ORACLE®

ATG WEB COMMERCE

Commerce Reference Store

Version 10.1.2

Overview

Oracle ATG
One Main Street
Cambridge, MA 02142
USA

ATG Commerce Reference Store Overview

Product version: 10.1.2
Release date: 12-17-12
Document identifier: CRSOverview1404301402

Table of Contents

1 Introduction

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

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

Getting Started

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

- A publishing server for managing and deploying site assets
- A production server that runs the site storefronts
- Catalog search facilities using Oracle Endeca Commerce
- 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 Oracle ATG Web Commerce Service Center

Commerce Reference Store is intended to illustrate best practices for creating stores with Commerce. You can use the application as a template for building your own stores. The Commerce Reference Store installation includes all of the Commerce Reference Store-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 Commerce Reference Store application modules. For example, class files for Commerce Reference Store-specific extensions to the Commerce Fulfillment system are found in:

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

More general functionality is found in:

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

The Commerce Reference Store 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/productDetailSingleSku.jsp` refers to:

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

Note: Although Commerce Reference Store simulates a production environment, the code is not truly production-ready. Before you go live with any web application built with Commerce Reference Store, 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* (page 21) chapter.

In addition, you may want to change the way your application serves images and other static content. The use of a WAR file for this purpose makes it easier to set up Commerce Reference Store for demo purposes on a variety of different application servers, but increases the amount of load on the application server. For production purposes, you may want to use an alternative approach to static content delivery, such as a dedicated file caching server or a content delivery network.

Multisite Support

Commerce Reference Store is an example of a multisite application, where multiple sites are run from the same Commerce instance and share certain data and other resources. Commerce Reference Store includes three sites: ATG Store US, ATG Store Germany, and ATG Home. See the *Multisite Features* (page 9) chapter for information about these sites.

Configuring Commerce Reference Store

This section describes mechanisms for specifying global configuration options for your installation of Commerce Reference Store:

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

In addition to these global settings, Commerce Reference Store adds site configuration properties to the `siteConfiguration` item type in the Site Repository. See the *Multisite Features* (page 9) chapter for information about these site-specific settings.

Note that to enable Commerce Reference Store 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 Platform 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.
`epochDate`	The sample date that appears on the registration page below the Date of Birth field to illustrate the format to be used when entering the date. The value of this property should be in the format specified by the `epochDatePattern` property; it is set to `01/31/1986` by default.
`epochDatePattern`	The format for entering a customer's date of birth on the registration page. By default, this property is set to `mm/dd/yyyy`.
`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.
`showUnindexedCategories`	Boolean indicating whether new categories that have not been indexed yet should appear on the sites. This property is set to `false` by default, but is set to `true` by the `preview` configuration layer on the Content Administration server if preview is enabled.

Assembling Your EAR Files

Commerce Reference Store includes all of the code, data, and configuration necessary for setting up a complete multi-server environment. When you set up Commerce Reference Store using the Configuration and Installation Manager (CIM), it assembles separate EAR files for your publishing server and your production server. 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 Commerce Reference Store 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 Commerce Fulfillment with Commerce Reference Store extensions.	Production (or separate instance)
`Store.DCS-CSR`	Adds configuration to Commerce Service Center for working with Commerce Reference Store.	Agent-facing
`Store.Fluoroscope`	Enables Fluoroscope, a tool for viewing site HTML pages that reveals key JSP elements involved in rendering those pages.	Production (in a development environment only)
`Store.Recommendations`	Enables integration with Oracle Recommendations on Demand.	Production

Module	Description	Server
`Store.Endeca`	Enables use of Oracle Endeca Commerce.	Publishing and Production
`Store.Endeca.Assembler`	Enables querying of the Endeca Assembler.	Publishing and Production
`Store.Endeca.International`	Enables use of Oracle Endeca Commerce on an internationalized site.	Publishing and Production
`Store.Estore`	Contains the back-end classes for Commerce Reference Store.	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

If Your Environment Does Not Include Content Administration

In an environment that includes Content Administration, 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 Commerce Reference Store storefronts. This process ensures that file-based assets are managed properly through Content Administration's versioned file store.

This means, however, that if you run the Commerce Reference Store storefronts in an environment that does not include CA, the file-based assets will not be present, and the site will not function properly. Therefore, Commerce Reference Store 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 Business Control Center 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 Business Control Center

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

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

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

Viewing the Commerce Reference Store Storefronts

Once you assemble, deploy, and start up your production EAR file, you can access the various Commerce Reference Store 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 Commerce Reference Store is installed. See the *ATG Commerce Reference Store Installation and Configuration Guide* for more information.

Commerce Reference Store Documentation

The Commerce Reference Store documentation is primarily intended to address the needs of two types of users:

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

About This Manual

The *ATG Commerce Reference Store Overview* discusses the main features of the Commerce Reference Store 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 Commerce Reference Store, and to suggest starting points for further exploration and development.

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

Related Documents

In addition to this *Overview*, the documentation for Commerce Reference Store includes:

ATG Commerce Reference Store Installation and Configuration Guide
Instructions for installing, configuring, and running the Commerce Reference Store application.

ATG CRS API Reference
Javadoc descriptions of the Commerce Reference Store classes.

2 Multisite Features

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

Note: This section is focused on 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

Commerce Reference Store is an example of a multisite implementation, where multiple sites are run from the same Commerce instance and share certain data and other resources. Commerce Reference Store 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 Commerce Reference Store 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 Commerce Reference Store 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

Commerce Reference Store adds a number of site configuration properties to the Site repository for specifying values used by each of its sites. The following sections describe the site configuration properties added by the Commerce Reference Store `Store.Estore` module.

Note: These sections provide a quick description of the Commerce Reference Store 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 Commerce Reference Store 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.

Commerce Reference Store 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 Commerce Reference Store 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.

Commerce Reference Store 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. Commerce Reference Store sets this value to 15 days for all three sites.

Note: Because demonstration data goes stale over time, Commerce Reference Store uses some additional logic to determine which products are new. See the *Product Catalog* (page 85) chapter for more information.

resourceBundle

The path to the default resource bundle containing the translatable text for the store pages. Commerce Reference Store 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 Commerce Reference Store, but Commerce Reference Store 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, Commerce Reference Store uses the `/atg/store/StoreConfiguration` component's `defaultResourceBundle` property instead. This property is set to `atg.projects.store.web.WebAppResources` in Commerce Reference Store.

defaultPageSize

The default number of products to be displayed on various site pages, such as the wish list and order history page. Commerce Reference Store sets this value to 12 for all three sites.

Note: Because they use cartridges to render their content, the category and default browse pages use a different mechanism for setting the page size. See the *Using Cartridges* (page 115) chapter for more information.

cssFile

The path for the base filename of the cascading stylesheets for the site. Commerce Reference Store 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. Commerce Reference Store includes additional logic for resolving the full pathname when it uses this property. See Cascading Stylesheets (CSS) (page 24) for more information.

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

largeSiteIcon

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

defaultCountry

The default country for the site. Commerce Reference Store 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. Commerce Reference Store 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. Commerce Reference Store 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. Commerce Reference Store sets this property to enabled for all three sites.

FromAddress Properties

Commerce Reference Store includes a number of `FromAddress` properties:

- `backInStockFromAddress`
- `newPasswordFromAddress`
- `orderConfirmationFromAddress`
- `orderShippedFromAddress`
- `promotionEmailAddress`

These properties specify 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.

Commerce Reference Store sets these properties to:

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

priceSliderMaximumValue, priceSliderMinimumValue

Sets the upper and lower bounds for the Price Slider control that appears in the left hand rail. The Price Slider is rendered through the `PriceSlider` cartridge. See PriceSlider (page 132) in the *Using Cartridges* (page 115) chapter for more information.

Site Data Sharing

This section discusses data and resources shared by the three Commerce Reference Store sites.

Commerce Reference Store Uses the Default Site Category

Commerce Reference Store 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

Commerce Reference Store 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 (page 13) 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, Commerce Reference Store 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. Commerce Reference Store 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.

Commerce Reference Store 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 Commerce Reference Store 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 (page 15) 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 Commerce Reference Store'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 Commerce Reference Store, see Changing Site Context (page 15). 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 Platform 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

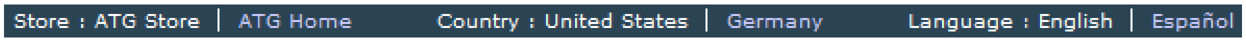
Commerce Reference Store implements a path-based URL strategy that defines `/crs` as the context root for the entire application along with the following virtual context roots for the individual sites:

- 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 Platform Programming Guide*.

Changing Site Context

Commerce Reference Store includes links for changing the store, country, and language context.


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

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. Commerce Reference Store 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. Commerce Reference Store 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. Commerce Reference Store uses the `/atg/dynamo/droplet/multisite/SharingSitesDroplet` to retrieve these IDs:

- For the store picker, Commerce Reference Store 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, Commerce Reference Store 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/dynamo/droplet/multisite/SharingSitesDroplet"/>
  <dsp:importbean bean="/atg/dynamo/droplet/ForEach"/>

  <dsp:importbean bean="/atg/multisite/Site" var="currentSite"/>

  <%--
    SharingSitesDroplet returns all the sharing sites within the given site
    group. If the siteId is not provided as an input parameter, it will be
    retrieved from the SiteContext. Similarly, the ShareableTypeId may be
    provided as an optional input parameter. The droplet is used here to
    return all the sites which have the atg.ShoppingCart sharable
    type with the current site, these are stores which share the same cart.
  --%>
  <dsp:droplet name="SharingSitesDroplet" shareableTypeId="atg.ShoppingCart"
                                        var="sharingSites">
    <dsp:oparam name="output">
```

```
      <%-- Ensure we have more than 1 site --%>
      <c:if test="${fn:length(sharingSites.sites) > 1}">

        <dl id="atg_store_sites">
          <dt>
            <%-- Display the Sites text --%>
            <fmt:message key=
                       "navigation_internationalStores.internationalStoresTitle"/>
            <fmt:message key="common.labelSeparator"/>
          </dt>

          <%--
            ForEach droplet renders the open parameter output for each
            element in its array input parameter. For each site found render
            a link to the site.
          --%>
          <dsp:droplet name="ForEach" array="${sharingSites.sites}"
                                      sortProperties="-name"
                                      var="current">
            <dsp:oparam name="output">
              <dsp:setvalue param="site" value="${current.element}"/>

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

              <%-- Display a link to the other sites --%>
              <dd class="<crs:listClass count="${current.count}"
                                         size="${current.size}"
                                         selected="${siteId == currentSite.id}" />">
                <c:choose>
                  <%-- Display the current site name as text --%>
                  <c:when test="${siteId == currentSite.id}">
                    <dsp:valueof value="${siteName}" />
                  </c:when>
                  <%-- Otherwise generate a link to the site --%>
                  <c:otherwise>
                    <dsp:include
                          page="/global/gadgets/crossSiteLinkGenerator.jsp">
                      <dsp:param name="siteId" value="${siteId}"/>
                      <dsp:param name="customUrl" value="/"/>
                    </dsp:include>
                    <dsp:a href="${siteLinkUrl}" title="${siteName}">
                      <c:out value="${siteName}"/>
                    </dsp:a>
                  </c:otherwise>
                </c:choose>
              </dd>
            </dsp:oparam>
          </dsp:droplet>
        </dl>
      </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/dynamo/droplet/ComponentExists"/>
```

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

<dsp:importbean bean="/atg/multisite/Site" var="currentSite"/>
<dsp:importbean bean="/atg/store/droplet/DisplayCountryDroplet"/>

<%--
  ComponentExists droplet conditionally renders one of its output parameters
  depending on whether or not a specified Nucleus path currently refers to a
  non-null object.  It it used to query whether a particular component has been
  instantiated, in this case the InternationalStore. If the InternationalStore
  component has been instantiated then we render the alternate regions.
--%>
<dsp:droplet name="ComponentExists" path="/atg/modules/InternationalStore">
  <dsp:oparam name="true">
    <%--
      SharingSitesDroplet returns all the sharing sites within the given site
      group. If the siteId is not provided as an input parameter, it will be
      retrieved from the SiteContext. Similarly, the ShareableTypeId may be
      provided as an optional input parameter. The droplet is used here to
      return all the sites which have the crs.RelatedRegionalStores sharable
      type with the current site, these are the regional stores.
    --%>
    <dsp:droplet name="SharingSitesDroplet"
                 shareableTypeId="crs.RelatedRegionalStores"
                 var="sharingSites">
      <dsp:oparam name="output">
        <dsp:getvalueof var="sites" param="sites"/>

        <%-- Ensure we have stores in different regions --%>
        <c:if test="${fn:length(sharingSites.sites) > 1}">
          <dl id="atg_store_regions">
            <%-- Display the Country text --%>
            <dt>
              <fmt:message key="navigation_internationalStores.RegionsTitle"/>
              <fmt:message key="common.labelSeparator"/>
            </dt>

            <%--
              ForEach droplet renders the open parameter output for each
              element in its array input parameter. For every regional site
              render a link to it.
            --%>
            <dsp:droplet name="ForEach" array="${sharingSites.sites}"
                                        var="current">
              <dsp:oparam name="output">
                <dsp:setvalue param="site" value="${current.element}"/>

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

                <dsp:getvalueof var="siteDefaultCountry"
                                param="site.defaultCountry"/>
                <dsp:getvalueof var="siteDefaultLanguage"
                                param="site.defaultLanguage"/>

                <%--
                  DisplayCountryDroplet takes a locale language key and country
                  code and returns the corresponding country display name in the
                  user's locale.
```

```
                    --%>
                    <dsp:droplet name="DisplayCountryDroplet"
                                 countryCode="${siteDefaultCountry}"
                                 language="${siteDefaultLanguage}"
                                 var="siteCountry">

                      <dsp:oparam name="output">
                        <c:set var="countryName"
                               value="${siteCountry.displayCountryName}"/>
                      </dsp:oparam>

                    </dsp:droplet>

                    <%-- Display a link to the related regional store --%>
                    <dd class="<crs:listClass count="${current.count}"
                                         size="${current.size}"
                                         selected="${siteId == currentSite.id}" />">
                      <c:choose>
                        <%-- Display the current region name as text --%>
                        <c:when test="${siteId == currentSite.id}">
                          <dsp:valueof value="${countryName}" />
                        </c:when>
                        <%-- Otherwise generate a link to the related store --%>
                        <c:otherwise>
                          <dsp:include page=
                                       "/global/gadgets/crossSiteLinkGenerator.jsp">
                            <dsp:param name="siteId" value="${siteId}"/>
                            <dsp:param name="customUrl" value="/"/>
                          </dsp:include>
                          <dsp:a href="${siteLinkUrl}" title="${countryName}">
                            <c:out value="${countryName}"/>
                          </dsp:a>
                        </c:otherwise>
                      </c:choose>
                    </dd>
                  </dsp:oparam>
                </dsp:droplet>
              </dl>
            </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* (page 177) chapter for more information on the International module.

3 Pages and Navigation

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

Note that Commerce Reference Store includes a mix of Endeca-driven pages (category pages and search results pages) and ATG-driven pages (all others, including product detail pages). This chapter primarily discusses ATG-driven pages, including how links from ATG-driven pages to Endeca-driven pages are generated. For information about the Endeca-driven pages themselves, see the *Using Cartridges* (page 115) chapter.

JSP Page Architecture

The JSP pages in Commerce Reference Store 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. The use of gadgets makes it possible to build pages from reusable elements. For example, the `store.war/navigation/gadgets/clickToCallLink.jsp` gadget, which is used on a number of pages, determines if the Click to Call integration is enabled, and if it is, renders a `<div>` element to serve as an anchor for placing a Click to Call link:

```
<dsp:page>
```

```
  <dsp:importbean bean="/atg/dynamo/droplet/ComponentExists"/>
  <%--
    If the Click To Call feature is disabled, this entire div can be safely
    bypassed. We can test if Click To Call is installed by the existence
    of the clicktoconnect Configuration component.
  --%>
  <dsp:droplet name="ComponentExists" path="/atg/clicktoconnect/Configuration">

    <dsp:oparam name="true">
      <%--
        In order to display the ClickToCall button we need to add an anchor in the
        place we want the ClickToCall button to be rendered. This anchor is the
        HTML div below. Its id attribute must match the 'Relative to Layer ID'
        property of the WebCare link
      --%>
      <dsp:getvalueof var="pageName" param="pageName"/>
      <div id="atg_store_c2c_${pageName}">&nbsp;</div>
    </dsp:oparam>
  </dsp:droplet>
</dsp:page>
```

Custom Tags

Commerce Reference Store includes the Commerce Reference Store 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 Commerce Reference Store 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 the `store.war/checkout/confirmResponse.jsp` page in the checkout process:

