`atg.commerce.pricing.ShippingPriceInfo`
`TaxPricingEngine`	`atg.commerce.pricing.TaxPriceInfo`

All of the price info classes are subclasses of the `atg.commerce.pricing.AmountInfo` class, where the `currencyCode` property is defined. Pages that display prices for items that are part of an order (for example, the order history page) retrieve the currency from the appropriate price info object when rendering the price information.

Appendix A: Implementing a Related Regional Store

This chapter provides detailed procedures you can follow to add a new related regional store to CRS. Specifically, it describes how to add an Italian version of ATG Store called ATG Store Italy. It includes the following topics:

Before You Begin

Your application server and your `ATGPublishing` and `ATGProduction` servers must be running before you begin the procedures in this chapter. If this is not the case, see *Starting the Application Server (WebLogic and WebSphere)* and *Starting the Publishing and Production Servers*, respectively, in the *ATG Commerce Reference Store Installation and Configuration Guide* for detailed instructions.

Adding Site Images

This section describes how to add site and/or locale-specific images.

Note: The images you add will not be visible until after you rebuild and redeploy the `ATGProduction.ear` file, which is the last step in this process and is described in Redeploying the ATGProduction.ear File.

Icons References by the Site Configuration

When you are configuring a new related regional store in Site Administration, there are several properties that reference image files: `siteIcon`, `largeSiteIcon`, and `favicon`. To demonstrate the use of locale-specific icon images, we will create an Italian version of `siteIcon` for ATG Store Italy.

To create a `siteIcon` image for ATG Store Italy:

1. Make a copy of the following file:

   ```
   <ATG10dir>/CommerceReferenceStore/Store/Storefront/j2ee-
   apps/Storefront/storedocroot.war/images/storefront/atgStore_logo_sm
   all.png
   ```

2. Rename the copied file to `atgStore_logo_small_it.png` and put it in the `storedocroot.war/images/storefront` directory.

3. Modify the image so that it is visually distinct from `atgStore_logo_small.png`, for example, change the text to read “ATG Store Italy.”

Adding an Image Associated with a promotionalContent Item

As described in Images Associated with promotionalContent Items, there are a number of images associated with `promotionalContent` items that are site and/or locale-specific. For the purposes of this document, we will add two example images, one that will function as the hero image for ATG Store Italy when the site is being viewed in Italian by an anonymous customer and another that will be the hero image for ATG Store Italy when the site is being viewed in English by an anonymous customer.

To create home page `promotionalContent` images for ATG Store Italy:

1. Create a copy of the following file:

   ```
   <ATG10dir>/CommerceReferenceStore/Store/Storefront/j2ee-
   apps/Storefront/storedocroot.war/images/atgstore/homepage_promo_sto
   reSiteUS_en_anonymous.png
   ```

2. Rename the copied file to `homepage_promo_storeSiteIT_it_anonymous.png` and put it in the `storedocroot.war/images/atgstore` directory.

3. Modify the image so that it is visually distinct from `homepage_promo_storeSiteUS_en_anonymous.png`, for example, add some text that says “ATG Store Italy, Italian language.”

4. Repeat these steps to create another copy of `homepage_promo_storeSiteUS_en_anonymous.png`. Rename the image to `homepage_promo_storeSiteIT_en_anonymous.png` and modify it so that it is visually distinct, for example, by adding text that says “ATG Store Italy, English language.”

Note: The instructions in this section describe how to create hero images when ATG Store Italy is being viewed by an anonymous user. In ATG Store, the Fashionista, Men Only, and Women Only user segments have their own hero images. If you want to see hero images when you are logged in as a Fashionista, Men Only, or Women Only user, you will need to create the associated hero images. In other words, you need to create `homepage_promo_storeSiteIT_en_fashionista.png`, homepage_promo_storeSiteIT_en_men.png, and homepage_promo_storeSiteIT_en_women.png images.

User Interface Elements

ATG Store US and ATG Store Germany use the same UI element images, as defined in store.war/css/site/store.css. In this example, ATG Store Italy will also use the same UI elements. If your new related regional store requires site-specific UI elements, see Images Associated with User Interface Elements for information on how to implement them.

Adding a Locale-specific CSS File

This section describes how to add a locale-specific CSS file and update the CRS JSP pages to use that CSS file.

Note: Your CSS and JSP changes will not appear until after you rebuild and redeploy the ATGProduction.ear file, which is the last step in this process and is described in Redeploying the ATGProduction.ear File.

The locale-specific CSS files included with CRS are primarily used to increase the width of various elements to accommodate longer word lengths. Since this type of change is not easily detectable in a UI, our example will instead add a locale-specific CSS file for ATG Store Italy that changes the color of link text to red.

To create a locale-specific CSS file for ATG Store Italy:

1. In the store.war/css/site directory, create a text file called store_it.css.
2. Add the following text to store_it.css and save the file:

```
a:link,
a:visited{
  text-decoration:none;
  color:#FF0000;
}
```

After creating the CSS file, you must add logic to the JSP pages so that the store_it.css file will be used for ATG Store Italy requests.

To update the JSP pages to use store_it.css:

1. In a text editor, edit the store.war/includes/pageStart.jsp file to include the following code. Add the new code to the end of the section that has the comment `<%-- Load language specific CSS if this is a language which requires additional styling --%>`.

```
<c:if test="${language eq 'it'}">
  <link rel="stylesheet" href="${contextPath}${siteCssFile}_it.css"
        type="text/css" media="screen" title="no title"
        charset="utf-8" />
</c:if>
```

2. Repeat the previous step with the `store.war/includes/popupStart.jsp` file.

Adding a Locale to the User Profile

Out of the box, the ATG platform includes five user profile locales that are defined in the profile repository:

- `unset` (code 0)
- `en_US` (code 1)
- `fr_FR` (code 2)
- `ja_JP` (code 3)
- `de_DE` (code 4)

In addition, the `International` module adds the following user profile locales:

- `fr_US` (code 100)
- `es_US` (code 101)
- `en_FR` (code 102)
- `es_FR` (code 103)
- `de_FR` (code 104)
- `de_US` (code 105)
- `en_DE` (code 106)

If your new related regional store requires a different user profile locale, you must add it to the profile repository definition.

To add a new user profile locale to the profile repository:

1. In `<ATG10dir>/home/localconfig`, create the following file:

   `/atg/userprofiling/userProfile.xml`

2. Edit this file to include any new locales. Locales are defined in the `dps_user` table, under the `locale` property. Specify a language and country code, as defined by ISO 639 and 3166 respectively, for each new locale. Separate the language and country codes using an underscore, for example `it_IT`. You must also supply a unique `code` for each locale. When you are finished, your `userProfile.xml` file will look similar to the following:

```
<gsa-template xml-combine="append">
  <item-descriptor name="user">
    <table name="dps_user">
      <property name="locale" >
```

```
        <option value="it_IT" code="107"/>
        <option value="en_IT" code="108"/>
      </property>
    </table>
  </item-descriptor>
</gsa-template>
```

Your changes will not take effect until you restart the `ATGProduction` and `ATGPublishing` servers, as discussed in the next section.

Adding Price Lists for the Related Regional Store

When creating a new price list for a related regional store, you must specify the price list's locale, which determines the currency the price list will use. Out of the box, ATG Commerce ships with three price list locales:

- `en_US` (code 0)
- `de_DE_EURO` (code 1)
- `ja_JP` (code 2)

If your new price list requires a different locale, you must add it to the price list repository definition before creating the price list.

To add a new price list locale to the price list repository:

1. In `<ATG10dir>/home/localconfig`, create the following file:

   `/atg/commerce/pricing/priceLists/priceLists.xml`

2. Edit this file to include any new locales. Locales are defined in the `dcs_price_list` table, under the `locale` property. Specify a language code, country code and currency code, as defined by ISO 639, 3166, and 4217 respectively, for each new locale. Separate the language, country, and currency codes using an underscore, for example `it_IT_EURO`. You must also supply a unique code for each locale. When you are finished, your `priceLists.xml` file will look similar to the following:

```
<gsa-template xml-combine="append">
  <item-descriptor name="priceList">
    <table name="dcs_price_list">
      <property name="locale" >
        <option value="it_IT_EURO" code="3"/>
      </property>
    </table>
  </item-descriptor>
</gsa-template>
```

3. Stop and restart your `ATGPublishing` and `ATGProduction` servers to access the changes to the user profile and price list repository definitions. See *Starting the*

Publishing and Production Servers in the *ATG Commerce Reference Store Installation and Configuration Guide* for detailed instructions.

At this point, you can add list price and sale price lists for your related regional store in the ATG Business Control Center. The procedure below provides an overview for adding a price list. For more specific details, see *Using and Defining Price Lists* in the *ATG Merchandising User Guide*.

To add price lists:

1. In a browser, navigate to:

   ```
   http://<hostname>:<HTTP_port>/atg/bcc
   ```

   For example, on JBoss:

   ```
   http://localhost:8180/atg/bcc
   ```

   On WebLogic:

   ```
   http://localhost:7005/atg/bcc
   ```

   On WebSphere:

   ```
   http://localhost:9081/atg/bcc
   ```

2. Enter `admin` for both the username and password and click Log In.
3. In the Operations menu, expand Merchandising, then click Manage Commerce Assets.
4. Enter a name and, optionally, a description for the project, for example, `ItalianPriceLists`.
5. Click Continue.
6. From the project types list at the top of the page, choose Price Lists.
7. From the Show menu, choose Price Lists and Folders.
8. Click the Create New button and choose Price List.
9. On the General tab, enter a name for the new price list, for example, Italian List Prices.
10. From the Locale menu, choose the correct locale. For our example, choose `it_IT_EURO`.
11. In order to have prices appear for all the items in our example, we will use List Prices as the base price list:
    - Click the Select button for the Base Price List.
    - Click the Browse tab, select List Prices, then click OK.
12. Click Select Parent, then click the Browse tab.
13. Click the radio button next to the Price Lists folder and click OK.
14. Click Create.

15. Click the Price Lists tab, specify prices for your catalog, then click Save. For detailed information on defining price lists, see *Using and Defining Price Lists* in the *ATG Merchandising User Guide*.
16. Repeat steps 8 through 15 for the sale price list. Use the name Italian Sale Prices for the price list name and Sale Prices for the base price list.
17. When you are finished adding price lists for your catalog, choose Ready for Review from the Task menu at the top of the window.
18. Optionally, enter a note for this task. Click OK.
19. For the Content Review task, choose the Approve Content action, then click Go.
20. Optionally, enter a note for this task. Click OK.
21. For the Approve for Production Deployment task, choose the Approve and Deploy to Production action, then click Go.
22. Optionally, enter a note for this task. Click OK.
23. Refresh the ATG Business Control Center window until you can see the Actions menu for the Verify Production Deployment task.
24. For the Verify Production Deployment task, choose the Accept Production Deployment action, then click Go.
25. Optionally, enter a note for this task. Click OK.

Creating a Related Regional Store Site

Follow the instructions below to add a new site for ATG Store Italy that will function as a related regional peer to ATG Store. After creating the ATG Store Italy site, you must also add it to the Related Regional Stores sharing group so that it functions as a regional peer to ATG Store US and ATG Store Germany.