```
<crs:checkoutContainer currentStage="success" title="${title}">
  <jsp:body>
     <dsp:include page="gadgets/confirmResponse.jsp"/>
  </jsp:body>
</crs:checkoutContainer>
```

In this example, the `<crs:checkoutContainer>` tag displays an indicator that the order has been placed successfully, and the enclosed `<jsp:body>` tag includes a gadget that renders information about the order.

JavaScript and Cascading Stylesheets

All Commerce Reference Store 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 Commerce Reference Store.

Note that the use of JavaScript is required. Commerce Reference Store does not work properly if JavaScript is disabled in the customer's browser.

JavaScript

The JavaScript used by Commerce Reference Store is stored in external `.js` files that are loaded by the `store.war/includes/pageStartScript.jsp` file, which is included in most Commerce Reference Store 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 Commerce Reference Store JavaScript files include:

- Core Dojo libraries
- Rich shopping cart widget
- Other Commerce Reference Store JavaScript

Core Dojo Libraries

Commerce Reference Store uses the Dojo 1.6.1 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 Commerce Reference Store, you should create an optimized Dojo build. The Dojo build scripts are included in this directory:

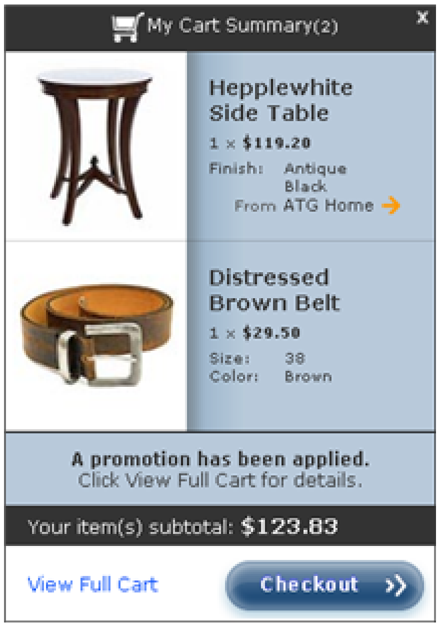
```
<ATG10dir>/WebUI/j2ee-apps/WebUI.ear/dojo-1-6-1.war/util/buildscripts/
```

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, Commerce Reference Store has a rich shopping cart user interface that is implemented as a custom Dojo widget. This widget, which is available on most pages, creates a drop-down display that shows the products in the shopping cart, information about promotions that have been applied, links to the full shopping cart page and checkout page, and a subtotal for the order. 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 `.js` files that extend the standard Dojo widget classes. The main files are:

- `store.war/javascript/widget/RichCartSummary.js` -- This is the primary 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 Commerce Reference Store JavaScript

Commerce Reference Store also includes a number of other JavaScript functions. Some of these are implemented as Dojo widgets, but a few are not. These functions are stored in the JavaScript files in the `store.war/javascript/` and `store.war/javascript/widgets/` directories. These files include implementations of features such as the SKU pickers on product detail pages.

Cascading Stylesheets (CSS)

Most of the styling information for Commerce Reference Store 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.)

Commerce Reference Store 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 Commerce Reference Store 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`

Most Commerce Reference Store pages have a `<crs:pageContainer>` tag, which includes the `store.war/includes/pageStart.jsp` file. Similarly, popups have a `<crs:popupPageContainer>` tag, which includes the `store.war/includes/popupStart.jsp` file. These JSP files contain code for determining which site-specific CSS files to load. 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 of the CSS files to include.

The individual filenames are determined by appending language-specific and browser-specific strings to the base filename. For example, `store.war/css/site/store_de.css` contains German-specific styling for ATG Store Germany, and `store.war/css/site/home_es_ie.css` contains Spanish-specific styling for ATG Store US in Internet Explorer.

Other CSS Files

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

ATG-Driven Pages

Commerce Reference Store category pages and search results pages are driven by Oracle Endeca Commerce. These pages are discussed in the *Using Cartridges* (page 115) chapter. All other pages are driven by Oracle ATG Web Commerce. This section discusses the various types of ATG-driven pages used in Commerce Reference Store.

Product Detail Pages

Commerce Reference Store uses a small number of template pages for displaying individual products. The main product template pages are:

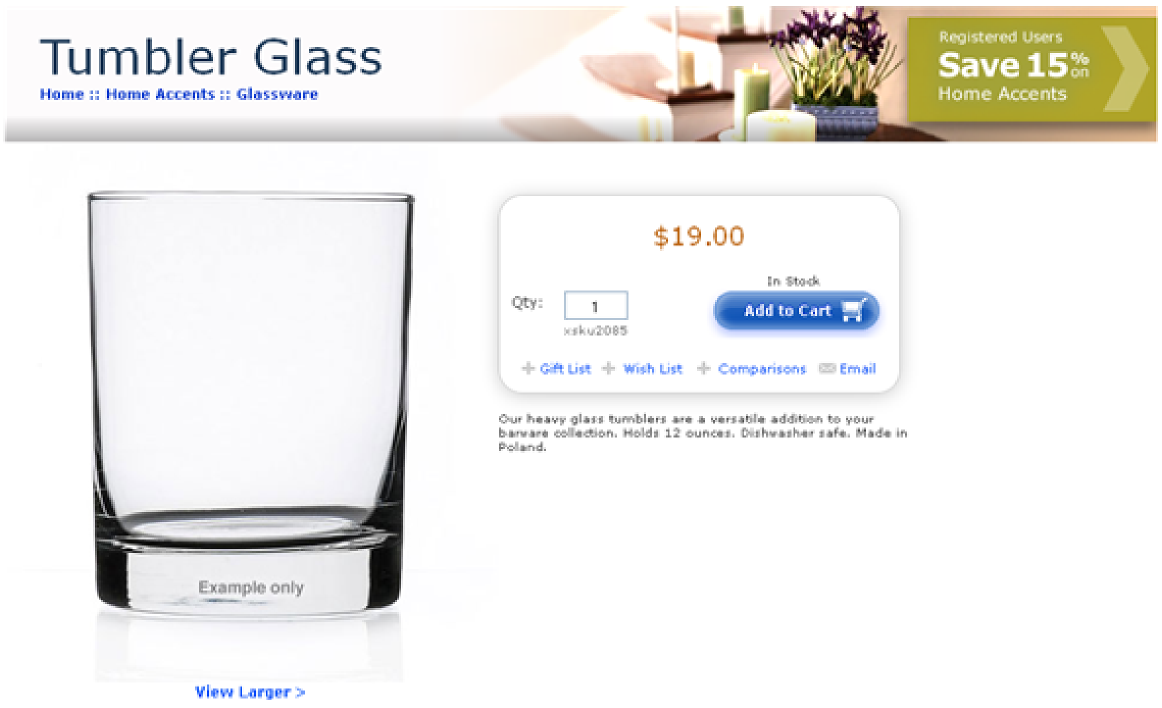
- `store.war/browse/productDetailSingleSku.jsp` -- Displays product details for a product that has a single SKU.
- `store.war/browse/productDetailMultiSku.jsp` -- Displays product details for a product with multiple SKUs. Provides a separate quantity field for each SKU.
- `store.war/browse/productDetailColorSizePicker.jsp` -- 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 for a given product is specified by the item's `template` property. 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 products, the actual product to display is specified through query parameters. The product itself is specified with the `productId` query parameter, and its parent category is specified with the `categoryId` query parameter. For example:

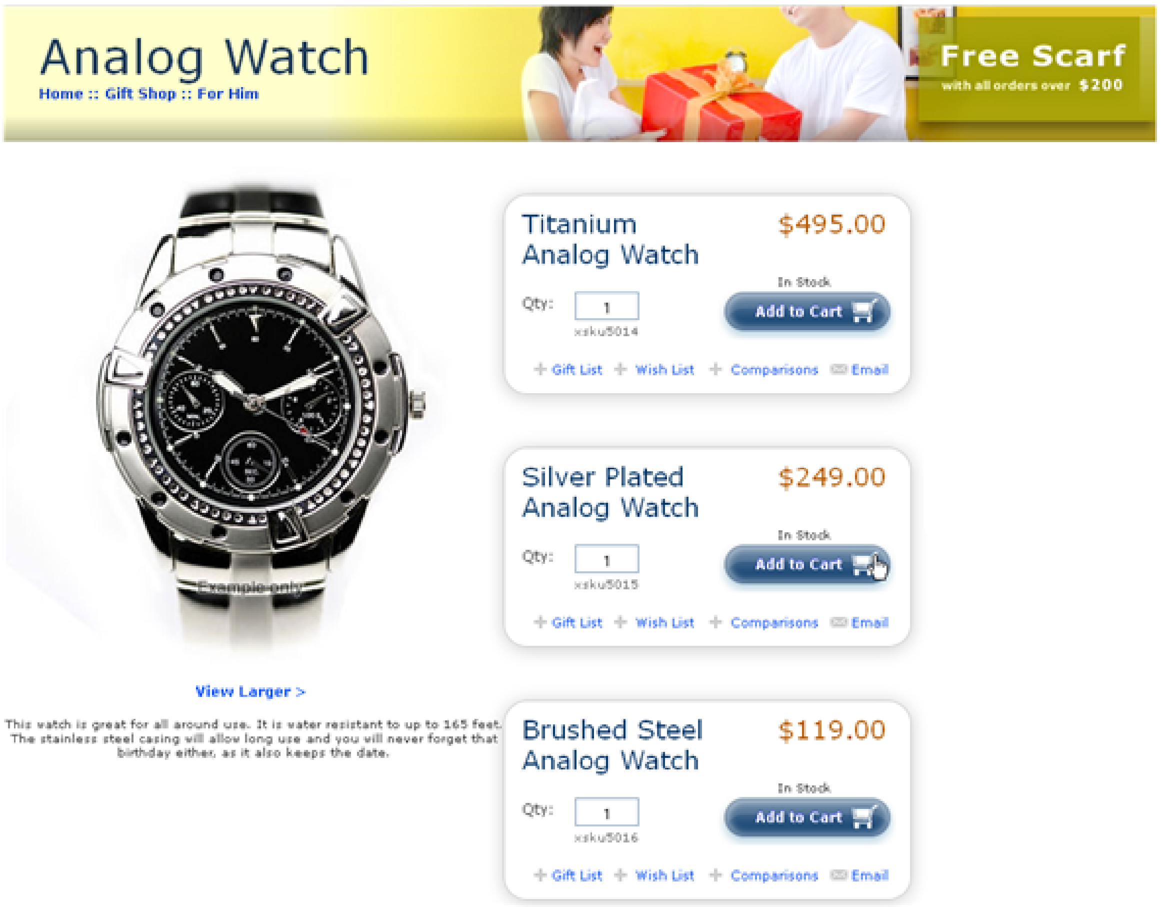
```
/browse/productDetailSingleSku.jsp?productId=xprod2035&categoryId=cat10056
```

The product detail page that a specific product is configured to use 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:



Analog Watch

Home :: Gift Shop :: For Him

View Larger >

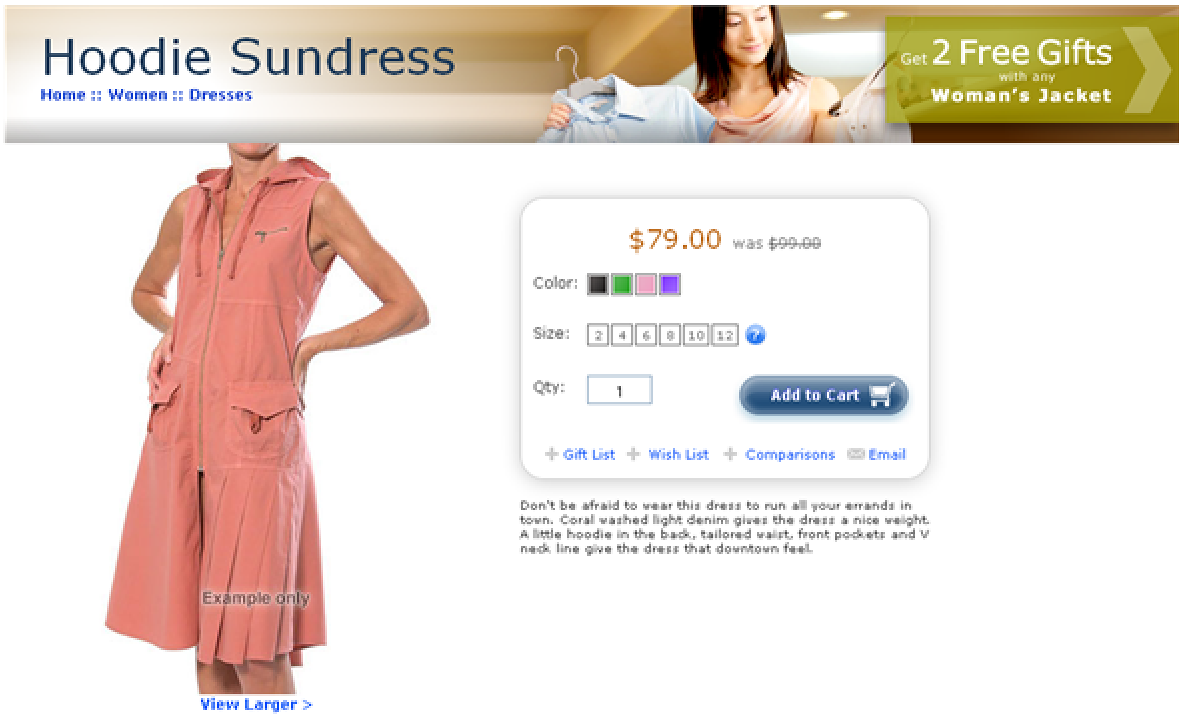
This watch is great for all around use. It is water resistant to up to 165 feet. The stainless steel casing will allow long use and you will never forget that birthday either, as it also keeps the date.

Titanium Analog Watch $495.00
In Stock
Qty: 1
xsku5014
Add to Cart
Gift List Wish List Comparisons Email

Silver Plated Analog Watch $249.00
In Stock
Qty: 1
xsku5015
Add to Cart
Gift List Wish List Comparisons Email

Brushed Steel Analog Watch $119.00
In Stock
Qty: 1
xsku5016
Add to Cart
Gift List Wish List Comparisons Email

Most products with multiple SKUs use a template that includes a SKU picker. Commerce Reference Store 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 (for example, 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

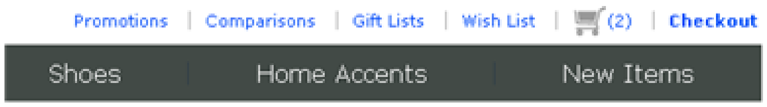
The `store.war/browse/gadgets/pickerColorPicker.jspf` page fragment uses the data in the `ColorSizeDroplet.Color` objects to render the color portion of the color/size picker. The `store.war/browse/gadgets/pickerSizePicker.jspf` page fragment uses the data in the `ColorSizeDroplet.Size` objects to render 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* (page 103) chapter for more information about availability statuses.

Product Lists

In addition to the product templates, Commerce Reference Store 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:



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/comparisons/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 Commerce Reference Store 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 Commerce Reference Store `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. Commerce Reference Store 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.

Commerce Reference Store Extensions

Commerce Reference Store uses standard Commerce gift and wish list functionality, with the following extensions:

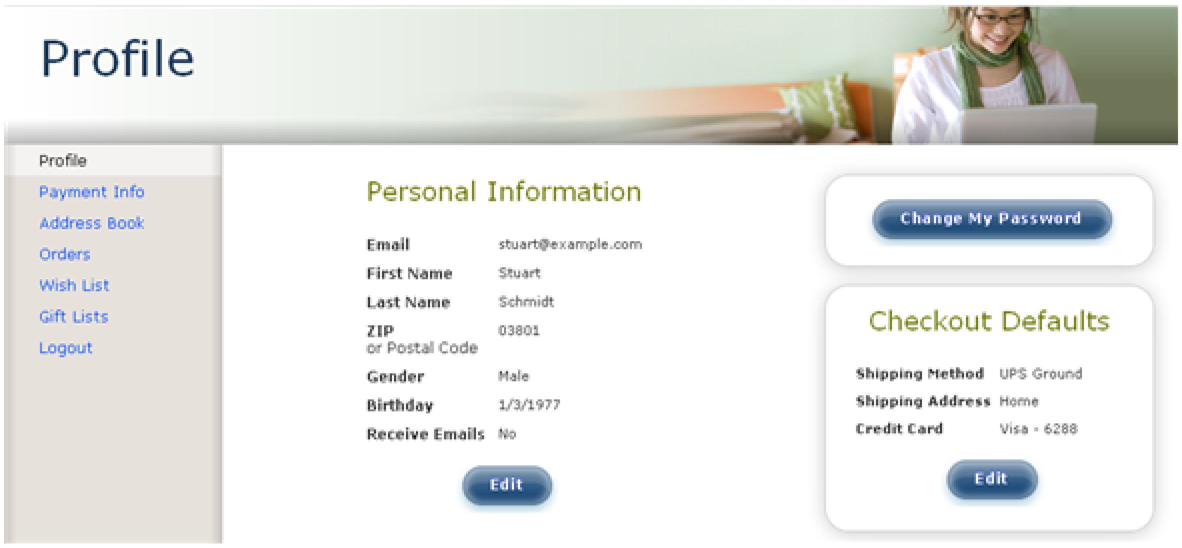
- Commerce Reference Store adds a derived property, `quantityRemaining`, to the `gift-item` item descriptor in the gift list repository. This property captures the quantity of a gift item that hasn't been purchased yet. It is defined in the `/atg/commerce/gifts/giftlists.xml` file in `<ATG10dir>/CommerceReferenceStore/Store/EStore/config/config.jar`. `quantityRemaining` is of type `atg.projects.store.gifts.GiftlistQuantityRemaining`, a Commerce Reference Store-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.

- Commerce Reference Store changes the class of the `/atg/commerce/gifts/GiftlistFormHandler` component to `atg.projects.store.gifts.StoreGiftlistFormHandler`, which is a Commerce Reference Store-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	1/3/1977
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

Commerce Reference Store 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 8 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 (page 54) section in the *Personalization* (page 37) 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 (page 105) section of the *Inventory Management* (page 103) 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. Commerce Reference Store then displays a form that the customer fills out, specifying the recipient's email address and the message text.

- `store.war/emailtemplates/registeredUser.jsp` -- Message confirming that the customer has successfully created a new account.

- `store.war/emailtemplates/changePassword.jsp` -- Message confirming that the customer has changed the password for his or her account.

- `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 (page 55) section in the *Personalization* (page 37) 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 (page 55) section in the *Personalization* (page 37) 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/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.

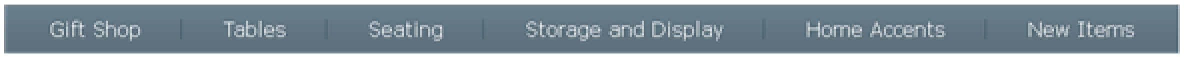
Unlike other Commerce Reference Store pages, the e-mail JSPs contain inline styling information, rather than referring to external `.css` files. This is because some e-mail clients don't support external `.css` files. As a result, the styling for a given page is the same for all sites.

Linking to Endeca-Driven Pages

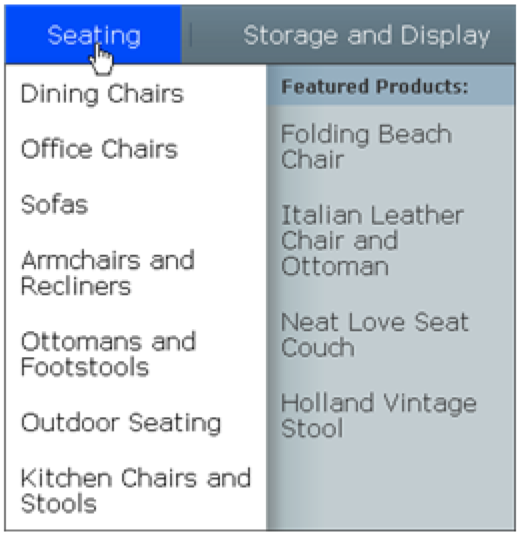
This section describes places in Commerce Reference Store ATG-driven pages that link to Endeca-driven pages, and how the links are constructed. Creating these links involves use of the dimension value caching facility described in the *ATG Endeca Integration Guide*.

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 (which is included by the `<crs:pageContainer>` tag) creates a menu bar. The menu bar, which is displayed at the top of most pages, lists the top-level categories of the site. For example, on the ATG Home site, the top-level categories are Gift Shop, Tables, Seating, Storage and Display, Home Accents, and New Items:



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



Selecting a category from the menu bar takes the customer to the corresponding category page. Because category pages are Endeca-driven, they are accessed using Endeca query URLs. These URLs encode Endeca dimension values in query parameters. For example, the URL for a category page might be:

```
http://hostname:port/crs/storeus/browse?N=10059
```

The `/browse` portion of the URL triggers a page request in the Endeca Assembler, and the `N=10059` query parameter specifies the ID of the `product.category` dimension value that corresponds to the ATG category.

To construct the links for the categories, `catalog.jsp` first obtains the category IDs of the top-level categories by looking up the child categories of the current site's root category:

```
<dsp:getvalueof var="topLevelCategories"
    bean="Profile.catalog.rootNavigationCategory.childCategories" />
```

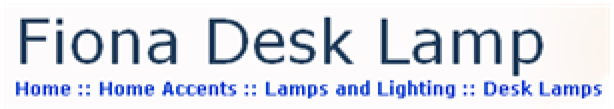
It also obtains the category IDs of the subcategories by iterating over the list of top-level category IDs and looking up the child categories of the top-level categories.

For each category ID it retrieves, `catalog.jsp` invokes the `/atg/commerce/endeca/cache/DimensionValueCacheDroplet` servlet bean to look up the URL of the corresponding category page in the dimension value cache.

See the *ATG Endeca Integration Guide* for information about `DimensionValueCacheDroplet` and the dimension value cache.

Breadcrumb Trail

Product detail pages display a “breadcrumb” trail that reflects the navigational path to the page. For example, the following illustration shows the breadcrumb trail for a product that a customer accessed by navigating the category path Home > Home Accents > Lamps and Lighting > Desk Lamps:



Each entry in the trail is a link to the corresponding Endeca-driven category page. So, in the trail 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 catalog 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.

Note that the breadcrumb trail displayed on product detail pages is distinct from the dimension-selection breadcrumbs displayed on category and browse (search results) pages, which are described in the *Using Cartridges* (page 115) chapter. The breadcrumbs that appear on product detail pages are created using the ATG Commerce historical navigation mechanism described in the *ATG Commerce Guide to Setting Up a Store*.

To track a customer’s navigational path, links to products pass parameters to the `/atg/commerce/catalog/CatalogNavHistoryCollector` servlet bean. These parameters specify modifications to the navigational history, which 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. This gadget iterates over the `navHistory` property to get the List of ATG categories, and then uses the `/atg/commerce/endeca/cache/DimensionValueCacheDroplet` servlet bean to look up the URLs of the corresponding Endeca-driven pages in the dimension value cache. See the *ATG Endeca Integration Guide* for information about this droplet and the cache.

Cross-Site Links

Commerce Reference Store 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
- Recently viewed products on the product detail page

- Gift and wish lists
- Shopping cart
- Product comparisons

Commerce Reference Store 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*.

Tags for Visual Merchandising

Several Commerce Reference Store pages are configured to work with the Visual Merchandising features, which let Merchandising users select assets to organize and edit directly from a preview session. These pages use `<preview:repositoryItem>` tags to explicitly mark items (products and categories) on pages.

The following Commerce Reference Store pages are tagged with `<preview:repositoryItem>` tags:

- `store.war/browse/gadgets/relatedProducts.jsp`
- `store.war/browse/productDetailDisplay.jsp`
- `store.war/browse/productDetailMultiSkuContainer.jsp`
- `store.war/browse/productDetailSingleSkuContainer.jsp`
- `store.war/global/gadgets/productListRangeRow.jsp`
- `store.war/global/gadgets/promotedProductRenderer.jsp`
- `store.war/navigation/gadgets/breadcrumbs.jsp`
- `store.war/navigation/gadgets/catalog.jsp`
- `store.war/promo/gadgets/homePromotionItemRenderer.jsp`
- `store.war/browse/gadgets/categoryChildProductsRenderer.jsp`
- `store.war/cartridges/Breadcrumbs/Breadcrumbs.jsp`
- `store.war/cartridges/HeaderBanner-ATGCategory/HeaderBanner-ATGCategory.jsp`
- `store.war/cartridges/RefinementMenu/RefinementMenu.jsp`

For more information about configuring pages to use the Visual Merchandising features, including best practice examples from Commerce Reference Store, see the *ATG Merchandising Administration Guide*. For information about using the Visual Merchandising features, see the *ATG Merchandising Guide for Business Users*.

4 Personalization

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

User Profile Extensions

Because every 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 Commerce. This section describes the custom properties and item descriptors that were added for Commerce Reference Store 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 the User Item Descriptor

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

age

The age property is a Commerce Reference Store-specific derived property calculated using the `dateOfBirth` property that is included in Commerce. This property is used by the `ThirtySomethings` and `Young` user segments to identify customers of a particular age. See User Segments (page 152) for more details.

autoLogin

Commerce Reference Store sets the `autoLogin` property default to `true`. This enables the 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. Commerce Reference Store extends the 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 Commerce Reference Store-specific property that stores the last category a customer looked at. For more information on how the categoryLastBrowsed is calculated, see Tracking a Customer's Catalog Navigation (page 121) in the *Using Cartridges* (page 115) chapter.

email

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

externalID

externalID is a Commerce Reference Store-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

Commerce Reference Store extends the firstName property by marking it as required.

gender

Commerce Reference Store 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 Commerce Reference Store.

itemsBought

itemsBought is a Commerce Reference Store-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 Commerce Reference Store-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

Commerce Reference Store extends the `lastName` property by marking it as required.

lastPurchaseDate

`lastPurchaseDate` is a Commerce Reference Store-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 Commerce Reference Store-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 Commerce Reference Store-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 Commerce Reference Store-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 Commerce Reference Store-specific property that allows the customer to opt in or out of marketing e-mails. New customers can specify whether to opt in or out of emails while they are registering. These options are presented to new users by the `store.war/myaccount/gadgets/registrationForm.jsp` gadget, which is included by `store.war/myaccount/registration.jsp`. Existing customers can change their opt-in setting using the `store.war/myaccount/accountProfileEdit.jsp` page. Both pages call the `/atg/store/profile/RegistrationFormHandler` component, which is an instance of `atg.projects.store.profile.StoreProfileFormHandler`, and set the `RegistrationFormHandler` component's `emailOptIn` property to either true or false. The `StoreProfileFormHandler.updateReceiveEmailProperty()` method then sets the `receivePromoEmail` property for the customer based on the value of the `emailOptIn` property.

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 (page 54) for more information on this promotion.

recentlyViewedProducts

`recentlyViewedProducts` is a Commerce Reference Store-specific property that stores a list of `recentlyViewedProduct` items that correspond to the products a customer has most recently viewed. For registered customers, the `recentlyViewedProducts` list is stored with the customer's profile. For anonymous users, the `recentlyViewedProducts` list is persisted in a transient profile for the length of the customer's session only. If the customer subsequently logs in, the data is persisted in the customer's profile as described in Additional Support for Recently Viewed Products (page 64).

referralSource

`referralSource` is a Commerce Reference Store-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.

Commerce Reference Store 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 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`. Commerce Reference Store extends the Commerce definition of `shippingAddress` with a modification to the `cascade` attribute. In 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 Commerce Reference Store 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

Commerce Reference Store 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, Commerce doesn't apply any encryption to credit card information.

Custom Item Descriptors in the Profile Repository

Commerce Reference Store adds two item descriptors to the profile repository, `backInStockNotifyItem` and `recentlyViewedProduct`.

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* (page 103) chapter.

Property	Description
`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.

recentlyViewedProduct Item Descriptor

The `recentlyViewedProduct` item descriptor supports the recently viewed products feature and it represents a single recently viewed product. A list of `recentlyViewedProduct` items is stored in each customer's `recentlyViewedProducts` profile property. The `recentlyViewedProduct` item descriptor has the properties described below.

Property	Description
`id`	A unique identifier for the `recentlyViewedProduct` item.
`product`	The product this `recentlyViewedProduct` item represents.
`siteId`	The ID of the site the product was viewed on.
`timeStamp`	A timestamp for when the product was viewed.

Capturing these property values for each recently viewed product allows Commerce Reference Store to filter the list of recently viewed products to display only those that are appropriate for the current site context, as described in Filtering Products in the Recently Viewed List (page 61).

ATG User Segments

Note: Both Oracle ATG Web Commerce and Oracle Endeca Commerce employ user segments. While they function in a similar way, this section is primarily concerned with the ATG implementation of user segments. For more information on Oracle Endeca Commerce user segments, refer to the Oracle Endeca Commerce documentation.

ATG 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 Business Control Center.

Commerce Reference Store 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, it is preferable to 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 Commerce Reference Store, 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. Commerce Reference Store 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.

ATG User Segments in Commerce Reference Store JSP Pages

These Commerce Reference Store gadgets display different content depending on the segments a customer belongs to:

- `store.war/global/gadgets/targetingRandom.jsp`
- `store.war/navigation/gadgets/homePagePromotions.jsp`
- `store.war/navigation/gadgets/homePagePromotionalItems.jsp`

All of these pages implement content targeters that define different content for different user segments. The `targetingRandom.jsp` gadget is included by a number of other JSPs that need to render targeted content, including:

- `store.war/atgsearch/searchResults.jsp`
- `store.war/atgsearch/categorySearchResults.jsp`

- `store.war/promo/gadgets/homeFeaturedProducts.jsp`
- `store.war/search/searchResults.jsp`
- `store.war/search/categorySearchResults.jsp`

For more information on Commerce Reference Store targeters, see Using Targeters, Slots, and Scenarios to Personalize Content for Customers (page 44).

CategoryOnly User Segment and the Category Page

The ATG-Endeca integration uses a user segment, `CategoryOnly`, to flag requests for the category page so that the Assembler handles them correctly. This user segment is added programmatically to the request and cannot be viewed or modified in the Business Control Center. For more information on how this user segment is created and used, see the Tracking a Customer's Catalog Navigation (page 121) section in the *Using Cartridges* (page 115) chapter.

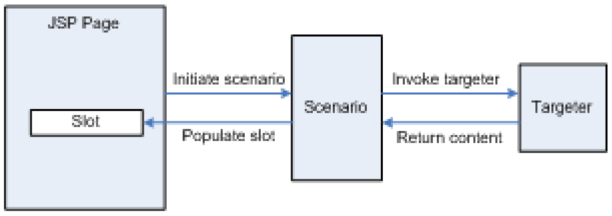
For More Information

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

Using Targeters, Slots, and Scenarios to Personalize Content for Customers

Commerce Reference Store 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 Commerce Reference Store.

Using Slots and Targeters on Endeca-driven Pages

Endeca-driven pages can take advantage of ATG slots and targeters, through the use of cartridges that reference ATG slots and targeters. When a cartridge references a slot or a targeter, it is the equivalent of a JSP page requesting content from a slot or targeter. Commerce Reference Store includes four cartridges that allow an Experience Manager user to add ATG slots or targeters to a page. The `PromotionalContent-ATGSlot` and `PromotionalContent-ATGTargeter` cartridges display promotional content items in the left hand rail of both the category and browse pages. The `ProductSpotlight-ATGSlot` and `ProductSpotlight-ATGTargeter` cartridges display products in the main content area of both category and browse pages. Each cartridge type has a property that references the ATG slot or targeter to use when retrieving content for the page where the cartridge has been placed. See the PromotionalContent-ATGSlot, PromotionalContent-ATGTargeter, ProductSpotlight-ATGSlot, and ProductSpotlight-ATGTargeter (page 126) section of the *Using Cartridges* (page 115) chapter for more details.

Commerce Reference Store Targeters

Each targeter returns either a product ID or a `promotionalContent` item. A `promotionalContent` item is a Commerce Reference Store-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 an example of the `promotionalContent` for the Save 25% on All Women's Apparel promotion:

Save 25%
On All Women's Apparel

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

Notes:

- See promotionalContent Item Descriptor (page 96) in the *Product Catalog* (page 85) chapter for more information on the `promotionalContent` item type.
- See the *Promotions* (page 65) chapter for more information about granting promotions in Commerce Reference Store.

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 for Customers (page 44). 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 Commerce Reference Store shopping cart page.

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

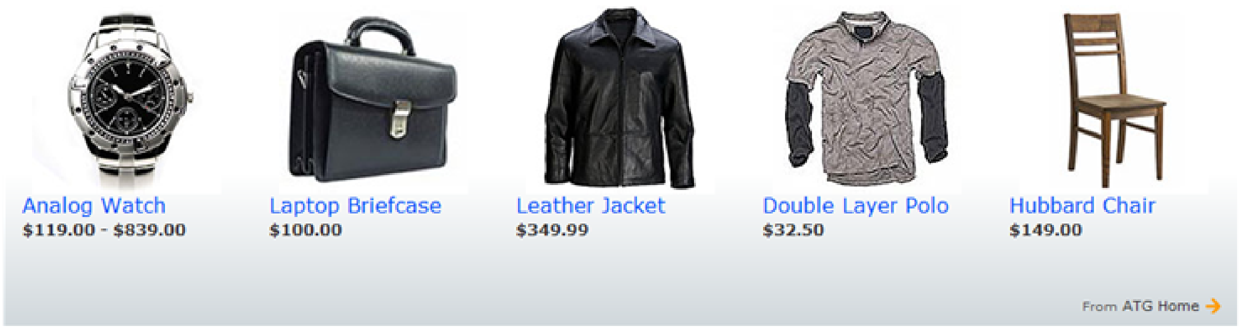
- It gives merchandisers access to the gift wrap item, through the Business Control Center, so that they can change it to point to a different product ID, if necessary. Because the gift wrap item is not part of the regular catalog hierarchy, the targeter is the only location where merchandisers can makes 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 Commerce Reference Store gift wrap features, see Gift Wrap (page 108) in the *Order Handling* (page 107) 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: Analog Watch, Laptop Briefcase, Leather Jacket, Double Layer Polo, and Hubbard Chair.



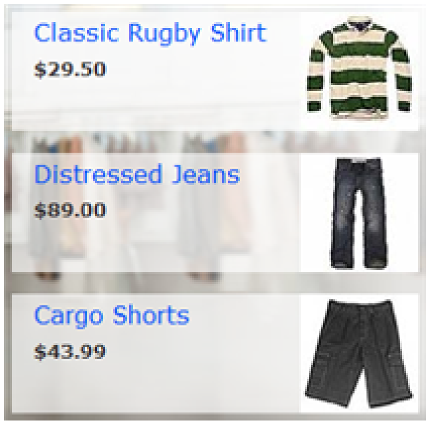
HomePromotionalItem1, HomePromotionalItem2, and HomePromotionalItem3

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 three products that are returned for the Men Only user segment on ATG Store US are: Classic Rugby Shirt, Distressed Jeans, and Cargo Shorts.

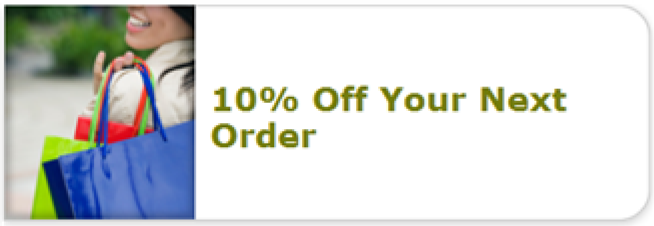


Targeters That Return promotionalContent Items

The following targeters return `promotionalContent` items.

AbandonedOrderPromotion

The `AbandonedOrderPromotion` targeter returns the 10% off your next order promotional content item, shown below, 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 that inform the customer about various promotions, such as the example below which shows the `promotionalContent` item for the Save $10 on all orders over $100 promotion:

The `promotionalContent` item returned is 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 Save 20% on any ATG Home item excluding Home Accents `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 category page and the default browse page both use these targeters. Specifically, these pages include instances of `PromotionalContent-ATGSlot` cartridges that reference the `CategoryPromtionContent1` and `CategoryPromotionContent2` ATG slots. These slots call the `categoryPromotion` scenario to get their content and the scenario, in turn, calls the `CategoryPromotionContent1` and `CategoryPromotionContent2` targeters.

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.

Commerce Reference Store Slots

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

Note: This section is referring to ATG slots that are configured in the ATG Control Center, not to be confused with Endeca slots that are configured in Experience Manager.

To view Commerce Reference Store slots:

1. Start up Commerce Reference Store and the ATG Control Center. For details on starting Commerce Reference Store, 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.

Commerce Reference Store 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`

Slot	Related Scenario
`RelatedItemsOfCart`	`RelatedItemsSlot`

Note that the first two slots in this table are referenced by cartridges on the Endeca-driven pages included with Commerce Reference Store. Specifically, the category and default browse pages each include two instances of the `PromotionalContent-ATGSlot` cartridge. The first cartridge instance references the `CategoryPromotionalContent1` slot, while the second instance references the `CategoryPromotionContent2` slot.

Commerce Reference Store 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 Commerce Reference Store scenarios using the ATG Control Center. Scenarios are not viewable in the Business Control Center.

To view Commerce Reference Store scenarios:

1. Start up Commerce Reference Store and the ATG Control Center. For details on starting Commerce Reference Store, 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`. Commerce Reference Store scenarios are stored under the `/store` directory.
5. Click a `/store` subdirectory, and then select a scenario to view it in the Scenarios editor.

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

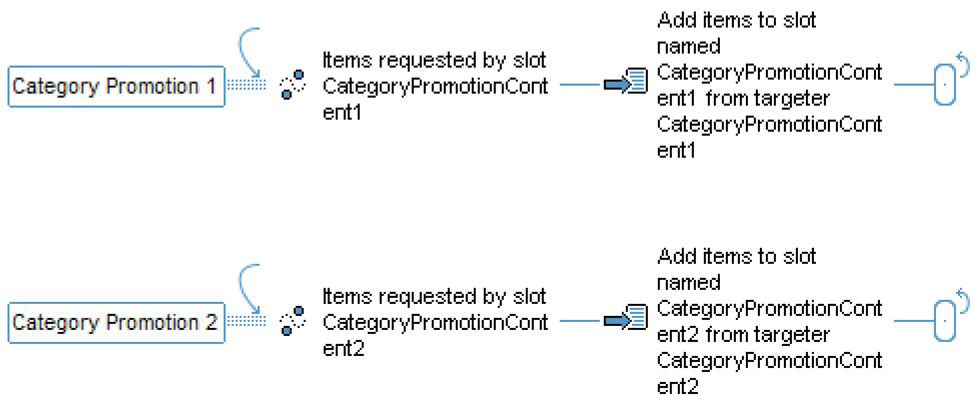
Note: Commerce Reference Store no longer uses the `CurrentPromotions` and `BestSellers` scenarios.

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 for Customers (page 44).

categoryPromotions

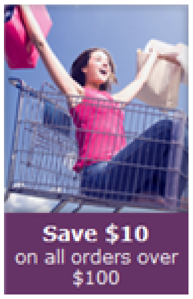
The `categoryPromotions` scenario, located in `/store/category`, looks like this:


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 an ATG slot with a `promotionalContent` item, but they use different targeters to determine which item to return.

To trace the full path of the `categoryPromotions` scenario, we must begin with the `PromotionalContent-ATGSlot` cartridges that exist on the Endeca-driven category and default browse pages. Each page includes two instances of the `PromotionalContent-ATGSlot` cartridge, one that references the `CategoryPromotionContent1` slot and another that references the `CategoryPromotionContent2` slot. When these pages are rendered, their ATG slots are executed and request content.

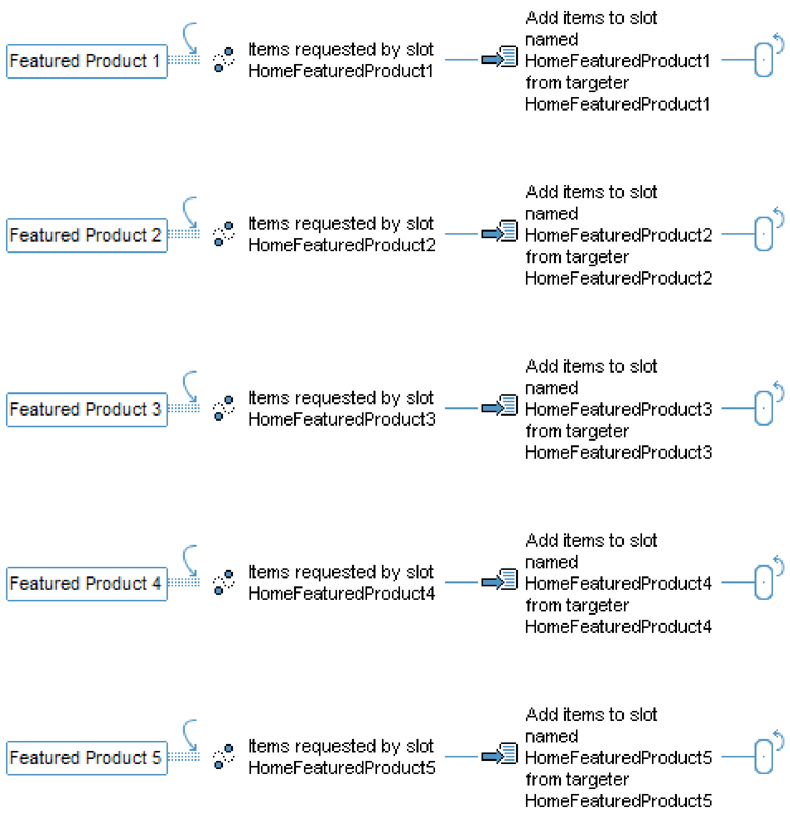
The first segment in the `categoryPromotions` scenario is invoked when the `CategoryPromotionContent1` slot requests content. To supply the slot with a `promotionalContent` item, the `categoryPromotions` scenario calls the `CategoryPromotionContent1` targeter. This targeter returns different `promotionalContent` 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, shown below.



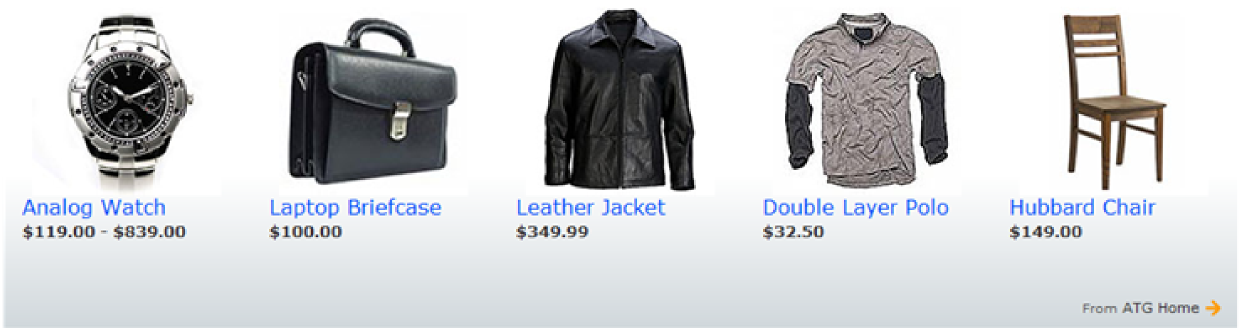
The second segment is invoked when the `CategoryPromotionContent2` slot requests content and it uses the `CategoryPromotionContent2` targeter to determine which `promotionalContent` item to return.

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

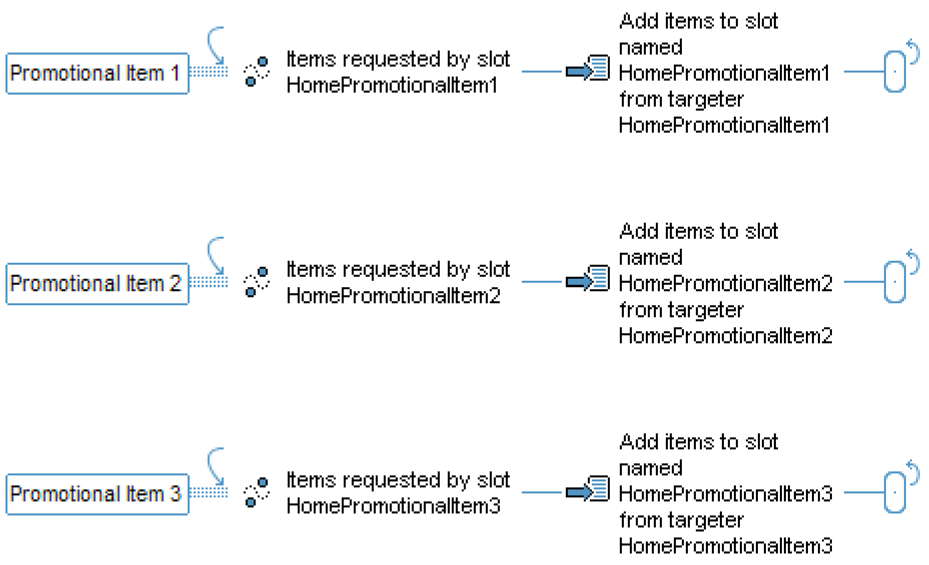
The `HomeFeaturedProducts` scenario has five segments, which all follow the same pattern: a slot initiates a segment by requesting content and the scenario invokes a targeter to provide that content. All of the targeters return a static product ID, based on user segment and site context. For example,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



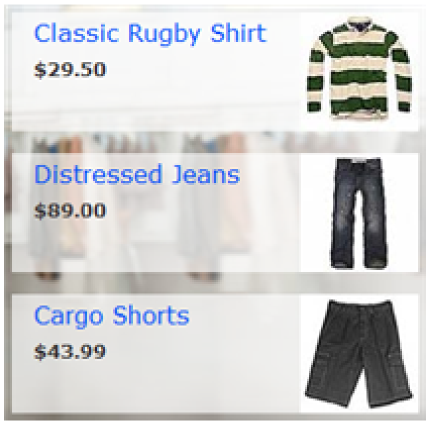
The first segment of the `HomeFeaturedProducts` scenario is initiated by the `HomeFeaturedProduct1` slot, the second is initiated by the `HomeFeaturedProduct2` slot, the third is initiated by the `HomeFeaturedProduct3` slot, and so on. All five slots appear in the `store.war/promo/gadgets/homeFeaturedProducts.jsp` 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

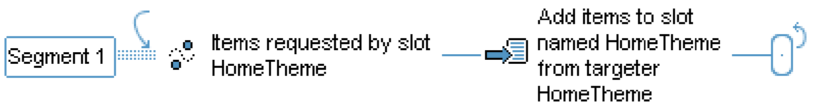
The `HomePromotionalItems` scenario has three segments, which all follow the same pattern: a slot initiates a segment by requesting content and the scenario invokes a targeter to provide that content. All of the targeters return a product ID, based on user segment and site context. For example, the following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



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/homePagePromotionalItems.jsp` gadget. Content for each slot is provided by the `PromotedProduct1`, `PromotedProduct2`, and `PromotedProduct3` 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/homePagePromotions.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:

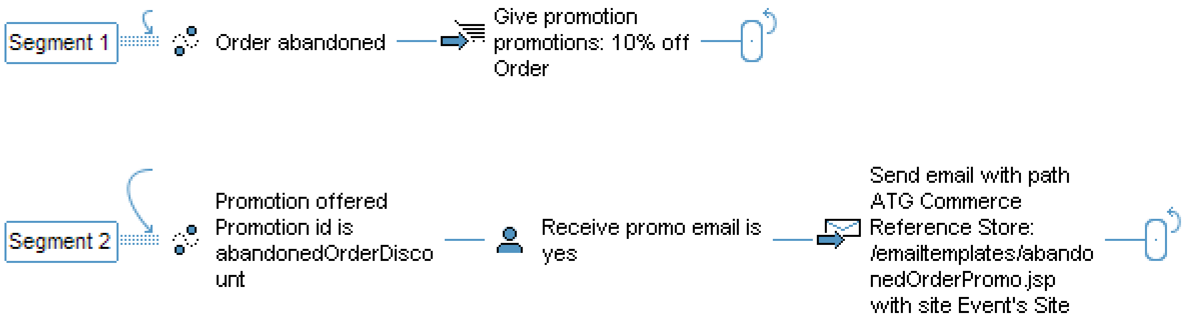
Scenarios That Grant Promotions

The scenarios described in this section grant promotions to customers.

Note: For detailed information on the promotions themselves, see the *Promotions* (page 65) chapter.

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 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.

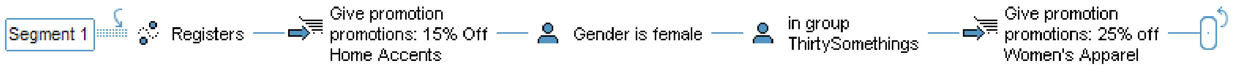
Segment 2 is responsible for notifying customers of the promotion they have been granted. When the 10% off Order promotion has been added to a customer's activePromotions property, segment 2 is invoked. This segment determines if the customer's receivePromoEmail profile property is set to Yes (code 1) 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 grants the Registered Users – Save 15% on 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.

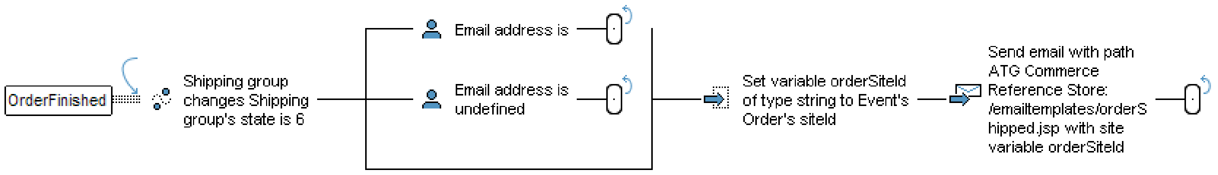
Note: For more information on the atg.dps.Register event message, see *Using Scenario Events* in the *ATG Personalization Programming Guide*.

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

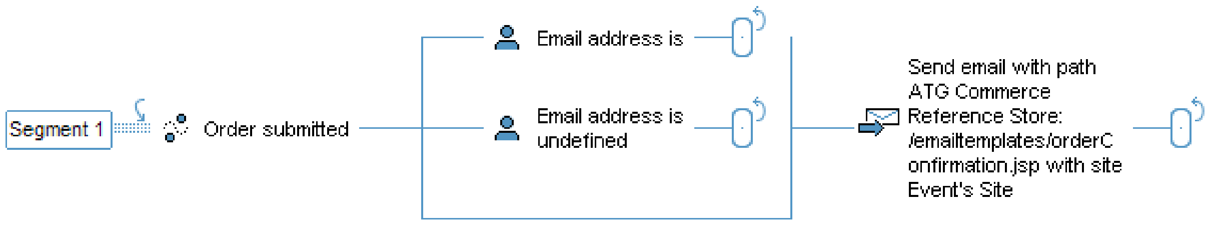
The OrderShipped scenario sends a confirmation e-mail after a customer's order has been shipped. This scenario listens for the atg.commerce.fulfillment.scenario.ShippingGroupModified event message that indicates that a shipping group has been shipped. These messages are sent by the `/atg/commerce/fulfillment/OrderChangeHandler`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

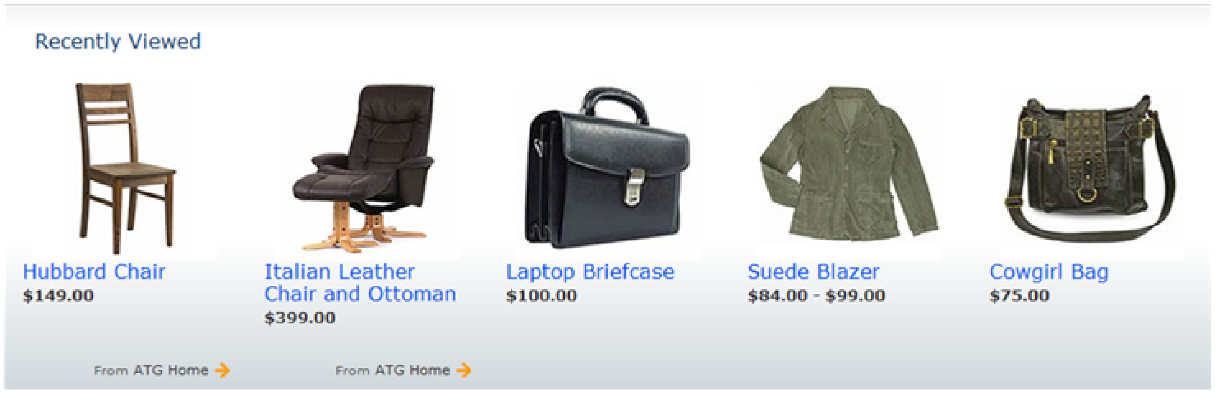

Segment 1
Order submitted
Email address is
Email address is undefined
Send email with path ATG Commerce Reference Store: /emailtemplates/orderConfirmation.jsp with site Event's Site

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.

Recently Viewed Products

The recently viewed products feature dynamically tracks a customer's browsing history as he or she views products on the Commerce Reference Store storefronts. Each time a customer views a product on the product detail page, the product is added to a list of recently viewed products. You can configure this list by setting the maximum number of products it can hold, the maximum number of products to be displayed, and how long a product should remain in the list, measured in hours. The recently viewed list is rendered in a panel at the bottom of the product detail page.



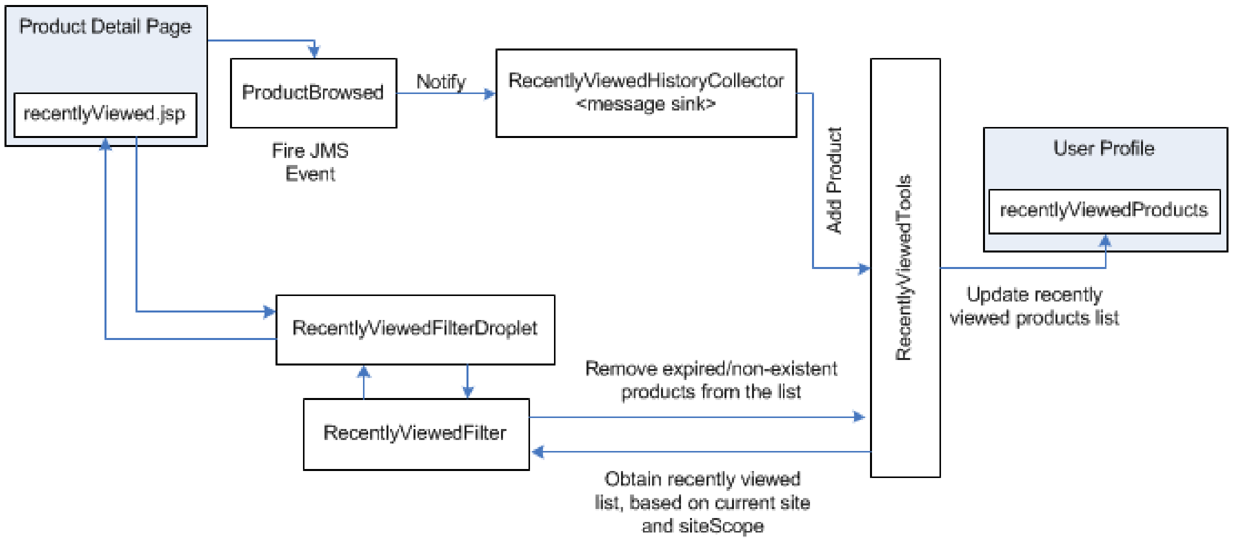
Note: The recently viewed products feature functions at a product level. In other words, it tracks the viewing of products, not the viewing of individual SKUs.

Profile Extensions that Support the Recently Viewed List

Commerce Reference Store adds two extensions to the profile repository to support the recently viewed products feature, a `recentlyViewedProduct` item descriptor and a `recentlyViewedProducts` profile property. The `recentlyViewedProduct` item descriptor represents a single recently viewed product. The `recentlyViewedProducts` profile property is a list of `recentlyViewedProduct` items and it represents the products a customer has most recently viewed. For more detail on these extensions, see User Profile Extensions (page 37).

Recently Viewed Products Architecture

The following illustration shows the architecture that supports the recently viewed products functionality:


Product Detail Page
recentlyViewed.jsp
ProductBrowsed
Notify
Fire JMS Event
RecentlyViewedHistoryCollector <message sink>
Add Product
RecentlyViewedTools
User Profile
recentlyViewedProducts
Update recently viewed products list
RecentlyViewedFilterDroplet
Remove expired/non-existent products from the list
RecentlyViewedFilter
Obtain recently viewed list, based on current site and siteScope

This diagram shows both the process for updating the recently viewed list and the process for retrieving and rendering the Recently Viewed panel on the product detail page.

When updating the recently viewed list, the following happens:

1. The customer views a product on the product detail page, thereby generating a request for the `store.war/browse/gadgets/productLookupForDisplay.jsp` gadget.

2. The `productLookupforDisplay.jsp` gadget calls the `ProductBrowsed` component. This component dispatches a JMS `ViewItem` message.

3. The `RecentlyViewedHistoryCollector` receives the `ViewItem` message, extracts the product, and calls the `RecentlyViewedTools` component, passing it the product.

4. The `RecentlyViewedTools` component updates the `RecentlyViewedProducts` list for the current customer.

   For detailed information on this process, see Adding Products to the Recently Viewed List (page 58).

When rendering the Recently Viewed panel on the product detail page, the following happens:

1. The product detail page includes the `store.war/browse/gadgets/recentlyViewed.jsp` gadget.

2. The `recentlyViewed.jsp` gadget calls the `RecentlyViewedFilterDroplet` to initiate filtering. The `RecentlyViewedFilterDroplet` calls the `RecentlyViewedFilter` to filter the list of recently viewed products.

3. `RecentlyViewedFilter` invokes some `RecentlyViewedTools` methods to remove any non-existent or expired products from the recently viewed list, and then filters any remaining products in the list based on the site context and site scope.

4. `RecentlyViewedFilter` returns the filtered list to `RecentlyViewedFilterDroplet` for rendering in the Recently Viewed panel.

   For detailed information on this process, see Filtering Products in the Recently Viewed List (page 61).

Adding Products to the Recently Viewed List

This section describes how Commerce Reference Store detects when a product is being viewed and then adds that product to the recently viewed list.

Detecting When a Product is Being Viewed

Depending on the type of product being displayed, the product detail page may be one of three JSPs found in `store.war/browse/productDetailDisplay.jsp`, `productDetailMultiSkuContainer.jsp`, or `productDetailSingleSkuContainer.jsp`. All three pages include the `store.war/browse/gadgets/productLookupForDisplay.jsp` gadget, which calls the `/atg/commerce/catalog/ProductBrowsed` component. This component dispatches a message each time a customer browses a product on the product detail page. To receive these messages, Commerce Reference Store includes a `MessageSink` component called the `/atg/userprofiling/recentlyviewed/RecentlyViewedHistoryCollector`.

When `RecentlyViewedHistoryCollector` receives a message, it determines if the message's type is `ViewItem` and if the message's item type is `product`. If both tests are true, `RecentlyViewedHistoryCollector` calls the `RecentlyViewedTools` component to add the product to the customer's recently viewed list. `RecentlyViewedHistoryCollector` is a globally scoped component of class `atg.projects.store.userprofiling.RecentlyViewedHistoryCollector` and it has the properties shown in the table below:

Property	Description
`messageType`	A string that represents the JMS message type that `RecentlyViewedHistoryCollector` should process. This property should be set to `atg.dps.ViewItem`.
`productItemType`	A string that represents the item type that `RecentlyViewedHistoryCollect` should process. This property should be set to `product`.
`catalogRepository`	A reference to the `productCatalog` repository. Commerce Reference Store sets this property to `/atg/commerce/catalog/ProductCatalog`.
`recentlyViewedTools`	A reference to the `RecentlyViewedTools` component. Commerce Reference Store sets this property to `/atg/store/profile/recentlyviewed/RecentlyViewedTools`.

In order to specify `RecentlyViewedHistoryCollector` as a message sink, Commerce Reference Store adds the necessary configuration to the `/atg/dynamo/messaging/dynamoMessagingSystem.xml` file.

Note: See the *ATG Platform Programming Guide* for more information on message sinks and the Patch Bay messaging system.

Adding a Product to the Recently Viewed List

The `/atg/userprofiling/recentlyviewed/RecentlyViewedTools` component performs all the calls required to read and write information to and from the recently viewed list stored in the Profile repository. This component also includes some convenience methods for tasks such as determining if a product has expired and determining if the recently viewed list contains duplicates. The `RecentlyViewedTools` component is a globally-scoped component of class `atg.projects.store.userprofiling` and it has the properties shown in the table below:

Property	Description
`profileTools`	A reference to the utility component `/atg/userprofiling/ProfileTools`, used for working with user profiles. `RecentlyViewedTools` uses this component to retrieve user profiles.
`recentlyViewedSize`	An `int` that specifies the maximum number of products that should persist in the `recentlyViewedProducts` profile property. Default is -1, which means no limit.
`expiryPeriod`	An `int` that specifies the amount of time a product should remain in the `recentlyViewedProducts` list, defined in hours. The default for this property is 0, which indicates that there is no expiry period.

Property	Description
`siteScope`	This property determines the behavior of recently viewed lists in a multisite environment. There are three potential values: `all`, `current`, and shareable type ID. This setting affects how products are added to the recently viewed list and also how they are filtered for display in the Recently Viewed Products panel. For detailed information on the effect of `siteScope` when adding products to the recently viewed list, continue reading this section. For detailed information on the effect of `siteScope` when filtering recently viewed products, see Filtering Products in the Recently Viewed List (page 61).
`currentDatePath`	The path to the `CurrentDate` component, which `RecentlyViewedTools` uses to retrieve current system times for timestamps.

In preparation for adding a product to the recently viewed list, `RecentlyViewedTools` retrieves the current recently viewed list from the customer's profile and scrubs it of any products that are out of date or no longer exist. It also creates a new `recentlyViewedProduct` item with the current site's ID and the current timestamp. This is the item that will potentially be added to the recently viewed list.

Next, `RecentlyViewedTools` builds a "virtual" list of `recentlyViewedProduct` items that contains only those items in the customer's recently viewed list that are appropriate for the current site context. The items that are included are determined by the `RecentlyViewedTools.siteScope` property. The size of the list is determined by the `RecentlyViewedTools.recentlyViewedSize` property. The process for building virtual lists is described below:

- A `siteScope` of `current` dictates that there is one virtual list for each site. For example, if `siteScope=current` and `recentlyViewedSize=10`, Commerce Reference Store would have three virtual lists—one each for ATG Store US, ATG Store Germany, and ATG Home—and each list could contain up to 10 recently viewed products. A `recentlyViewedProduct` item is included in a virtual list if it has a matching site ID; for example, the ATG Store US list includes only those items whose `recentlyViewedProduct.siteId` value is `storeSiteUS`.

- When the `siteScope` value is a shareable type ID, there is one list for each sharing group that is defined by the shareable type ID. Any sites that are not in a sharing group also have their own list. For example, if `siteScope=atg.ShoppingCart` and `recentlyViewedSize=10`, Commerce Reference Store would have two virtual lists—one for ATG Store US and ATG Home, because they share a shopping cart, and a separate list for ATG Store Germany. Each list can contain up to 10 recently viewed products. A `recentlyViewedProduct` item is included in a virtual list if it has a matching site ID; for example, the ATG Store US/ATG Home list includes only those items whose `recentlyViewedProduct.siteId` value is either `storeSiteUS` or `homeSite`. Note that this is the configuration Commerce Reference Store uses, `siteScope=atg.ShoppingCart` and `recentlyViewedSize=10`.

- Finally, a `siteScope` of `all` dictates that there is one list for all sites. As with the other options, the size of this list is limited to the value specified by `recentlyViewedSize`. Any `recentlyViewedProduct` item may be included on this one recently viewed list.

After building the virtual list for the current site context, `RecentlyViewedTools` determines if the new `recentlyViewedProduct` has a site ID that matches one of the IDs associated with the virtual list. If so, `RecentlyViewedTools` adds the `recentlyViewedProduct` to the list, assuming the list limit has not been reached yet. If the virtual list is full, `RecentlyViewedTools` removes one or more of the oldest `recentlyViewedProduct` items to make room for the new item before adding it to the beginning of the list.

Note: `RecentlyViewedTools` would only need to remove more than one item from a recently viewed list to make room for a new item if the `recentlyViewedSize` value had been changed to a lower value, for example, prior to a server restart.

Filtering Products in the Recently Viewed List

Commerce Reference Store includes functionality that allows you to filter the list of recently viewed products so that you display only those products that are appropriate for the customer's site context in the Recently Viewed panel. In a multisite environment, any time you retrieve a collection of recently viewed products by referring to the customer's `recentlyViewedProducts` profile property, you get back an unfiltered list that may contain products from multiple sites. To limit recently viewed product display to only those products that are appropriate for the site context, you must filter out any products affiliated with sites that are outside of the current site context.

Filtering Components

Two components facilitate the filtering of recently viewed products:

- The `/atg/store/droplet/RecentlyViewedFilterDroplet` calls the `RecentlyViewedFilter` component to filter the recently viewed products and then renders the filtered results.
- The `/atg/registry/CollectionFilters/RecentlyViewedFilter` component filters collections of recently viewed products. It returns only those products that are appropriate for the site context.

The `RecentlyViewedFilterDroplet` is a globally-scoped instance of the class `atg.service.collections.filter.droplet.CollectionFilter` designed to render a filtered set of recently viewed products. It has the properties shown in the table below:

Property	**Description**
`consultCache` and `updateCache`	Properties required for enabling the caching of filtered collections. See the *ATG Personalization Programming Guide* for more information.
`extraParameterNames`	A comma-separated list that identifies the additional parameters that a JSP can specify as input parameters to `RecentlyViewedFilterDroplet`. `RecentlyViewedFilterDroplet` subsequently passes these parameters, as a map, to the `RecentlyViewedFilter`. There are two extra parameters and both are optional: `exclude`: Either a string that represents a single product ID, or a list of product IDs or `product` repository items. Any product specified in this parameter is excluded from the filtered list. `size`: An `int` that specifies the maximum number of products the `RecentlyViewedFilter` should return. This value overrides the `RecentlyViewedFilter.maxProductsToDisplay` property.
`filter`	A reference to the `RecentlyViewedFilter` component.

`RecentlyViewedFilter` is a globally-scoped component of class `atg.projects.store.collections.filter.RecentlyViewedFilter`. This class extends the generic collection filtering class atg.service.collections.filter.CachedCollectionFilter by overriding the generateFilteredCollection() method to take site scope into consideration when filtering recently viewed products. The generateFilterCollection() method also filters the recently viewed products according to any size or exclude parameters that have been passed in through the RecentlyViewedFilterDroplet. The RecentlyViewedFilter component has the properties shown in the table below:

Property	Description
catalogTools	A reference to the utility component /atg/commerce/catalog/CatalogTools, used for working with catalogs.
maxProductsToDisplay	An int that specifies the maximum number of products the RecentlyViewedFilter should return. This value may be overridden by an extraParameter called size that may be passed to RecentlyViewedFilter through the RecentlyViewedFilterDroplet. Commerce Reference Store sets this value to 5.
profileTools	A reference to the utility component /atg/userprofiling/ProfileTools, used for working with user profiles.
recentlyViewedTools	A reference to the utility component RecentlyViewedTools.
siteGroupManager	A reference to the utility component /atg/multisite/SiteGroupManager, used for working with sites in a multi-site environment. RecentlyViewedFilter uses the SiteGroupManager component to check if a shareableType ID is registered and to get a list of sites that are sharing a shareableType object with the current site.

Filtering Process

RecentlyViewedFilter iterates over each product in the customer's recentlyViewedProducts property, starting with the most recently viewed product first. The first filtering operation determines if the product being evaluated has been explicitly excluded. If so, RecentlyViewedFilter omits the product from the filtered collection. Next, RecentlyViewedFilter takes into account the RecentlyViewedTools.siteScope property to do additional filtering:

- If the siteScope setting is current, RecentlyViewedFilter compares the siteId for the recently viewed product being evaluated to the current site ID. If they match, the product is added to the filtered collection.
- If the siteScope setting is a shareable type ID, RecentlyViewedFilter retrieves a list of IDs for any sites that are in a sharing group with the current site, as defined by the shareable type ID (for example, the list of sites that share a shopping cart with the current site). RecentlyViewedFilter then compares the siteId for the recently viewed product being evaluated to the site IDs it retrieved. If there is a match, the product is added to the filtered collection.

  Note that this is the type of filtering Commerce Reference Store uses. Commerce Reference Store sets the RecentlyViewedTools.siteScope setting to atg.ShoppingCart, so the customer will see recently viewed products from the current site and any sites that share a shopping cart with the current site. For example, a customer viewing ATG Store US would see recently viewed products from ATG Store US and ATG Home, but not from ATG Store Germany.

- If the `siteScope` setting is `all`, all recently viewed products are added to the filtered collection, regardless of site ID (in other words, no products are filtered out).

`RecentlyViewedFilter` performs two additional tests when determining if a recently viewed product should be included in the filtered collection. If the site the product was originally viewed on is disabled or unavailable, the product is omitted. Also, if the product no longer exists, it is omitted from the filtered collection. When filtering is complete, the filtered collection is returned to `RecentlyViewedFilterDroplet` for rendering.

Note: With a `RecentlyViewedTools.siteScope` setting of `all`, all recently viewed products are always returned and no filtering occurs, so there is no need to use a recently viewed product filter unless you are using a `siteScope` other than `all`.

Rendering the Recently Viewed Products Panel

The `store.war/browse/gadgets/recentlyViewed.jsp` gadget is responsible for rendering the Recently Viewed panel at the bottom of the product detail page. For each recently viewed product, `recentlyViewed.jsp` renders the product name, image, pricing, and site the product was viewed on. Also, each recently viewed product in the panel is rendered as a link that takes the customer to the product detail pages for the individual products.

To do its job, `recentlyViewed.jsp` uses the `RecentlyViewedFilterDroplet` to generate a filtered collection of recently viewed products that are appropriate for the current site context. As described in Filtering Products in the Recently Viewed List (page 61), the `RecentlyViewedFilterDroplet` can take an `exclude` parameter so that any specified products are excluded from the recently viewed products list. Commerce Reference Store's implementation of the product detail page takes advantage of this mechanism to prevent the currently displayed product from also appearing in the recently viewed products list. To do this, the product detail page specifies the current product's ID in an `exclude` page parameter when it calls `recentlyViewed.jsp`, and then `recentlyViewed.jsp` passes the excluded product to the `RecentlyViewedFilterDroplet`.

The `RecentlyViewedDroplet` calls `RecentlyViewedFilter` to do the actual filtering. `RecentlyViewedFilter` returns the filtered list of recently viewed products in descending timestamp order so that the most recently viewed product is returned first, the second most recently viewed product is returned second, and so on. A `ForEach` servlet bean is used to iterate over the filtered list in this same order, so that the most recently viewed product is rendered farthest to the left and the least recently viewed product is rendered farthest to the right. The `ForEach` servlet bean's `output` open parameter includes the generic gadget, `store.war/global/gadgets/promotedProductRenderer.jsp`, to render each recently viewed product, including its associated information and a link to its product detail page.

Note that the recently viewed product's site ID is passed to the `promotedProductRenderer.jsp` gadget. The `promotedProductRenderer.jsp` gadget then forwards this site ID on to the gadget that renders the site indicator portion of the recently viewed product information, namely `store.war/global/gadgets/siteIndicator.jsp`. This is necessary because a product may be associated with more than one site and, if a site ID is not provided, `siteIndicator.jsp` will pick a site ID based on its own internal logic. In the case of a recently viewed product, we want the site indicator to represent the site the product was viewed on, so we must explicitly pass the recently viewed product's site ID to `siteIndicator.jsp`.

On a final note, if `RecentlyViewedFilter` returns an empty list, the `ForEach` servlet bean terminates itself and the Recently Viewed panel is not rendered.

Additional Support for Recently Viewed Products

In addition to the components described earlier in this section, both the `atg.projects.store.profile.StoreProfileFormHandler` class and the `/atg/userprofiling/PropertyManager` component include functionality to support the recently viewed products feature.

StoreProfileFormHandler

When an anonymous customer browses the Commerce Reference Store web site, a list of recently viewed products is maintained for the customer. If the customer subsequently logs in, the list needs to be persisted. In order to support this functionality, the Commerce Reference Store class `atg.projects.store.profile.StoreProfileFormHandler`, which is an extension of `atg.commerce.profile.CommerceProfileFormHandler`, includes a property called `recentlyViewedProducts`. This property holds a list of the anonymous customer's recently viewed products so that it can be processed after the customer logs in. Along with the `recentlyViewedProducts` property, `StoreProfileFormHandler` overrides two of the `CommerceProfileFormHandler` methods and adds a new method:

- `preLoginUser()`: Commerce Reference Store modifies this method so that, if the customer is anonymous, any recently viewed products are stored in the `StoreProfileFormHandler.recentlyViewedProducts` property.
- `postLoginUser()`: Commerce Reference Store modifies this method so that, if the anonymous customer successfully logs in, the `postLoginAddRecentlyViewed()` is invoked.
- `postLoginAddRecentlyViewed()`: This method updates the logged in customer's `recentlyViewedProducts` profile property with the products stored in the `StoreProfileFormHandler.recentlyViewedProducts` property.

PropertyManager

The `/atg/userprofiling/PropertyManager` component holds a map of repository item property names and the hard-coded references to those properties used in the Commerce Reference Store code base. This map is useful in cases where a repository item property name changes. In such situations, you only need to update the repository item property name in the `PropertyManager` component, instead of having to update all of the references to the property name in the Java code. The recently viewed properties that are added to the `PropertyManager` component include:

```
# recentlyViewedProduct names
recentlyViewedProductItemDescriptorName=recentlyViewedProduct
productPropertyName=product
siteIdPropertyName=siteId
timeStampPropertyName=timestamp
```

The `PropertyManager` component is of class `atg.projects.store.profile.StorePropertyManager`, which is an extension of the `atg.commerce.profile.CommercePropertyManager` class.

5 Promotions

Commerce Reference Store includes a number of promotions that demonstrate how to implement the most common types of promotions found on commerce web sites. This chapter discusses these promotions and it includes the following topics:

Evaluating Promotions

Commerce Reference Store evaluates each promotion every time it does a pricing operation. The evaluation process is as follows: item discounts are evaluated first, followed by order discounts, and finally shipping discounts.

For each promotion type (item/order/shipping), priority settings further refine the order in which the promotions are evaluated. Promotions with a higher priority are evaluated before promotions with a lower priority. Promotions that are of the same type and have the same priority have no guaranteed sequence, so the order in which they are evaluated is undefined. For example, Commerce Reference Store includes three shipping discount promotions:

- TENSHIP Coupon – Free Shipping, priority 1
- 10LIEFERUNG Coupon – Free Shipping, priority 1
- ATG Home – Free Shipping, priority 2

When evaluating these promotions, Commerce Reference Store randomly evaluates one of the priority 1 promotions, then it evaluates the other priority 1 promotion, and finally it evaluates the priority 2 promotion. The following table shows the promotions defined for Commerce Reference Store, grouped by promotion type and priority:

Promotion	Description	Type	Priority
Registered Users – Save 15% on Home Accents	Products in the Home Accents category are discounted 15% for registered users. If the product is available on multiple sites, it will still qualify for the discount, regardless of the site it was added from. This promotion is granted by the `GrantPromotions` scenario to registered customers when an order qualifies. It is available on all sites. **Note:** See Scenarios That Grant Promotions (page 54) for more information on the `GrantPromotions` scenario.	Item discount	1
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. This promotion is automatically granted to all customers. It is available on the ATG Store US and ATG Home sites.	Item discount	1
Get 2 Free Gifts With Any Woman's Jacket	Customers can pick two free gifts from the Handbags and Wallets category when their shopping cart contains an item from the Jackets category. This promotion is automatically granted to all customers and it is available on all sites. **Note:** For more information on Commerce Reference Store's implementation of gift with purchase promotions, see Gift with Purchase Promotions (page 72).	Item discount	1
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. This promotion is granted by the `GrantPromotions` scenario when a registered female customer that is part of the `ThirtySomethings` user segment has an order that qualifies. It is available on all sites and may be granted to a customer multiple times. **Note:** See Scenarios That Grant Promotions (page 54) for more information on the `GrantPromotions` scenario.	Item discount	2
Save 20% On Any ATG Home Item Excluding Home Accents	A discount of 20% is applied to the highest priced SKU in the customer's shopping cart that comes from the ATG Home site. If a SKU belongs to the Home Accents category, it is excluded from the promotion and will not be discounted. This promotion is automatically granted to all customers. It is available on the ATG Store US and ATG Home sites.	Item discount	2

Promotion	Description	Type	Priority
Free Scarf With All Orders Over $200	Customers receive a free Cashmere Scarf when their order total is greater than $200. This promotion is automatically granted to all customers and it is available on all sites. **Note:** For more information on Commerce Reference Store's implementation of gift with purchase promotions, see Gift with Purchase Promotions (page 72).	Item discount	3
10% Off Order	A discount of 10% is applied to the entire order. This promotion is granted by the `giveAbandonedOrderPromotion` scenario when a customer has an abandoned order. It is available on all sites. **Note:** See Scenarios That Grant Promotions (page 54) for more information on the `giveAbandonedOrderPromotion` scenario.	Order discount	1
10LIEFERUNG Coupon – 10% Off Order	A discount of 10% is applied to the entire order. This promotion is granted to any customer who enters the 10LIEFERUNG coupon code on the ATG Store Germany site.	Order discount	1
TENSHIP Coupon – 10% Off Order	A discount of 10% is applied to the entire order. This promotion is granted to any customer who enters the TENSHIP coupon code on the ATG Store US or ATG Home site.	Order discount	1
$10 Off Orders Over $100	A discount of $10 is applied on orders over $100. This promotion is automatically granted to all customers on all sites.	Order discount	2
TENSHIP Coupon – Free Shipping	A shipping discount for free shipping is applied to the entire order. This promotion is granted to any customer who enters the TENSHIP coupon code on the ATG Store US or ATG Home site.	Shipping discount	1
10LIEFERUNG Coupon – Free Shipping	A shipping discount for free shipping is applied to the entire order. This promotion is granted to any customer who enters the 10LIEFERUNG coupon code on the ATG Store Germany site.	Shipping discount	1
ATG Home – Free Shipping	When customers check out on ATG Home, they receive free shipping. This promotion raises customer awareness of the ATG Home site. This promotion is automatically granted to all customers and it is available on the ATG Home site only.	Shipping discount	2

Promotion Availability

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 table in the previous section, Evaluating Promotions (page 65), lists the granting mechanism for each Commerce Reference Store promotion. 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*.

Stacking Rules

Promotion type and priority dictate the order in which Commerce evaluates promotions during a pricing operation; however, Commerce must also take into account stacking rule definitions as it evaluates each individual promotion. Stacking rules allow you to create groups of promotions and then define which groups may be used together and which may not. For example, Commerce Reference Store has a Gift With Purchase (GWP) Promotions stacking rule that includes the Free Scarf With All Orders Over $200 and Get 2 Free Gifts With Any Woman's Jacket promotions. It also has an Apparel Promotions stacking rule that includes the 25% Off Women's Apparel promotion. These two stacking rules are defined as mutually exclusive in the stacking rule definitions. This means that, if a GWP Promotions promotion has already been applied to an order, an Apparel Promotions promotion may not be applied to that order and vice versa.

You can create overrides to stacking rule exclusions within the definitions of individual promotions. For example, the Hubbard Chair Tiered Price Break promotion explicitly excludes the $10 Off Orders Over $100 promotion. This means that, regardless of how the stacking rules interact with each other, if the Hubbard Chair Tiered Price Break promotion has been applied to an order, the $10 Off Orders Over $100 promotion may not be applied. In another example, the $10 Off Orders Over $100 promotion explicitly includes the ATG Home – Free Shipping promotion. Again, regardless of the fact that order promotions and shipping promotions cannot be applied together due to stacking rule exclusions, if the $10 Off Orders Over $100 promotion is applied to an order, the ATG Home – Free Shipping promotion may also be applied if the order qualifies for it.

Finally, stacking rules can be used to define limits on promotions that have been grouped together. For example, Commerce Reference Store defines a limit of 1 on the GWP Promotions stacking rule. This means that if one of the free gift promotions has been applied to an order, the other free gift promotion may not be.

The following table shows the stacking rule definitions for Commerce Reference Store along with any explicit overrides defined for individual promotions:

Stacking Rule Name and Limit	Groups These Promotions	Exclusions Due to Stacking Rules	Explicitly Excluded Promotions	Explicitly Included Promotions
Item Promotions (No limit)	Registered Users – Save 15% On Home Accents	N/A	N/A	N/A
	Hubbard Chair Tiered Price Break	N/A	$10 Off Orders Over $100	N/A

Stacking Rule Name and Limit	Groups These Promotions	Exclusions Due to Stacking Rules	Explicitly Excluded Promotions	Explicitly Included Promotions
	Save 20% On Any ATG Home Item Excluding Home Accents	N/A	N/A	N/A
Apparel Promotions (No limit)	25% Off Women's Apparel	Excludes GWP Promotions	N/A	N/A
GWP Promotions (Limit = 1)	Get 2 Free Gifts With Any Woman's Jacket	Excludes Apparel Promotions	N/A	N/A
	Free Scarf With All Orders Over $200	Excludes Apparel Promotions	N/A	N/A
Order Promotions (Limit = 1)	10LIEFERUNG Coupon – 10% Off Order	Excludes Shipping Promotions	N/A	10LIEFERUNG Coupon – Free Shipping
	TENSHIP Coupon – 10% Off Order	Excludes Shipping Promotions	N/A	TENSHIP Coupon – Free Shipping
	$10 Off Orders Over $100	Excludes Shipping Promotions	N/A	ATG Home – Free Shipping
Shipping Promotions (No limit)	TENSHIP Coupon – Free Shipping	Excludes Order Promotions	N/A	N/A
	10LIEFERUNG Coupon – Free Shipping	Excludes Order Promotions	N/A	N/A
	ATG Home – Free Shipping	Excludes Order Promotions	N/A	N/A

Because stacking rules can cause promotions to be excluded, the stacking rules implementation uses the Commerce messaging API to inform the customer if a promotion has not been applied and why. In such instances, Commerce Reference Store displays a message on the shopping cart page that says "Where did my promotion go?" Clicking this link displays a separate window with the promotion exclusion details. The link on the shopping cart page is included in the `store.war/checkout/gadgets/checkoutOrderSummary.jsp` gadget and the "Where did my promotion go" page is rendered by `store.war/global/gadgets/promotionDetailsPopup.jsp`. For more details on the messaging API, see the *ATG Commerce Programming Guide*.

Notes:

- Stacking rules are designed to always be mutually exclusive, so in Commerce Reference Store, the Order Promotions stacking rule excludes the Shipping Promotions stacking rule and the Shipping Promotions stacking rule excludes the Order Promotions stacking rule. The reality, however, is that order promotions are evaluated and applied to an order before shipping promotions are, so the Shipping Promotions stacking rule will never actually prevent an order promotion from being applied. The same would be true of any order promotion stacking rules that exclude item promotions because item promotions are evaluated before order promotions are.

- If a promotion has been excluded due to a stacking rule exclusion or an explicit exclusion, any associated promotion upsell will also not be evaluated.
- A promotion can be associated with only one stacking rule at any given time.
- For more details on creating stacking rules, see the *ATG Merchandising Guide for Business Users*.

Notifying a Customer of a Promotion

Commerce Reference Store must notify customers to make them aware of the promotion and encourage purchasing behavior. Notification is done through a number of different mechanisms, described below.

Promotion Messaging

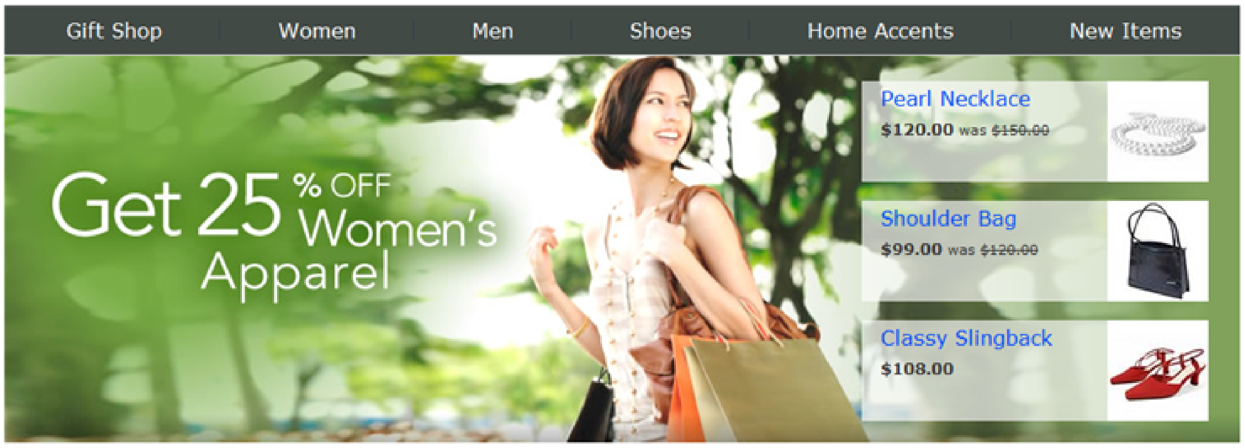
Messages appear in both the rich cart and the shopping cart when a customer's cart has been modified as the result of a promotion. These messages are handled through the Commerce messaging functionality. For more information, see Promotion Messages (page 81) later in this chapter.

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 (page 54) for more details.

Notification on the Home Page Splash Image

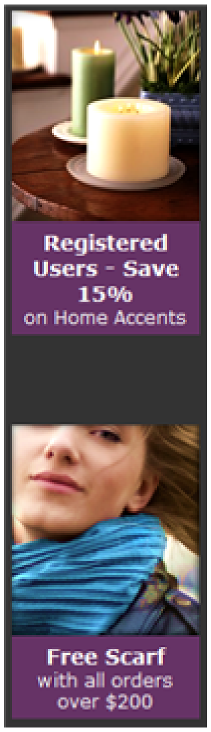
The home page splash image is used to notify customers about some of the Commerce Reference Store promotions. For example, the home page splash image for registered customers who are female and a member of the `ThirtySomethings` user segment notifies customers about the 25% Off Women's Apparel promotion, as shown in the illustration below:



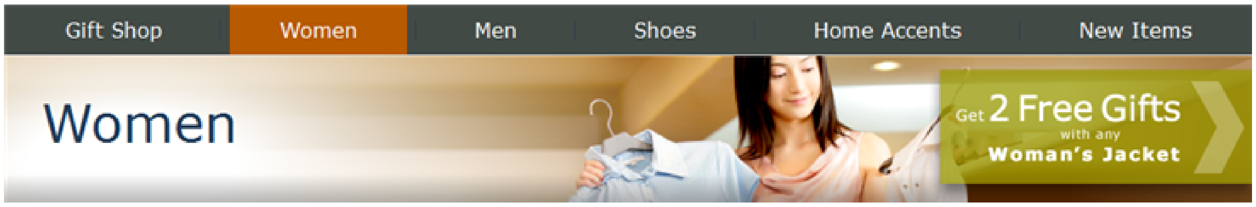
Notification on the Category and Default Browse Pages

The category and default browse pages are Endeca-driven pages. As such, their contents are managed and rendered via cartridges.

Both pages use `PromotionalContent-ATGSlot` cartridges to render two promotional content items in the lower portion of the left hand rail that notify the customer about various promotions. See the PromotionalContent-ATGSlot, PromotionalContent-ATGTargeter, ProductSpotlight-ATGSlot, and ProductSpotlight-ATGTargeter (page 126) section in the *Using Cartridges* (page 115) chapter for more details on this cartridge type.



The category page also incorporates a `HeaderBanner-ATGCategory` cartridge that shows a promotional content item in the right hand side of the header banner. This cartridge retrieves the category name, the hero image, and the promotional content item to display from the category repository. For more details on the `HeaderBanner-ATGCategory` cartridge, see the HeaderBanner-ATGCategory (page 123) section in the *Using Cartridges* (page 115) 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.

Gift with Purchase Promotions

Commerce Reference Store includes two gift with purchase promotions, Get 2 Free Gifts With Any Woman's Jacket and Free Scarf With All Orders Over $200. Gift with purchase promotions have a significant impact on the Commerce Reference Store UI and behavior, so they warrant some additional discussion.

Commerce Reference Store Gift with Purchase Promotions

This section provides more details about the gift with purchase promotions themselves.

Get 2 Free Gifts with Any Woman's Jacket

With this promotion, customers can pick two free gifts from the Handbags and Wallets category when their shopping cart contains an item from the Jackets category. This promotion is automatically granted to all customers and it is available on all sites.

The two free gifts are automatically added when a customer's cart contains a woman's jacket. Placeholders are rendered in both the rich cart and on the shopping cart page to indicate to the customer that she needs to pick her free gifts. If any items from the Handbags and Wallets category already exist in the cart, they are re-priced to free. If the cart ceases to qualify, these items are re-priced back to their original prices (this functionality is controlled by setting the Auto Remove option in the promotion's definition to No). Any gift placeholders are also removed if the cart ceases to qualify.

Free Scarf with All Orders Over $200

With this promotion, customers receive a free Cashmere Scarf when their order total is greater than $200. This promotion is automatically granted to all customers and it is available on all sites.

The Cashmere Scarf is automatically added to the cart when a customer's order total is $200 or greater. The Cashmere Scarf is a non-navigable item (that is, a customer cannot navigate to it in the catalog), so it can never exist in the cart unless it is added as a result of the promotion. If the cart ceases to qualify for the promotion, the Cashmere Scarf is automatically removed (auto-remove is set to Yes).

Note: See Catalog Structure (page 85) for more information on Commerce Reference Store's non-navigable items.

Extensions to Commerce Gift with Purchase Classes

Commerce Reference Store extends several core Commerce gift with purchase components.

StoreGiftWithPurchaseFormHandler

In Commerce Reference Store, the `/atg/commerce/promotion/GiftWithPurchaseFormHandler` component is of class `atg.projects.store.promotion.StoreGiftWithPurchaseFormHandler`. This class extends the core Commerce `atg.commerce.promotion.GiftWithPurchaseFormHandler` class to handle gift selection requests through AJAX. This allows the application to perform better when validation errors occur because the whole page does not need to be reloaded, only the error messages do. `StoreGiftWithPurchaseFormHandler` extends the `handleMakeGiftSelection()` method to redirect to different success and error URLs, depending on whether or not the request was made through AJAX. `StoreGiftWithPurchaseFormHandler` also extends the `preMakeGiftSelection()` method so that the application displays different error messages for each SKU attribute (color, size, wood finish).

StoreGWPManager

In Commerce Reference Store, the `/atg/commerce/promotion/GWPManager` component is of class `atg.projects.store.promotion.StoreGWPManager`. This class extends the core Commerce `atg.commerce.promotion.GWPManager` class. `StoreGWPManager` modifies the `processAutoAdd()` method in order to check the inventory level of an item before adding it as a gift to the order. Also, this class adds two helper methods:

- `getQuantityToBeSelected()` determines the gift quantity that is still available for selection in the order.
- `getItemGiftQuantity()` determines gift quantity in the current `CommerceItem`.

StoreGiftWithPurchaseSelectionsDroplet

In Commerce Reference Store, the `/atg/commerce/promotion/GiftWithPurchaseSelectionsDroplet` component is of class `atg.projects.store.droplet.StoreGiftWithPurchaseSelectionsDroplet`. This class extends the core Commerce `atg.commerce.promotion.GiftWithPurchaseSelectionsDroplet` class by adding an additional `quantityToBeSelected` output parameter that represents the gift quantity in the order that has not been selected yet.

Gift with Purchase and the Rich Cart

The rich cart displays any free gifts currently in the cart. These gifts may be any of the following:

- An item that was chosen through the free gift picker
- An item that was in the cart and then re-priced to free when the cart qualified for the promotion
- An item that was automatically added to the cart by the promotion (such as the Cashmere Scarf)

- A placeholder for a free gift that has not been chosen yet

The JSP that renders the cart items portion of the rich cart is located in `store.war/cart/json/cartItems.jsp`. This JSP makes two calls to the `/atg/commerce/promotion/GiftwithPurchaseSelectionsDroplet`. The first call is used when rendering gifts that are not placeholders (the first three cases in the list above). The second call is used when rendering gift placeholders.

To render gifts that are not placeholders, `cartItems.jsp` iterates through all the `CommerceItem` objects in the current cart, passing each `CommerceItem` to the `GiftWithPurchaseSelectionsDroplet`. The droplet determines if there is any free gift quantity within each `CommerceItem` by looking for a gift with purchase marker. Because a `CommerceItem` can contain a quantity greater than one, it is possible that a portion of the `CommerceItem` is free while the rest of it is not. For example, a `CommerceItem` may contain three Cowgirl Bags, with two of the bags qualifying as free gifts while the third bag is priced as a regular item.

Note: See the *ATG Commerce Programming Guide* for information about gift with purchase markers.

If `GiftWithPurchaseSelectionsDroplet` determines that any portion of the current `CommerceItem` is a gift, `cartItems.jsp` calculates the portion of the `CommerceItem` that is free. `cartItems.jsp` does this using the `GiftWithPurchaseSelection` bean that the `GiftWithPurchaseSelectionsDroplet` generates for the free portion of the item. This bean has properties that represent the quantity of the `CommerceItem` that:

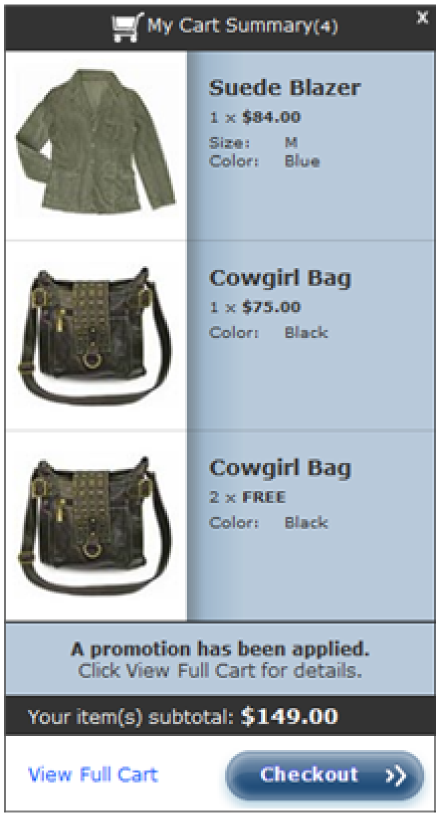
- Was automatically added to the cart (`automaticQuantity`)
- Is the result of re-pricing items that already existed in the cart (`targetedQuantity`)
- Was explicitly selected by the customer in the free gift picker (`selectedQuantity`)

Adding these quantities together results in a total gift quantity for this `CommerceItem`. Subtracting the total gift quantity from the `CommerceItem`'s total quantity provides the quantity of the `CommerceItem` that is not a gift.

Next, `cartItems.jsp` includes the `cartItem.jsp` page to render the non-gift portion of the `CommerceItem` on a line in the rich cart. For this include, the gift with purchase `promotionID` is passed in the `excludeModelId` parameter. Passing the `promotionID` in this way tells `cartItem.jsp` to not render any price information for items associated with the gift with purchase promotion. This avoids rendering a zero price on this line of the rich cart. (Price information is only rendered for items that are not free.)

The next call to the `cartItem.jsp` page renders the gift portion of the `CommerceItem` on another line in the rich cart. For this call, the `isGift` parameter is passed so that `cartItem.jsp` will render the price as FREE in the rich cart.

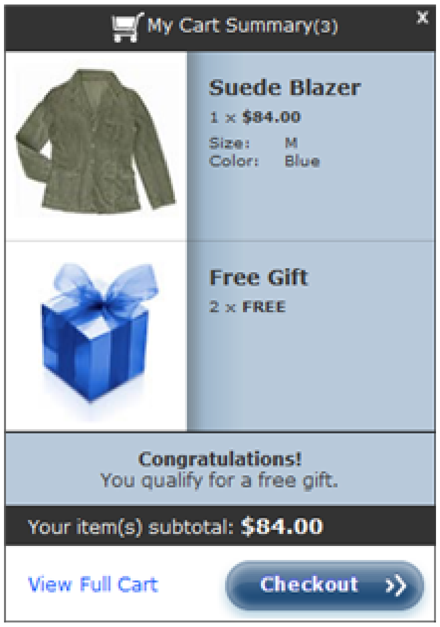
For an example of how the splitting of a `CommerceItem` into free and non-free items looks, consider the following illustration:



To create the situation depicted in this illustration, the following happened:

- The customer added the Suede Blazer to her cart, qualifying her for two free gifts from the Handbags & Wallets category.
- The customer chose the Cowgirl Bag for both of her free gifts and, because she wants to give them as gifts to three friends, she added a third bag to her cart. This third bag is priced normally.
- The rich cart shows two line items, one for the regularly priced Cowgirl Bag and a second for the two free Cowgirl Bags. The first line item considers only the price for the Cowgirl Bag that is not free. The second line item marks the Cowgirl Bag as FREE instead of rendering the zero price.

The final task `cartItems.jsp` must do is render any gift placeholders. To do this, `cartItems.jsp` calls the `GiftWithPurchaseSelectionsDroplet` again. This call passes the current order without a specific `CommerceItem`, so the `GiftWithPurchaseSelectionsDroplet` returns `GiftWithPurchaseSelection` beans for any gifts in the entire order. `cartItems.jsp` further refines what `GiftWithPurchaseSelectionsDroplet` returns by passing in the `onlyOutputAvailableSelections` parameter. This parameter tells the `GiftWithPurchaseSelectionsDroplet` to return only those `GiftWithPurchaseSelection` beans whose `quantityAvailableForSelection` is not zero. These are the beans for which gift placeholders need to be rendered. Each bean is then passed to the `store.war/cart/json/giftPlaceholder.jsp` page for rendering. This illustration shows what a gift placeholder looks like in the rich cart. Note that the two free gifts are rendered as one line in the rich cart (as opposed to the shopping cart, which renders a separate line for each free gift, as described in Gift with Purchase and the Shopping Cart (page 76)).



Gift with Purchase and the Shopping Cart

`store.war/cart/gadgets/cartItems.jsp` is responsible for rendering the line items that appear on the shopping cart page for the cart contents. This JSP page works similarly to the `store.war/cart/json/cartItems.jsp` page in that it iterates over each `CommerceItem` in the cart, passing the current `CommerceItem` to the `/atg/commerce/promotion/GiftwithPurchaseSelectionsDroplet`, to determine if the item has a gift with purchase marker. If there isn't one, the current `CommerceItem` is not a gift and the `GiftWithPurchaseSelectionsDroplet`'s `empty` open parameter is rendered. This is where the code for rendering standard `CommerceItem` objects resides.

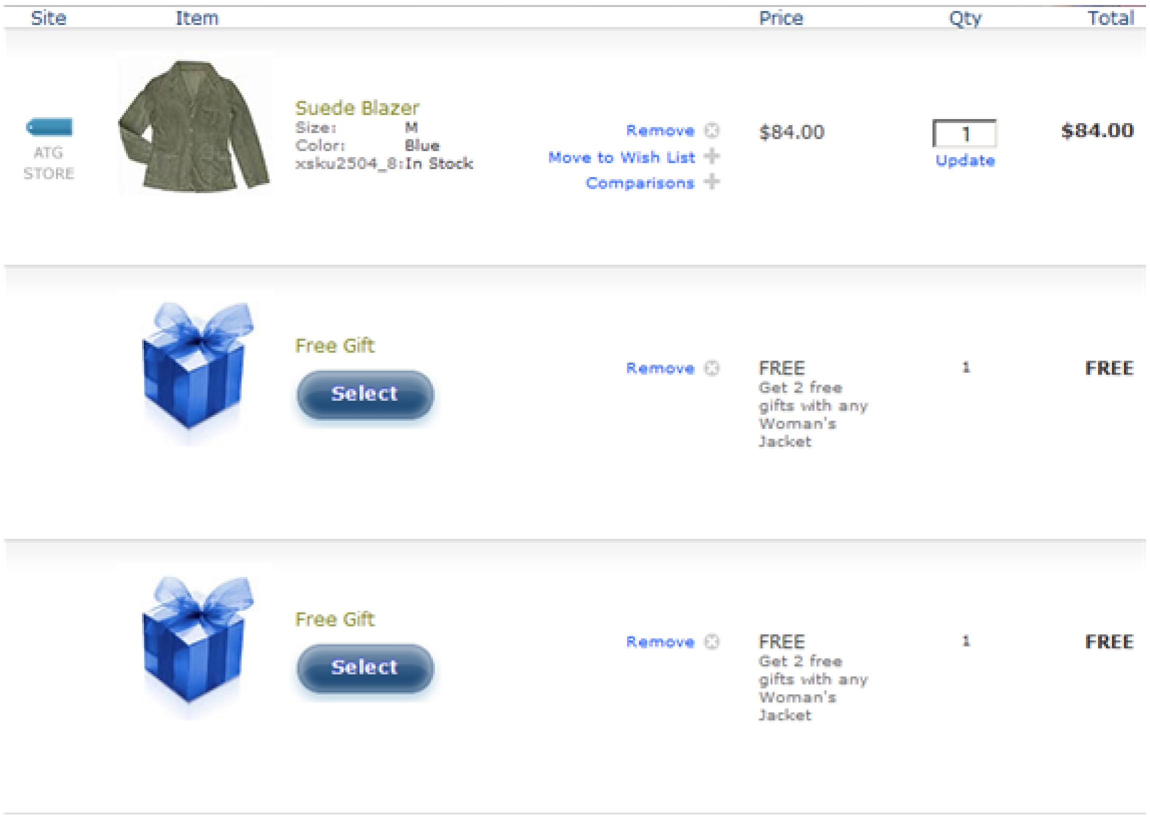
If the current `CommerceItem` does have a gift with purchase marker, the `GiftWithPurchaseSelectionsDroplet` produces a `GiftWithPurchaseSelection` bean for it. `store.war/cart/gadgets/cartItems.jsp` uses the same logic as `store.war/cart/json/cartItems.jsp` to calculate the portion of the `CommerceItem` that is free versus the portion that is not. It adds the `automaticQuantity`, `targetedQuantity`, and `selectedQuantity` values provided by the bean to arrive at a total gift quantity, then it subtracts the total gift quantity from the overall `CommerceItem` quantity. `cartItems.jsp` stores this value in a `missedQty` variable and renders a line in the shopping cart for the `CommerceItem`, using the `missedQty` value in the Quantity column. This line represents the portion of the `CommerceItem` that is not a gift.

To handle the `automaticQuantity` portion of the `CommerceItem`, `cartItems.jsp` uses a `forEach` loop with a `begin` attribute of 1 and an `end` attribute of `automaticQuantity`. Each pass through the loop calls the `store.war/cart/gadget/giftItem.jsp` once to render an individual line in the shopping cart. In this way, `cartItems.jsp` renders, for example, three lines in the shopping cart for an `automaticQuantity` of three. `cartItems.jsp` repeats this process for the `targetedQuantity` and `selectedQuantity` portions of the `CommerceItem`. These lines together represent the gift portion of the `CommerceItem` that has been selected (i.e., that is tied to a SKU and is not represented by a gift placeholder).

To render the gift placeholders, `cartItems.jsp` includes the `store.war/cart/gadgets/giftPlaceholders.jsp` page. `giftPlaceholders.jsp` calls the `GiftWithPurchaseSelectionsDroplet` again, but this time it passes in the order without the current `CommerceItem`, so the `GiftWithPurchaseSelectionsDroplet` returns `GiftWithPurchaseSelection` beans for any gifts in the entire order. `giftPlaceholders.jsp` also passes in the `onlyOutputAvailableSelections` parameter so that the `GiftWithPurchaseSelectionsDroplet` returns only those `GiftWithPurchaseSelection` beans whose `quantityAvailableForSelection` is not zero. These are the beans for which gift placeholders need to be rendered.

Next, `giftPlaceholders.jsp` iterates through each `GiftWithPurchaseSelection` bean and retrieves the quantity of gifts still to be selected from the bean's `quantityAvailableForSelection` property. `giftPlaceholders.jsp` uses this value as the `end` attribute in a `forEach` loop that renders an individual gift placeholder, on its own line in the shopping cart, for each outstanding gift selection. For example, two gift placeholders are rendered on two separate lines for a `GiftWithPurchaseSelection` bean whose `quantityAvailableForSelection` value is two.

The following illustration shows a customer's shopping cart after she has added the suede blazer and qualified for two free gifts. Notice that each gift placeholder resides on its own line:



Site	Item			Price	Qty	Total
ATG STORE		Suede Blazer Size: M Color: Blue xsku2504_8:In Stock	Remove Move to Wish List Comparisons	$84.00	1 Update	$84.00
		Free Gift Select	Remove	FREE Get 2 free gifts with any Woman's Jacket	1	FREE
		Free Gift Select	Remove	FREE Get 2 free gifts with any Woman's Jacket	1	FREE

In Commerce Reference Store, each gift placeholder in the shopping cart represents a single gift selection. Note that this behavior is different from how the rich cart behaves, where gift placeholders are aggregated into a single line item to conserve on space. The shopping cart, however, is where the user interface for choosing, editing, or removing free gifts resides, so the gift placeholders themselves must be tied to a single gift selection. The gift selection interface is described in the following section, Choosing Gifts in Commerce Reference Store (page 77).

Choosing Gifts in Commerce Reference Store

The `store.war/cart/gadgets/giftSelection.jsp` page renders the gift selection options for each gift item. This page use the `atg.commerce.promotion.GiftWithPurchaseSelectionChoicesDroplet` to obtain a list of `GiftWithPurchaseSelectionChoice` beans associated with the current gift. Each

GiftWithPurchaseSelectionChoice bean contains a product, and its associated SKUs, that is available for selection for the current gift.

Depending on how a gift is configured in the Business Control Center, it may have a giftType that consists of a single product or multiple products. The sku and product gift types include a single product, hence one GiftWithPurchaseSelectionChoice bean is returned for this gift type. The category, skuContentGroup, and productContentGroup gift types include multiple products, so multiple GiftWithPurchaseSelectionChoice beans are returned.

The list of GiftWithPurchaseSelectionChoice beans is passed to the /atg/commerce/collections/filter/droplet/InventoryFilterDroplet. This droplet iterates over each bean, using the Commerce Reference Store-specific filter /atg/registry/CollectionFilters/GWPSelectionInventoryFilter, and checks if the SKUs are available in inventory. Any GiftWithPurchaseSelectionChoice bean that has no SKUs available in inventory is filtered out.

The gift selection page iterates over the filtered GiftWithPurchaseSelectionChoice beans and renders a row for each bean that displays the product and SKU choices for that bean. The giftSelection.jsp gadget renders a JavaScript-based picker similar to what is used on the product detail pages.

Error Messages on the Gift Selection Page

The handling of error messages on the gift selection page is done through AJAX. An AJAX request is sent to the server when the customer chooses a gift in the popup page. If an error occurs during the request processing, JavaScript retrieves the error messages from a JSON response and injects them into the HTML structure of the popup. To implement this solution, the store.war/cart/gadgets/giftSelection.jsp gadget specifies the store.war/cart/json/giftErrors.jsp JSP page as the error URL to use when the gift selection has been made via an AJAX request:

```
<dsp:input bean="GiftWithPurchaseFormHandler.ajaxMakeGiftSelectionErrorURL"
           type="hidden" value="${contextroot}/cart/json/giftErrors.jsp"/>
```

The giftErrors.jsp page extracts the errors from the /atg/commerce/promotion/GiftWithPurchaseFormHandler.formExceptions property using the /atg/dynamo/droplet/ErrorMessageForEach droplet and then creates a JSON object that contains an array of the errors that occurred during request processing. The handleJSON() JavaScript function from store.war/javascript/picker.js processes the received JSON object and checks whether it contains any errors. If so, the errors are injected into a special reserved <div> container inside the gift selection popup. Using this method to render error messages improves the performance of the gift popup page because only a portion of the page is re-rendered when errors occur, not the entire page.

Removing Gifts from the Shopping Cart

When removing gifts from the shopping cart, Commerce Reference Store has to take into account several cases:

- Removing gifts that are only a portion of a CommerceItem's total quantity (in other words, the CommerceItem includes both a gift quantity and a non-gift quantity of the given item)
- Removing gift placeholders from the shopping cart page
- Removing all unselected gifts from the shopping cart so that the customer can proceed to checkout

Removing Gifts that are Only a Portion of a Commerce Item's Total Quantity

When a CommerceItem is split across multiple lines in the shopping cart, some of which represent gifts and some of which represent regularly priced items, care must be taken to modify the appropriate quantity in the

CommerceItem and not remove the entire CommerceItem from the cart when a Remove link is clicked. The logic for handling this situation is contained in the store.war/cart/gadgets/itemListingButtons.jsp page. This page tests each line item in the cart to determine if the item is a gift item or a regularly priced item that is part of a CommerceItem that also contains gifts.

For a gift item, the itemListingButtons.jsp page includes the following code to render the Remove link:

```
<dsp:input iclass="atg_store_textButton atg_store_actionDelete" type="submit"
                       name="remove_gwp_${nameSuffix}"
                       bean="CartFormHandler.removeItemGwp"
                       value="${deleteButtonTitle}"
                       submitvalue="${currentItemId}"/>
```

When a customer clicks the Remove link associated with this code, the following happens:

- The gift ID that is specified in the submitvalue attribute is stored in the /atg/store/order/purchase/CartFormHandler.removeItemGwp property.
- The atg.projects.store.order.purchase.StoreCartFormHandler.handleRemoveItemGwp() method is called. This method is Commerce Reference Store-specific and it calls the StoreCartFormHandler.handleUpdate() method to update each line item in the cart.

  During the process of updating the cart, the StoreCartFormHandler.getQuantity() method gets called. This method is a Commerce Reference Store extension that checks for the presence of the removeItemGwp property when calculating the item's quantity. If removeItemGwp is set, getQuantity() decreases the overall quantity of the CommerceItem by one. The overall quantity is reduced by one because this is a gift item and gifts are separated out in the shopping cart so that there is one gift per line item; therefore, removing a gift line item constitutes a reduction in quantity of one.

  Note that Commerce Reference Store's implementation of the getQuantity() method decreases the total quantity of the CommerceItem but it does not change the item's gift with purchase markers. The effect of this change is to decrease the total quantity while leaving the gift quantity constant. For example, if the initial total quantity is three and the gift quantity is one, the CommerceItem contains two regularly priced items and one free item. After the getQuantity() decrease, the total quantity is two, the gift quantity is still one, and the regularly priced quantity is also one. So, effectively, the regularly priced quantity of the CommerceItem is reduced first, before the free quantity is reduced.

  Note: See the *ATG Commerce Programming Guide* for more information on gift with purchase markers.

For a regularly priced item that is part of a CommerceItem that also contains gifts, the itemListingButtons.jsp page includes the following code:

```
<dsp:input type="hidden" converter="map"
           bean="CartFormHandler.itemNonGwpQuantities"
           value="${currentItemId}=${missedQty}"/>

<dsp:input iclass="atg_store_textButton atg_store_actionDelete" type="submit"
           name="remove_non_gwp_${nameSuffix}"
           bean="CartFormHandler.removeItemNonGwp"
           value="${deleteButtonTitle}"
           submitvalue="${currentItemId}"/>
```

The first input statement stores a map with the current item's ID and the quantity of that item that is regularly priced in the CartFormHandler.itemNonGwpQuantities property. As each item in the cart is iterated over, this map grows to hold the regularly priced quantities for all items in the cart that have a combination of free and regularly priced quantities.

Note: The cart item iteration code is contained in `store.war/cart/gadgets/cartItems.jsp`, which is the page that calls `itemListingButtons.jsp`.

The second statement renders the Remove link. When a customer clicks the Remove link, the following happens:

- The item ID that is specified in the `submitvalue` attribute is stored in the `/atg/store/order/purchase/CartFormHandler.removeItemNonGwp` property.
- The `atg.projects.store.order.purchase.StoreCartFormHandler.handleRemoveItemNonGwp()` method is called. This method is Commerce Reference Store-specific and it calls the `StoreCartFormHandler.handleUpdate()` method to update each line item in the cart.

  During the process of updating the cart, the `StoreCartFormHandler.getQuantity()` method gets called. This method is a Commerce Reference Store extension that checks for the presence of the `removeItemNonGwp` property when calculating the item's quantity. If `removeItemNonGwp` is set, `getQuantity()` decreases the overall quantity of the `CommerceItem` by the amount specified for that item in the `itemNonGwpQuantities` map.

Removing Gift Placeholders

The Remove link for gift placeholders is rendered by the `store.war/cart/gadgets/giftPlaceholders.jsp` page. This page includes the following code:

```
<dsp:input type="hidden" bean="CartFormHandler.gwpPlaceholderPromotionIds"
           paramvalue="selection.promotionId"/>

<dsp:input type="hidden" bean="CartFormHandler.gwpPlaceholderHashCodes"
           paramvalue="selection.giftHashCode"/>

<dsp:input iclass="atg_store_textButton atg_store_actionDelete"
           type="submit" submitvalue="${gwpRowCount}"
           bean="CartFormHandler.removeGwpPlaceholderFromOrder"
           value="${removeText}"/>
```

The first `input` statement stores a list of the promotion IDs associated with all the gift placeholders, in the order they appear in the cart, in the `/atg/store/order/purchase/CartFormHandler.gwpPlaceholderPromotionIds` property. The second `input` statement stores a list of all the gift placeholder hash codes, in the order they appear in the cart, in the `/atg/store/order/purchase/CartFormHandler.gwpPlaceholderHashCodes` property.

The third `input` statement renders the Remove link. When a customer clicks the Remove link, the following happens:

- The row number of the gift placeholder, which is specified in the `submitvalue` attribute, gets stored in the `/atg/store/order/purchase/CartFormHandler.removeGwpPlaceholderFromOrder` property.
- The `atg.projects.store.order.purchase.StoreCartFormHandler.handleRemoveGwpPlaceholderFromOrder()` method is called. This handler method is Commerce Reference Store-specific. It gets the row number of the submitted placeholder and uses that information to retrieve the corresponding promotion ID and hash code from the `gwpPlaceholderPromotionIds` and `gwpPlaceholderHashCodes` lists. These two values identify the promotion in this particular order whose selectable gift quantity should be reduced. Next, the

`atg.commerce.promotion.GWPManager.updateSelectableQuantity()` method is called and the gift quantity available for selection for this promotion is reduced by one.

Removing All Unselected Gifts

When customers attempt to checkout with unselected free gifts still in the shopping cart, they are notified and asked whether they want to return to the shopping cart page to choose their free gifts or continue with checkout. If the shopper chooses to continue with checkout, all unselected free gifts must be removed from the shopping cart. The code that manages this task resides in the `store.war/cart/giftNotSelected.jsp` page.

If the customer chooses to proceed with checkout, the `giftNotSelected.jsp` page sets the `/atg/commerce/promotion/GiftWithPurchaseFormHandler.removeAllSelectableQuantitySuccessURL` property with an appropriate URL. The URL that gets sets depends on whether login is required for checkout or the customer is using express checkout. Next, the `atg.commerce.promotion.GiftWithPurchaseFormHandler.handleRemoveAllSelectableQuantity` method is called to remove all unselected free gifts from the order and redirect the customer to the URL stored in `removeAllSelectableQuantitySuccessURL`.

Promotion Messages

Both the rich cart and the shopping cart display messages that alert the customer when he has qualified for a promotion and when he no longer qualifies for a promotion. Out of the box, Commerce includes a messaging API that creates appropriate messages for gift with purchase promotions and stacking rule exclusions. Commerce Reference Store also implements messaging for other types of promotions. This section describes both.

Commerce Message Slots

Commerce defines and populates two slots for messages:

- `/atg/commerce/pricing/CleanBeforePricingSlot`: This slot contains stacking rule exclusion messages, for example, "The Free Scarf with all orders over $200 promotion cannot be applied as the limit on the number of promotions of this type has been reached." This slot is emptied and repopulated with appropriate messages during each pricing operation.
- `/atg/commerce/pricing/NoCleanBeforePricingSlot`: This slot contains gift with purchase messages, for example, "Congratulations! You qualify for a free gift." Unlike `CleanBeforePricingSlot`, messages in this slot are not removed until they are retrieved by the application. This means that this slot can contain messages from multiple pricing operations and those messages can be duplicates of each other or conflict with each other.

Enabling Commerce Messaging

By default, Commerce promotion messaging is disabled. To use it, Commerce Reference Store must set the `/atg/commerce/pricing/PricingTools.sendUserMessages` property to true, as shown below:

```
$class=atg.projects.store.pricing.StorePricingTools

# Enable application messaging
```

```
sendUserMessages=true
```

Rich Cart Messaging

The rich cart displays both gift with purchase messages and other promotion-related messages.

Retrieving Gift with Purchase Messages for the Rich Cart

The `store.war/cart/json/cartMessages.jsp` uses a `TargetingArray` droplet to retrieve all the messages Commerce has placed in the `NoCleanBeforePricingSlot`. The `cartMessages.jsp` page then filters those messages using the `/atg/store/collections/filter/PromotionMessagesFilter`. This filter is a Commerce Reference Store-specific filter of class `atg.projects.store.collections.filter.RemoveDuplicateMessagesFilter`. The filter removes any duplicate gift with purchase messages. It also checks for gift with purchase promotion “failed” messages and, if it finds one, all previous “success” messages for the same promotion are ignored.

Retrieving Other Promotion-Related Messages for the Rich Cart

To retrieve messages related to other types of promotions, Commerce Reference Store includes a custom droplet, `/atg/store/droplet/CartMessages`. This droplet checks the current order for active promotions and, if it finds any, it returns the message “A promotion has been applied. Click View Full Cart for details.” The `CartMessages` droplet is of class `atg.projects.store.droplet.CartMessagesDropet`. It has an `order` input parameter that specifies the order to return messages for; if not specified, the current order is used. It has a `message` output parameter that contains an `atg.web.messaging.UserMessage` bean. This bean has several properties: `identifier`, `type`, `priority` and `summary`. The rich cart displays both the title for the promotion message and the summary. The title is retrieved from the Commerce Reference Store Web application resource bundle using the message’s identifier as a key. (This is the “A promotion has been applied” text.) The summary is retrieved from the message’s `summary` property (the “Click View Full Cart for details” text).

Rendering the Rich Cart Messages

The `cartMessages.jsp` page uses the two methods described above to retrieve all the promotion messages for the order. It passes each message it retrieves to the `store.war/cart/json/cartMessage.jsp` page for translation into a JSON object. `cartMessages.jsp` stores these JSON objects in a parent JSON object called the `messagesContainer`. The `store.war/javascript/widget/RichCartSummary.js` widget’s `setAllCartData` function, in turn, uses the contents of the `messagesContainer` to render all of the messages at the bottom of the rich cart.

Shopping Cart Messaging

Gift with purchase messages appear in an orange banner at the top of the shopping cart page. The `store.war/cart/gadgets/cartMessages.jsp` is responsible for rendering this portion of the shopping cart. This page uses a similar technique to `store.war/cart/json/cartMessages.jsp`. Specifically, it uses a `TargetingArray` droplet to retrieve the messages Commerce has placed in the `NoCleanBeforePricingSlot`, and it filters out duplicates or conflicting messages using the `PromotionMessagesFilter` before rendering the messages.

Stacking Rule Messages

Stacking rules can result in promotions being excluded from use in a customer’s shopping cart. In order to make it clear to the customer why a promotion has been excluded, the Order Summary portion of the shopping cart page has a “Where did my promotion go?” link. The `store.war/checkout/gadgets/checkoutOrderSummary.jsp` page renders this link only if there are stacking rule exclusion messages in the `CleanBeforePricingSlot`. If there are messages in the slot, `checkoutOrderSummary.jsp` renders a link to the `store.war/global/gadgets/promotionDetailsPopup.jsp` page. Using a `TargetingArray` droplet, this popup page retrieves and renders all of the messages in the `CleanBeforePricingSlot`. Note that, unlike gift with purchase messages, we do not need to filter out any stacking rule exclusion messages from the `CleanBeforePricingSlot` because this slot is cleared out before each pricing operation. This ensures that only up-to-date messages related to the latest pricing operation exist in the slot.

6 Product Catalog

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

Catalog Structure

This section provides an overview of the Commerce Reference Store catalog structure.

Commerce Reference Store Master Catalogs

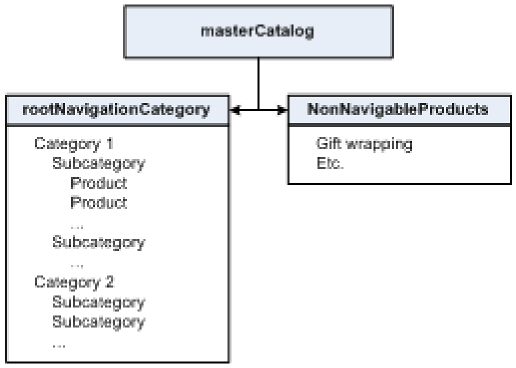
Commerce Reference Store has two catalog items in the Catalog repository, `masterCatalog` and `homeStoreCatalog`. ATG Store US and ATG Store Germany both use `masterCatalog` as their default catalog. ATG Home uses `homeStoreCatalog` as its default catalog.

`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).

The `rootCategory` should not be visible to a Web site visitor, although its child categories and products should be. Neither the `NonNavigableProducts` category nor its children should be visible to a Web site visitor. For this reason, both categories are explicitly excluded from the index. For details on these exclusions, see the Configuring the Indexing Components chapter in the *Endeca Integration Guide*.

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 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 Content Administration environment, `CatalogCompletionService` keeps site IDs updated for catalog folders, catalogs, and categories as you work with your catalogs in 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 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 Commerce Reference Store, 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 Commerce Reference Store. This chapter covers those that are.

Extensions to the Catalog Item Descriptor

Commerce Reference Store 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.

Note: The `rootCategory` should not be visible to a Web site visitor, although its child categories and products should be. Neither the `NonNavigableProducts` category nor its children should be visible to a Web site visitor. For this reason, both categories are explicitly excluded from the index. For details on these exclusions, see the Configuring the Indexing Components chapter in the *Endeca Integration Guide*.

Extensions to the Category Item Descriptor

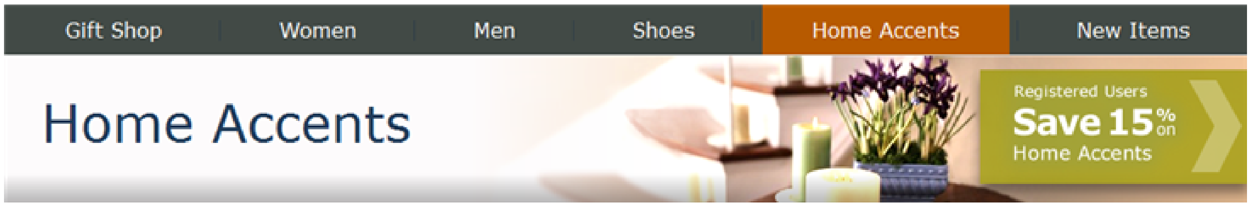
This section describes additions and modifications to the `category` item descriptor.

feature and myFeature

The `feature` and `myFeature` properties work together to ensure that a promotional image, such as the one below for the Registered Users – Save 15% on Home Accents promotion, 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 Commerce Reference Store JSP code.

The `feature` property is used on the Endeca-driven category page. This page includes a `HeaderBanner-ATGCategory` cartridge that renders all parts of the header banner, including the category title, a hero image, and the promotional content item specified by the category's `feature` property, as shown below:

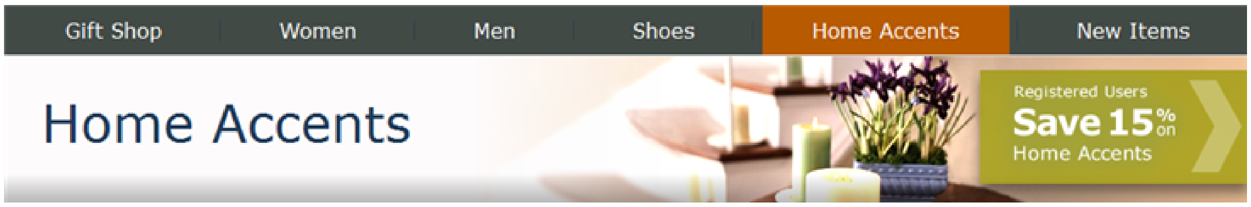


The handler for the `HeaderBanner-ATGCategory` cartridge is responsible for retrieving all of the header content from the catalog repository and passing that content to the cartridge's JSP renderer. See the HeaderBanner-ATGCategory (page 123) section in the *Using Cartridges* (page 115) chapter for more details on the `HeaderBanner-ATGCategory` cartridge.

Note: `promotionalContent` is a Commerce Reference Store-specific item type. For detailed information, see Custom Item Descriptors in the Catalog Repository (page 94).

heroImage

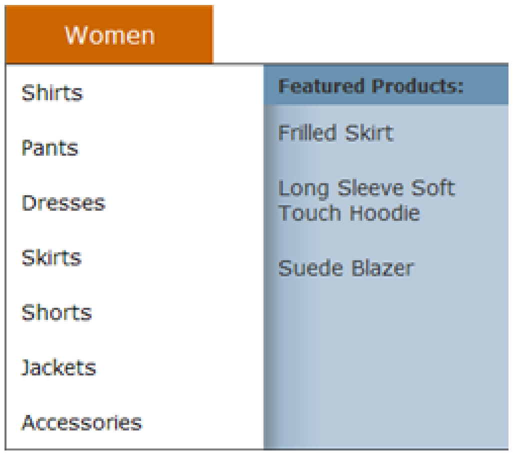
`heroImage` stores a splash image for the category in the form of a `media` item. The `heroImage` property is used on the Endeca-driven category page. This page includes a `HeaderBanner-ATGCategory` cartridge that renders all parts of the header banner, including the category title, a hero image, and the category's promotional content item, as shown below:



The handler for the `HeaderBanner-ATGCategory` cartridge is responsible for retrieving all of the header content from the catalog repository and passing that content to the cartridge's JSP renderer. See the HeaderBanner-ATGCategory (page 123) section in the *Using Cartridges* (page 115) chapter for more details on the `HeaderBanner-ATGCategory` cartridge.

relatedProducts

`relatedProducts` stores a list of items of type `product` that represent a set of products in a category. The `store.war/navigation/gadgets/catalog.jsp` gadget uses `relatedProducts` to populate the list of featured products that appear in the menu bar. When you click a category in the menu bar, you see a menu with a list of subcategories down the left and a list of related products down the right.



Commerce Reference Store also ships with a cartridge that uses the `relatedProducts` property to render a set of related product images in the main content area of the category page. This cartridge, however, is not used out of the box. To get more information for it, refer to the ProductSpotlight-ATGCategoryRelatedProducts (page 128) section of the *Using Cartridges* (page 115) chapter.

Extensions to the media-internal-text Item Descriptor

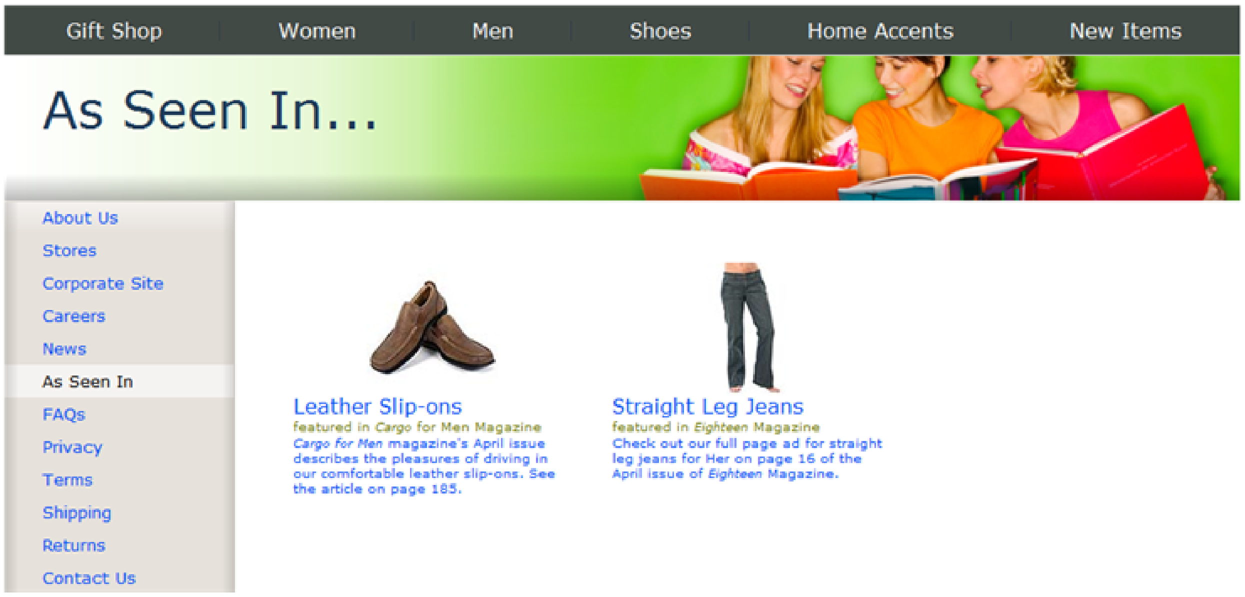
Commerce Reference Store extends the `media-internal-text` item descriptor with changes that allow these items to be edited in the ATG Control Center. These changes don't affect how items of this type function in ATG Store.

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. For example, in the illustration of the As Seen In page below, the `asSeenIn` property for the Straight Leg Jeans product is "Check out our full page ad for straight leg jeans for her on page 16 of the April issue of Eighteen Magazine."



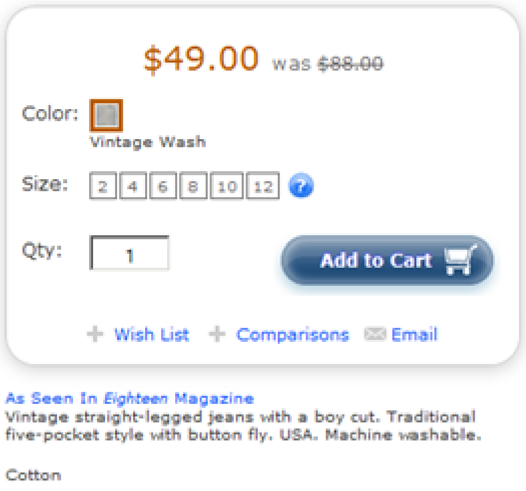
Leather Slip-ons
featured in *Cargo* for Men Magazine
Cargo for Men magazine's April issue describes the pleasures of driving in our comfortable leather slip-ons. See the article on page 185.

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.

The `asSeenIn` property is of type `asSeenIn`.

The `store.war/browse/asSeenIn.jsp` page is responsible for populating the As Seen In page in ATG Store. `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. After filtering, `asSeenIn.jsp` calls `store.war/browse/asSeenInElement.jsp` which uses the contents of the `asSeenIn` property when rendering the individual as-seen-in items.

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 Commerce Reference Store-specific item type. For detailed information, see Custom Item Descriptors in the Catalog Repository (page 94).

brand

`brand` provides a manufacturer's brand name for the product.

briefDescription

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

emailAFriendEnabled

`emailAFriendEnabled` is a Boolean property that controls whether an Email link appears on the product detail page, below the color/size picker, 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 dimension for guided navigation. See Guided Navigation (page 144) in the *Search Integration* (page 143) chapter for more details.

Note: `feature` is a Commerce Reference Store-specific item type. For detailed information, see Custom Item Descriptors in the Catalog Repository (page 94).

fullImage

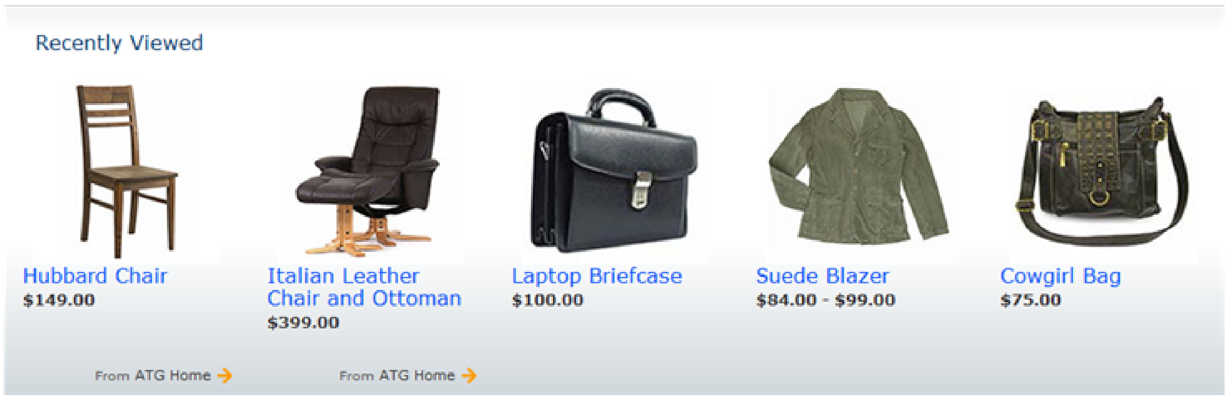
`fullImage` stores the largest version of a product image, in the form of a `media-external` item. The `store.war/browse/gadgets/productImage.jsp` gadget uses the `fullImage` property to render a larger image of the product when a customer clicks the View Larger link on the product detail page.

The image is displayed in a `dojox.image.Lightbox`. An example of the html rendered is:

```
<a class="atg_store_largerImage" href="/crsdocroot/content/images/products/full/
APP_FrilledSkirt_full.jpg" dojotype="dojox.image.Lightbox" title="Frilled Skirt"
id="dojox_image_Lightbox_1" widgetid="dojox_image_Lightbox_1">View Larger &gt;</a>
```

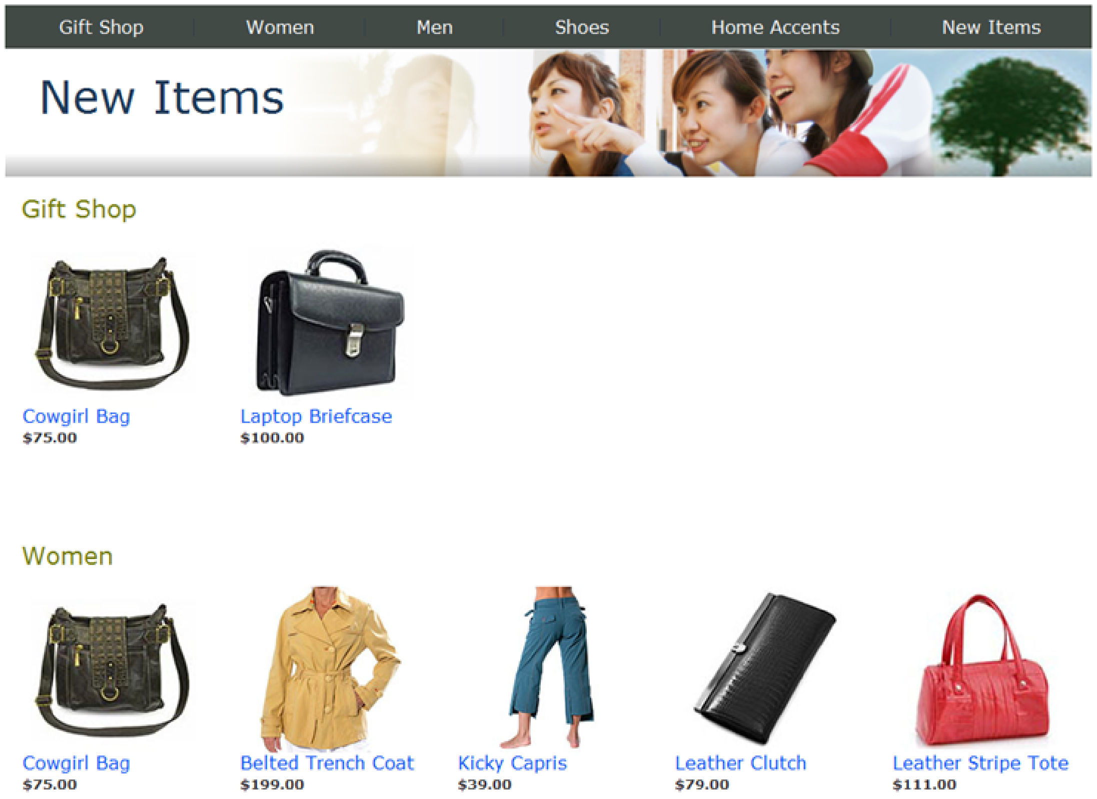
mediumImage

`mediumImage` stores a 150 x 150 pixel version of the product image, in the form of a `media-external` item. This version of the product image is displayed in a variety of locations such as the Recently Viewed panel of the product detail page.



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 Commerce Reference Store sample data. The `store.war/browse/newItems.jsp` page uses new to determine which products to render on the New Items page, a portion of which is shown below.



The best practice approach to identifying new products is to calculate which products are new based on date. Because demonstration data gets stale quickly, Commerce Reference Store cannot use this methodology, so `newItems.jsp` uses the manually defined `new` property instead when it builds the query for new products in the catalog repository. If the `new` property is not set to `true`, `newItems.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 or equal to the new product threshold, the product is considered new and rendered accordingly.

preorderable, preorderEndDate, and useInventoryForPreorder

Commerce Reference Store 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.

Commerce Reference Store includes two components that can be used to determine if a product is preorderable:

- `/atg/store/droplet/ProductDetailsDroplet`, which is an instance of `atg.projects.store.droplet.DefaultProductDetailsDroplet`. Commerce Reference Store uses this component when rendering the product detail page.
- `/atg/store/droplet/SkuAvailabilityLookup`, which is an instance of `atg.projects.store.droplet.SkuAvailabilityLookup`. Commerce Reference Store uses this component when rendering the gift list, wish list, and comparisons pages.

Both components call the `atg.projects.store.inventory.StoreInventoryManager.queryAvailabilityStatus()` method, which determines availability based on a product's properties and the inventory levels reported by the `/atg/commerce/inventory/InventoryManager` component. `StoreInventoryManager.queryAvailabilityStatus()` identifies 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.

For an example of `ProductDetailsDroplet` use, see `store.war/browse/productDetailMultiSkuContainer.jsp` and `store.war/browse/productDetailSingleSkuContainer.jsp`.

For an example of `SkuAvailabilityLookup` use, see `store.war/browse/gadgets/skuAvailabilityLookup.jsp` and `store.war/browse/gadgets/multiSkuAvailabilityLookup.jsp`.

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 Commerce Reference Store gadgets, considers both properties when determining if a shipping address is valid for a particular item:

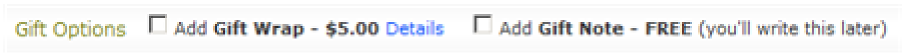
- If `shippableCountries` is null, all countries are shippable, except those listed in `nonShippableCountries`.
- If `shippableCountries` contains one or more countries, only those countries are shippable and `nonShippableCountries` is ignored.

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. The illustration below shows the gift wrap options the customer sees in her shopping cart.


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

margin

The `margin` property store sales margin data for each sku, for example, a margin of .12 indicates that 12% of the sku's price is profit. Note that, in Commerce Reference Store, this data is static, manually populated, and intended for demonstration purposes only. Typically, this property would be populated by a third-party system. The `ResultsList` cartridge included on the default browse page uses the `margin` property in its relevance ranking strategy.

taxStatus

The `taxStatus` property tags items with different product codes that indicate whether a product should be taxed. Commerce Reference Store 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 (page 94) for more information on the clothing and furniture SKU sub-types.

Custom Item Descriptors in the Catalog Repository

Commerce Reference Store adds the following item descriptors to the catalog repository:

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

This section describes the Commerce Reference Store-specific item types and their key properties. For a complete list of item type properties, use the Component Browser in the 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` and it has the properties described in the table below.

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 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, as described in the table below.

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 (page 28) in the *Pages and Navigation* (page 21) chapter for details. - As a dimension for guided navigation. See Guided Navigation (page 144) in the *Search Integration* (page 143) 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.
`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 (page 28) in the *Pages and Navigation* (page 21) chapter for details. - As a dimension for guided navigation. See Guided Navigation (page 144) in the *Search Integration* (page 143) 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, as described in the table below.

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 (page 28) in the *Pages and Navigation* (page 21) chapter for details. - As a dimension for guided navigation. See Guided Navigation (page 144) in the *Search Integration* (page 143) 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

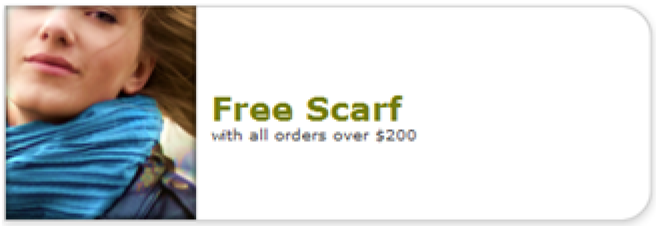
The `feature` item type stores information about a specific product feature, such as type of fabric or material. The `features` product property is list of items of type `feature` and it has the properties described in the table below.

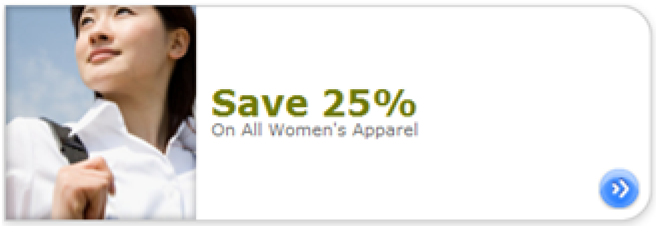
Property	Description
`description`	A short description of the `feature` item.
`displayName`	A name for the `feature` item that appears in the Business Control Center.
`featureName`	A string that identifies the `feature` item and may be used on a Web site to identify the item.
`id`	A unique ID for the `feature` item.
`largeImage` `smallImage`	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 Commerce Reference Store, a `promotionalContent` item is essentially an image, sometimes with supporting text. This illustration shows two examples of `promotionalContent` items:





The table below describes the `promotionalContent` properties:

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.

Property	Description

`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. Commerce Reference Store 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.
`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

A number of 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. The `store.war/browse/gadgets/recommendedProducts.jsp` page uses this data to render product suggestions on the right side of the product detail page.

Commerce Reference Store Pricing

This section describes how Commerce Reference Store 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.

Commerce Reference Store Price Lists

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

Commerce adds two properties to user profiles, `priceList` and `salePriceList`, which specify the price lists to use for a customer. 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*.

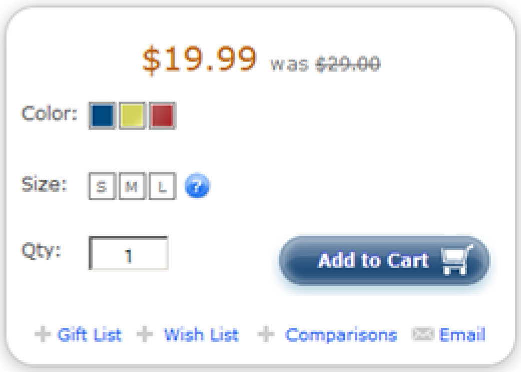
Rendering Prices on the Product Detail Page

Commerce Reference Store 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.

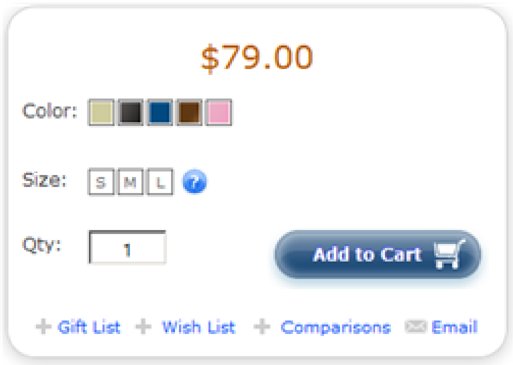
Rendering Prices for a Selected SKU

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:

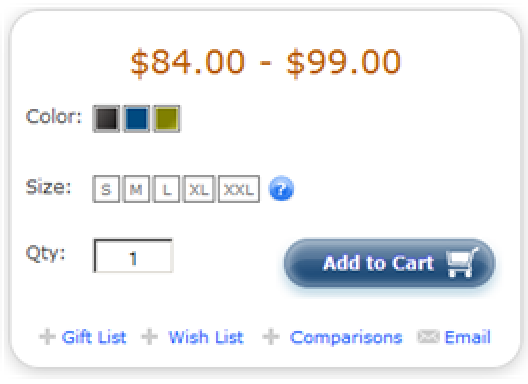


This illustration shows the pricing portion of the product detail page for a SKU that is not on sale:



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

7 Inventory Management

Commerce Reference Store uses the standard 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 Commerce by the `/atg/commerce/inventory/InventoryManager` component, which is of class `atg.commerce.inventory.RepositoryInventoryManager`. Commerce Reference Store changes this component's class to `atg.projects.store.inventory.StoreInventoryManager`, which is a Commerce Reference Store-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 Commerce Reference Store 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 Commerce Reference Store catalog, there is a corresponding `inventory` item in the inventory repository that tracks inventory data for the SKU. 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 Commerce Reference Store 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.

For most Commerce Reference Store SKUs, the values of these properties are all null, which means they use the default values set in the `InventoryManager.properties` file:

```
# These are the values used for the respective levels, if there
# is no value defined in the repository.  Remember, -1 implies an
# infinite amount.
#
defaultStockLevel=-1
defaultBackorderLevel=0
defaultPreorderLevel=0
```

Preorderable Customizations

Commerce Reference Store 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 Commerce Reference Store-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`.