Note: The new site will not be visible until it is deployed to the `ATGProduction` server through the ATG Business Control Center. Because deployments take time, we will wait to deploy until after we have added translations for repository items, as described in Adding Repository Item Translations.

To create a related regional store site:

1. In the ATG Business Control Center, click BCC Home at the top of the page.
2. Expand Site Administration, then click Manage Site Assets.
3. Enter a name and optional description for the project, for example `ItalianSiteAndTranslations`.
4. Click Continue.
5. Click the + action menu and choose Site.
6. Select the Default Category and click OK.
7. Edit the site properties. The site properties you set depend on your site’s requirements. For this example, we’ll assume we are adding an Italian regional peer to

ATG Store US. The properties we set mirror those set for ATG Store Germany, which is also a regional peer of ATG Store US. The table below provides the Italian store property values (properties that are omitted from this list have no value specified or are system generated):

Property	Value
ID	storeSiteIT
Site Name	ATG Store Italy
Site Description	ATG Store Italy site
Site Category	Default Category
Site Icon	`/crsdocroot/images/storefront/atgStore_logo_small_it.png`
Favicon	`/crsdocroot/images/storefront/site/storeUS/favicon.png` **Note:** The `favicon` is not site-specific, so we will use the same image that ATG Store US uses.
Site Priority	2
Site Types	Commerce
Site Base URL	`/crs/storeit`
Context Root	`/crs`
Enabled Status	Yes
Default Catalog	Master Catalog **Note:** The quickest way to locate the Master Catalog is to click Select, then search for “Master Catalog.”
Default List Price List	Italian List Prices **Note:** The quickest way to locate the list prices price list is to click Select, then search for “Italian Price Lists.”
Default Sale Price List	Italian Sale Prices **Note:** The quickest way to locate the sale price list is to click Select, then search for “Italian Sale Prices.”
Billable Country Codes	US, CA, MX, DE, IT
Shippable Country Codes	US, CA, MX, DE, IT

New Product Threshold Days	15
Resource Bundle	`atg.projects.store.web.WebAppResources`
Default Page Size	12
Path to CSS File	`/css/site/store`
Large Site Icon	`/crsdocroot/images/atgstore/logo.png`
Default Country	IT
Default Language	it
Languages	it, en
Back In Stock From Address, New Password From Address, Order Confirmation From Address, Order Shipped From Address, Promotion From Address	ATG Store Italy <ATGStoreItaly@example.com>
Email a Friend Enabled	Yes
Search Content	Content Label = Catalog Content Set = Catalog Content Set

8. Click Create.

   Note: Adding a new site requires a full re-index of the CRS content. You will re-index after adding ATG Store Italy to the Related Regional Stores sharing group.

To add the new related regional store to the Related Regional Stores sharing group:

1. Under Site Groups, double-click Related Regional Stores.
2. Click the Sites in Group property to enable its editor.
3. Click the Select button.
4. Double-click the ATG Store Italy site, then click OK.
5. Click Save.

To re-index the CRS content:

1. From the project types list at the top of the page, choose BCC Home.
2. Click Search Administration, then click Browse Search Projects.
3. On the Projects tab, expand Projects, then expand CRS Search Project.
4. Click Indexing.
5. Click Build Index to initiate a full index.

6. After the index is complete, click BCC Home at the top right to return to the BCC home page.

Modifying CRS Targeters

When creating any new site, you should consider whether the site can use default rules for personalization assets, or whether you need to specify rule overrides for the new site. Several of the CRS targeters use default rules that are not appropriate for ATG Store Italy. For example, the `CategoryPromotionContent1` targeter's default rules return `promotionalContent` items that encourage customers to make purchases on ATG Home. Since ATG Store Italy does not share a shopping cart with ATG Home, it does not make sense for ATG Store Italy to promote ATG Home. This means that ATG Store Italy should not use the default rules defined for `CategoryPromotionContent1`, instead it should use rule overrides. The instructions in this section tell you how to modify the rules for existing targeters so that they are appropriate for ATG Store Italy.

Notes:

- Your targeter changes will not be visible until they are deployed to the `ATGProduction` server through the ATG Business Control Center. Because deployments take time, we will wait to deploy until after we have added translations for repository items, as described in Adding Repository Item Translations.
- For more detailed information on the CRS targeters and other personalization assets, see the Personalization and Promotions chapter.

To modify CRS targeters for ATG Store Italy:

1. On the BCC Home page, expand Personalization, then click Targeting and Segmentation.
2. From the Show menu, choose Targeters.
3. Expand ProductCatalog.
4. Click `CategoryPromotionContent1` to see its properties in the right pane.
5. Click the Rules tab and locate Override Rule Sets 2.
6. Click the control next to "executes on" for Override Rule Sets 2.
7. Select ATG Store Italy and ATG Store Germany, then click OK. With this configuration, ATG Store Italy will use the same override rules as ATG Store Germany.
8. Click Save.
9. Click `HomePromotionalItem1` to see its properties.
10. Click the Rules tab and locate Override Rule Sets 1.
11. Click the control next to "executes on" for Override Rule Sets 1.
12. Select ATG Store, ATG Store Italy, and ATG Store Germany, then click OK. With this configuration, ATG Store Italy will use the same override rules as ATG Store and ATG Store Germany.

13. Click Save.
14. Repeat steps 8 to 12 for `HomePromotionalItem2` and `HomePromotionalItem3`.

Adding Translations

The instructions in this section describe how to add translations for repository items, strings, and error messages. Note that, after you complete the procedures below, your repository item changes will be immediately visible upon deploying them to the `ATGProduction` server, but your displayable string and error message translations will not be available until after you rebuild and redeploy the `ATGProduction.ear` file, which is the last step in this process and is described in Redeploying the ATGProduction.ear File.

Adding Repository Item Translations

To add translations for repository items, you must create a series of corresponding <*baseType*>`Translation` items, as described in Repository Item Translations. You add these items through the ATG Business Control Center.

To add translations for repository items:

1. From the project types list at the top of the page, choose Merchandising.
2. Follow the instructions below for the type of item you want to add translations to.

   To add translations for categories, products, and SKUs:

   - Click the arrow next to Site Catalogs.
   - Click the arrow next to the site whose catalog you want to add translations to. In this example, click the arrow next to ATG Store Italy.
   - Continue to navigate down into the site's catalog until you arrive at the item you want to add translations to, then double-click the item to see its properties in the pane on the right.
   - Click the Advanced tab.
   - Click the Translations table to activate the Translations options.
   - Click the plus sign to see the New [Item] Translation options.
   - Enter translations for the item properties that support translated data.

     Note: You may have to scroll down to see all the properties that support translations.
   - Click Create, then enter a two-letter language code as defined by ISO 639 in the Key column to identify the translations; for example, for Italian enter `it`.
   - Click Save.
   - Repeat these steps to add additional translations for other languages.

   To add translations for promotions:

- Return to the ATG Merchandising home page.
- Click the arrow next to Promotion & Coupons, then click the arrow next to Promotions.
- Double-click a promotion.
- On the General tab, click the Translations table to activate the Translations options.
- Click the plus sign to see the New Promotion Translation options.
- Enter translations for the item properties that support translated data.
- Click Create, then enter a two-letter language code as defined by ISO 639 in the Key column to identify the translations; for example, for Italian enter `it`.
- Click Save.
- Repeat these steps to add additional translations for other languages.

To add translations for store text:

- Return to the ATG Merchandising home page.
- Click the arrow next to Store Text.
- Double-click a store text item.
- Click the Translation table to activate the Translations options.
- Click the plus sign and choose New Resource Text Translation if your resource is 256 characters or less. Otherwise, choose New Resource Long Text Translation. The New Resource Text Translation options appear.
- Enter a translation for the store text item.
- Click Create, then enter a two-letter language code as defined by ISO 639 in the Key column to identify the translation; for example, for Italian enter `it`.
- Click Save.
- Repeat these steps to add additional translations for other languages.

To add translations for SEO tags:

- From the project types list at the top of the page, choose SEO tags.
- Select a tag.
- In the Presentation section, click Add New.
- In the Key field, enter a two-letter language code as defined by ISO 639; for example, for Italian enter `it`.
- Click Next.
- Enter a description, a display name, a comma-separated list of keywords, and a title in the language of your choice.
- Click Create & Close.

After you have added translations, you can deploy your project to the `ATGProduction` server, using the instructions below:

1. From the project types list at the top of the page, choose BCC Home.
2. In the To Do List area, choose Projects I Created from the Show menu.
3. Click Go to Project for the `ItalianSiteAndTranslations` project.
4. For the Author task, select the Ready for Review action and click Go.
5. Optionally, enter a note for this task. Click OK.
6. For the Content Review task, select the Approve Content action and click Go.
7. Optionally, enter a note for this task. Click OK.
8. For the Approve for Production Deployment task, select the Approve and Deploy to Production action and click Go.
9. Optionally, enter a note for this task. Click OK.
10. Refresh the ATG Business Control Center window until you can see the Actions menu for the Verify Production Deployment task.
11. For the Verify Production Deployment task, select the Accept Production Deployment action and click Go.
12. Optionally, enter a note for this task. Click OK.

Adding Displayable String Translations for the New Language

In this step, we create a copy of the default displayable strings file, `WebAppResources.properties`, and use it as the basis for the new resource bundle for Italian. The copied file will have all the resource property/value pairs the site needs, which makes it easier to edit the file for the new language.

Note: These changes will not appear until after you rebuild and redeploy the `ATGProduction.ear` file, which is the last step in this process and is described in Redeploying the ATGProduction.ear File.

To add translations for displayable strings:

1. Extract the `WebAppResources.properties` file from the `<ATG10dir>/CommerceReferenceStore/store/EStore/lib/classes.jar` file.
2. Create the following directory:

   `<AT1010dir>/home/locallib/atg/projects/store/web`
3. Place a copy of the `WebAppResources.properties` file in the `<AT1010dir>/home/locallib/atg/projects/store/web` directory you just created.
4. Rename the copied file to `WebAppResources_`<*code*>`.properties`, replacing <*code*> with a two-letter, lower-case language code as defined by ISO 639. For our example, we will rename the file to `WebAppResources_it.properties`.
5. Edit the `WebAppResources_`<*code*>`.properties` file so that the resource values contain your localized strings and save the file.

Updating Existing WebAppResources Files With the New Country

The Country picker in CRS references the `navigation_country` keys in the `WebAppResources_<`*code*`>.properties` files when displaying the list of countries you can pick from; a `navigation_country` key exists for each site/country combination. In order for a new country's name to appear in the Country picker you must add it to all of the `WebAppResources_<`*code*`>.properties` files.

Note: These changes will not appear until after you rebuild and redeploy the `ATGProduction.ear` file, which is the last step in this process and is described in Redeploying the ATGProduction.ear File.

To add a new country's name to the Country Picker:

1. Extract the following files from the `<ATG10dir>/CommerceReferenceStore/store/EStore/lib/classes.jar` file:
   - `WebAppResources.properties`
   - `WebAppResources_de.properties`
   - `WebAppResources_es.properties`
2. Place copies of the extracted files in the `<AT1010dir>/home/locallib/atg/projects/store/web` directory.

   At this point, you should have four `WebAppResources` files in this directory, one each for English, German, Spanish, and Italian.
3. In each `WebAppResources` file, search for the `navigation_country` key. You should see existing `navigation_country` keys for United States and Germany.
4. Add a `navigation_country.storeSiteIT` key for Italy. The value for this key will depend on the language of the resource bundle, for example:
   - For `WebAppResources.properties`, add `navigation_country.storeSiteIT=Italy`
   - For `WebAppResources_de.properties`, add `navigation_country.storeSiteIT=Italien`
   - For `WebAppResources_es.properties`, add `navigation_country.storeSiteIT=Italia`
   - For `WebAppResources_it.properties`, add `navigation_country.storeSiteIT=L'Italia`
5. Save the resource bundles.

Adding Error Message Translations

Error message translations are contained in a number of files. ATG recommends creating a copy of the default error message files, making it easy to edit them for your chosen language. See Error Message Translations for details on where you can find the default error message files.

Note: These changes will not appear until after you rebuild and redeploy the `ATGProduction.ear` file, which is the last step in this process and is described in Redeploying the ATGProduction.ear File.

To add translations for error messages:

1. In the `<ATG10dir>/home/locallib` directory, create the following error message property files, replacing <*code*> in the filenames with a two-letter, lower-case language code as defined by ISO 639:

   - `/atg/commerce/catalog/comparison/UserMessages_<`*code*`>.properties`
   - `/atg/commerce/gifts/UserMessages_<`*code*`>.properties`
   - `/atg/commerce/order/UserMessages_<`*code*`>.properties`
   - `/atg/commerce/order/purchase/PurchaseProcessResources_<`*code*`>.properties`
   - `/atg/commerce/order/purchase/UserMessages_<`*code*`>.properties`
   - `/atg/commerce/profile/UserMessages_<`*code*`>.properties`
   - `/atg/commerce/promotion/PromotionResources_<`*code*`>.properties`
   - `/atg/commerce/util/CountryStateResources_<`*code*`>.properties`
   - `/atg/payment/creditcard/CreditCardResources_<`*code*`>.properties`
   - `/atg/projects/store/catalog/EmailAFriendResources_<`*code*`>.properties`
   - `/atg/projects/store/inventory/UserMessage_<`*code*`>.properties`
   - `/atg/service/email/UserMessages_<`*code*`>.properties`
   - `/atg/userprofiling/ProfileUserResources_<`*code*`>.properties`

2. Edit the property files so that the resource values contain your localized error messages and save the files.

Redeploying the ATGProduction.ear File

Rebuilding and redeploying the `ATGProduction.ear` file adds the displayable string and error message resource bundles to the EAR file. You must remove the existing EAR file before you can rebuild and redeploy a new one. In the section below, we use CIM to rebuild and redeploy the EAR file.

Note: If you want to preview CRS in the ATG Business Control Center using different languages, you will also have to delete and then rebuild and redeploy the `ATGPublishing.ear` file.

Follow the instructions below for your application server.

Redeploying on JBoss

To remove the `ATGProduction.ear` file on JBoss:

1. Stop the `ATGProduction` server.

2. Delete the `ATGProduction.ear` file in the JBoss application server, for example, `/jboss-eap-5.0/jboss-as/server/ATGProduction/deploy/ATGProduction.ear`.

To rebuild and redeploy the `ATGProduction.ear` file:

1. Start CIM:
   - In a UNIX shell or Windows command prompt, change directories to `<ATG10dir>/home/bin`.
   - Enter one of the following commands.

     UNIX:

     ./cim.sh

     Windows:

     cim.bat
2. Select [3] Application Assembly & Deployment – Done.
3. Select [A] ATGProduction – Production with a Server Lock Manager – Done.
4. In the ENTER EAR FILE NAME(S) menu, press Enter to accept `ATGProduction.ear` as the EAR filename.
5. In the SELECT JBOSS SERVER TO USE menu, select [A] ATGProduction.
6. In the JBOSS DEPLOYMENT MENU, choose [D] Deploy Production with a Server Lock Manager ATGProduction.ear to JBoss – Done.
7. In the JBOSS DEPLOYMENT MENU, choose [O] Configure Another Server Instance.
8. In the DEPLOYMENT SERVER INSTANCE SELECTION menu, choose [D] Done.
9. In the CIM MAIN MENU, enter q to exit CIM.

To restart the production server:

1. In a UNIX shell, navigate to the JBoss `bin` directory, for example, `/usr/local/jboss-eap-5.0/jboss-as/bin`.
2. Enter the following command:

```
./run.sh -c ATGProduction -b 0.0.0.0
```

Redeploying on WebLogic

To remove the `ATGProduction.ear` file on WebLogic:

1. Go to the shell where you started the WebLogic server that supports the `ATGProduction` server instance and stop the WebLogic server.
2. Open a browser and enter the address of the WebLogic Server Administration Console:

```
http://localhost:7001/console
```

3. Enter your WebLogic username and password.
4. Under Domain Structure, click Deployments for your user domain.
5. Select the `ATGProduction.ear`, then click Delete.

To rebuild and redeploy the `ATGProduction.ear` file:

1. Start CIM:
   - In a UNIX shell, change directories to `<ATG10dir>/home/bin`.
   - Enter the following command:

     ```
     ./cim.sh
     ```
2. In the CIM MAIN MENU, select [3] Application Assembly & Deployment.
3. In the DEPLOYMENT SERVER INSTANCE SELECTION menu, select [A] ATGProduction - ATG Commerce Store Production with Server Lock Manager.
4. In the ENTER EAR FILE NAME(S) menu, press Enter to accept `ATGProduction.ear` as the EAR filename.
5. In the WEBLOGIC ONLINE DEPLOYMENT menu, select [D] Deploy ATG Commerce Store Production with Server Lock Manager ATGProduction.ear to Weblogic Online.
6. In the WEBLOGIC POST DEPLOYMENT OPTIONS menu, select [O] Configure Another Server Instance.
7. In the DEPLOYMENT SERVER INSTANCE SELECTION menu, select [D] Done.
8. In the CIM MAIN MENU, enter q to exit CIM.

To restart the production server:

1. In a UNIX shell, navigate to your user domain's `bin` directory, for example, `/usr/local/Oracle/Middleware/user_projects/domains/base_domain/bin`.
2. Enter the following command:

   ```
   ./startManagedWebLogic.sh ATGProduction
   ```
3. Enter your WebLogic username and password.

   When you see the message "Server started in RUNNING mode" continue with the next step.
4. Return to the WebLogic Server Administration Console.
5. If necessary, under Domain Structure, click Deployments for your user domain.
6. Select the `ATGProduction.ear`.
7. Click Start and choose Servicing all requests.

Redeploying on WebSphere

To remove the `ATGProduction.ear` file on WebSphere:

1. Open a browser and enter the address of the WebSphere Integrated Solutions Console, for example:

   ```
   http://localhost:9060/ibm/console
   ```

2. Enter your WebSphere ID.
3. Expand Servers, then click Application Servers.
4. Select the `ATGProduction` server and click Stop.
5. Click OK to confirm.
6. After the server has stopped, click OK to close the Server Status Feedback page.
7. Expand Applications, then click Enterprise Applications. You see the Enterprise Applications page.
8. Select the `ATGProduction.ear`, then click Uninstall.
9. Click OK to confirm the uninstall.
10. Click Save to save the change to the master configuration.
11. Log out of the WebSphere Integrated Solutions Console.

To rebuild and redeploy the `ATGProduction.ear` file:

1. Start CIM:
   - In a UNIX shell, change directories to `<ATG10dir>/home/bin`.
   - Enter the following command:

     ```
     ./cim.sh
     ```

2. In the CIM MAIN MENU, select [3] Application Assembly & Deployment.
3. In the DEPLOYMENT SERVER INSTANCE SELECTION menu, select [A] ATGProduction - ATG Commerce Store Production with Server Lock Manager.
4. In the ENTER EAR FILE NAME(S) menu, press Enter to accept `ATGProduction.ear` as the EAR filename.
5. Select [2] <*host*>`Node01` for the node.
6. In the SELECT WEBSPHERE SERVER TO USE menu, select [A] ATGProduction.
7. In the WEBSPHERE DEPLOYMENT MENU, select [D] Deploy ATG Commerce Store Production with Server Lock Manager ATGProduction.ear to WebSphere.
8. In the WEBSPHERE DEPLOYMENT MENU, select [O] Configure Another Server Instance.
9. In the DEPLOYMENT SERVER INSTANCE SELECTION menu, select [D] Done.
10. In the CIM MAIN MENU, enter q to exit CIM.
11. Log back into the WebSphere Integrated Solutions Console, expand Applications, then click Enterprise Applications. The new EAR file you just created should be listed.

To restart the production server and application:

1. In the WebSphere Integrated Solutions Console, expand Servers, then expand Server Types.
2. Click WebSphere application servers.
3. Select the `ATGProduction` server and click Start. This process starts the `ATGProduction` server and also automatically starts the `ATGProduction.ear` application.

Verifying Your Changes

Once your ATG Store Italy site is running, you can verify that following requirements have been met:

- Italy appears in the country picker.
- Italian appears in the language picker.
- The correct hero images appear on the ATG Store Italy home page for anonymous users viewing the site in English and Italian.

  Note: The procedures in this document create hero images for anonymous users only. If you log in as a user that belongs to the Fashionista, Men Only, or Women Only user segments, you will not see a hero image.
- Prices appear in the correct currency (euro).
- Hyperlinks when viewing ATG Store Italy in Italian are red, indicating the correct CSS file is being used.
- Translations appear for any repository items you edited. This includes categories, products, SKUs, promotions, and store text items.
- Translations appear for strings in resource bundles, specifically, displayable strings and error messages.
- To verify the user profile locales you added are accessible and being used, follow the procedures below. Note that the ATG Business Control Center displays the default locale for a given user and this value does not change unless you change it in the BCC. By contrast, the ATG Control Center shows the current locale for the user. This value is updated as the user navigates through the CRS application.

  To see the default locales that are available for users:

  - Log into the ATG Business Control Center.
  - Under the Operations area, expand Personalization, then click External Users.
  - Click a user to see the user profile properties in the right pane.
  - Click the Locale menu and verify that `en_IT` and `it_IT` appear.

  To see the current locale for a user:

  - In a browser window, go to the CRS Web site and log in as a sample user (for example, log in using adrian@example.com as the username and `password` as the password).

- Click Italy in the Country picker. Optionally, click Italian in the Language picker.
- Start the ATG Control Center (for details on how to do this, see the *ATG Installation and Configuration Guide*).
- Enter `admin` as the password and enter your `ATGProduction` server's hostname and the RMI port you used when configuring CRS (default is 8860), then click Connect.
- Click People and Organizations, then click Users.
- Click List to see all the sample users.
- Select the user you used when logging into the CRS Web site. The user's locale should be `en_IT` if the user was viewing the site in English or `it_IT` if the user was viewing the site in Italian.

  Note: You must refresh the ACC window to see changes in the user's properties as the user navigates around the CRS Web site.

Appendix B: Sample Data

This chapter describes the user profile, catalog, and store location data included with ATG Commerce Reference Store. It includes the following topics:

Sample User Data

This section describes the sample data created for CRS user profiles. It includes:

- A list of the sample profiles.
- Information on which customers have multiple addresses, wish lists, and gift lists.
- A breakdown of customers by user segment.
- A breakdown of customers by promotions granted.
- A breakdown of customer orders by order state.

Sample Customers

CRS ships with 22 user profiles for the following customers:

Adrian Robinson
Alex Jackson
Amy Scott
Andrew Wright
Anna Mitchell
Anna Parker
Brandon Johnson
Chuck Taylor
Elisabeth Robinson
Eric Adams
Erica Clark
Jacob Smith
Juan Rodriguez
Kim Anderson
Lindsay Moore

Lisa Moore
Maria Lopez
Peter Nelson
Rachel Thomas
Stuart Schmidt
Victoria Lewis
Vincent Lee

All users, except Anna Parker, have a login that follows this pattern:

first_name@example.com

For example, Stuart Schmidt's login is stuart@example.com. Anna Parker's login is annaparker@example.com, to distinguish her from Anna Mitchell. The password for all customers is `password`.

The repository for user profiles is `/atg/userprofiling/ProfileAdapterRepository`.

Customers with Multiple Addresses

The following customers have billing addresses that differ from their shipping addresses:

Amy Scott
Brandon Johnson
Erica Clark
Kim Anderson
Lindsay Moore
Peter Nelson
Stuart Schmidt

Two customers, Stuart Schmidt and Kim Anderson, also have secondary addresses in addition to their shipping and billing addresses. For example, Kim Anderson's profile includes a secondary address for shipping items to her mother's house.

Customers with Gift and Wish Lists

Stuart Schmidt and Kim Anderson each have a wish list and a gift list. Stuart's wish list is confined to products from ATG Store Germany. Kim's wish list has products from both ATG Store US and ATG Home. The gift lists for both customers include products from ATG Store US and ATG Home.

Sample User Segments

This table lists the customers that belong to each user segment:

User Segment	Customers
Fashionista	Kim Anderson

User Segment	Customers
Men Only	Adrian Robinson Alex Jackson Andrew Wright Brandon Johnson Chuck Taylor Eric Adams Jacob Smith Juan Rodriguez Peter Nelson Stuart Schmidt Vincent Lee
Women Only	Amy Scott Anna Mitchell Anna Parker Elisabeth Robinson Erica Clark Kim Anderson Lindsay Moore Lisa Moore Maria Lopez Rachel Thomas Victoria Lewis

In addition to the segments listed above, CRS implements two user segments, `ThirtySomethings` and `Young`, that are based on a customer's `age` profile property. `age` is a derived property that changes over time as a customer gets older. Because these segments are age-based, the customers included in them will also change over time. The `ThirtySomethings` segment includes customers whose age is greater than or equal to 30 and less than 40. The `Young` segment defines young customers differently on different sites:

- On ATG Store US, customers whose `age` is less than 35 are in the `Young` user segment.
- On ATG Home, customers whose `age` is less than 40 are in the `Young` user segment.
- ATG Store Germany does not use the `Young` user segment.

To see a customer's age and user segments, do the following:

1. Start the ATG Business Control Center.
2. Expand Personalization, then click External Users.
3. Make sure Users is selected in the Show menu.
4. Click a customer name in the list to see the customer's details. The customer's age is listed on the General tab. The segments the customer belongs to are listed on the Segments tab.

Sample Promotions

This table lists the customer orders to which promotions have been applied:

Promotion	Customer (Order number)
$10 Off Orders Over $100	Adrian Robinson (xco30045) Andrew Wright (xco30072) Brandon Johnson (xco30021, xco30086, xco50013) Kim Anderson (xco30081) Stuart Schmidt (xco50017, xco30106, xco30098, xco30076, xco30012) Vincent Lee (xco30041)
15% Off Home Accents	Stuart Schmidt (xco30106, xco30098, xco30076)
25% Off Women's Apparel	Kim Anderson (xco30015, xco30081) Maria Lopez (xco30031)
ATG Home – Free Shipping	Brandon Johnson (xco30021, xco30086) Lindsay Moore (xco30049) Stuart Schmidt (xco30106, xco30076) Vincent Lee (xco30041)
ATG Home Save 20% On One Item	Adrian Robinson (xco30045) Andrew Wright (xco30072) Brandon Johnson (xco30021, xco30086) Lindsay Moore (xco30049) Stuart Schmidt (xco30106, xco30098, xco30076) Vincent Lee (xco30041)
Hubbard Chair Tiered Price Break	Stuart Schmidt (xco30076)

Note: CRS sample data does not include orders that have had the following promotions applied to them:

- 10% Off Order
- TENSHIP Coupon – 10% Off Order
- TENSHIP Coupon – Free Shipping
- 10LIEFERUNG Coupon - 10% Off Order

- 10LIEFERUNG Coupon – Free Shipping

Order States

This table lists customers that have orders in various order states:

Order State	Customers
Order processing	Lindsay Moore Stuart Schmidt
Order processing with backordered item	Brandon Johnson Elisabeth Robinson
Order processing with preordered item	Brandon Johnson Stuart Schmidt
Order shipped	Adrian Robinson Andrew Wright Brandon Johnson Kim Anderson Lindsay Moore Maria Lopez Stuart Schmidt Vincent Lee

The repository for orders is `/atg/commerce/order/OrderRepository`.

Sample Catalog Data

This section describes the sample data created for the CRS product catalogs. It includes:

- The catalog hierarchy for `masterCatalog`, which is shared by ATG Store US and ATG Store Germany.
- The catalog hierarchy for `homeStoreCatalog`, which is used by ATG Home.
- A list of products representing various CRS features.

The repository for catalog data is `/atg/commerce/catalog/ProductCatalog`.

masterCatalog Hierarchy

```
Gift Shop
     For Him
     For Her
```

```
Women
     Shirts
     Pants
     Dresses
     Skirts
     Shorts
     Jackets
     Accessories
Men
     Shirts
     Pants
     Shorts
     Jackets
     Accessories
Shoes
     Women's Shoes
     Men's Shoes
Home Accents
     Clocks
     Radios
     Cushions and Pillows
     Glassware
     Lamps and Lighting
          Table Lamps
          Floor Lamps
          Desk Lamps
          Mounted Lamps
          Candles & Oil Lamps
     Decor
```

homeStoreCatalog Hierarchy

```
Gift Shop
     Glassware
     Unique & Different
Tables
     Dining Tables
     Side Tables
Seating
     Dining Chairs
     Office Chairs
     Sofas
     Armchairs and Recliners
     Ottomans and Footstools
     Outdoor Seating
     Kitchen Chairs and Stools
Storage & Display
     Cabinets
     Storage Containers
     Dressers and Drawers
Home Accents
     Clocks
     Radios
     Cushions and Pillows
     Glassware
     Lamps and Lighting
          Table Lamps
          Floor Lamps
          Desk Lamps
          Mounted Lamps
          Candles & Oil Lamps
     Decor
```

Representative Products

This section lists representative products that demonstrate various CRS features like preordering, backordering, pages with and without a color/size picker, etc.

Use case	Representative product(s)
Product with a single SKU	ATG Home > Gift Shop > Glassware > Tumbler Glass
Product with multiple SKUs, with color/size picker	ATG Store > Women > Shirts > Polka Dot Blouse
Product with multiple SKUs, no color/size picker	ATG Home > Seating > Sofas > Contemporary Italian Sofa and Chair Set ATG Store > Gift Shop > For Him > Analog Watch
Product with wood finish picker	ATG Home > Tables > Dining Tables > Block Table
Product with SKUs that have different prices	ATG Store > Women > Jackets > Suede Blazer
Preorderable SKU, checking inventory	ATG Store > Women > Shirts > Long Sleeve Soft Touch Hoodie (color=red, size=small)
Preorderable SKU, not checking inventory	ATG Home > Seating > Kitchen Chairs and Stools > uPod Stool ATG Store > Gift Shop > For Him > Laptop Briefcase

Use case	Representative product(s)
Backorderable SKU	ATG Home > Home Accents > Lamps and Lighting > Table Lamps > Cottage Lamp ATG Store > Gift Shop > For Him > Leather Everyday Belt (size=30, color=brown) ATG Store > Women > Shirts > Long Sleeve Soft Touch Hoodie (color=blue, size=medium)
Out of stock SKU	ATG Store > Gift Shop > For Him > Organized Wallet
Product with As Seen In data	ATG Home > Seating > Armchairs and Recliners > Italian Leather Chair and Ottoman ATG Home > Tables > Dining Tables > Block Table ATG Store > Shoes > Men’s Shoes > Leather Slip-ons ATG Store > Women > Pants > Straight Leg Jeans

Store Locations

CRS ships with a list of sample store locations, rendered by the `store.war/company/store.jsp` gadget:

Site	Store Locations
ATG Store US	Biltmore Fashion Park 2404 East Camelback Road #100E Phoenix, AZ, 85016 USA Phone: (602) 955-1963 Biltmore@westcor.com South Coast Plaza 3333 Bristol Street Costa Mesa, CA, 92626 USA Phone: 800-782-8888 Fashion Island 401 Newport Center Drive Newport, CA, 92660 USA Phone: 949-721-2000 service@shopfashionisland.com

Site	Store Locations
ATG Store Germany	Südküste-Piazza Alfred-Brehm-Platz 16 Frankfurt, 60316 Deutschland Telefon: 089 592 80 12 Piazza@example.de Art- und Weisepark Reisingerstrasse 12 Muenster, 48147 Deutschland Telefon: 089 343 80 14 Weisepark@example.de Bekleidungsgeschäft Museumsinsel 1 Muenster, 80538 Deutschland Telefon: 431 914 43 31 Beklei@example.de
ATG Home	Aventura Mall 19501 Biscayne Blvd Aventura, FL, 33180 USA Phone: (305) 935-1110 Fax: (305) 935-9360 Queens Center 90-15 Queens Boulevard Elmhurst, NY, 11373 USA Phone: (718) 592-3901 Fax: (718) 592-4157 Westlake Center 400 Pine Street Seattle, WA, 98101 USA Phone: (206) 467-1600 westlakeadmin@generalgrowth.com

Store locations are stored as entries in the `/atg/store/stores/StoreRepository` repository. To view store locations in ATG Merchandising:

1. Start the ATG Business Control Center.
2. Expand Merchandising, then click Browse next to Manage Commerce Assets.
3. Click the arrow next to Store Locations.
4. To view details for an individual location, double-click its name.

The `stores.jsp` page displays store locations using the `/atg/store/droplet/StoreLookupDroplet` servlet bean to retrieve the locations and the `/atg/store/droplet/StoreSiteFilterDroplet` to filter out stores that aren't appropriate for the current site context. Maintaining store location information in a repository provides a starting point for building more sophisticated store-locator logic (e.g., finding the store closest to a customer's location).

Index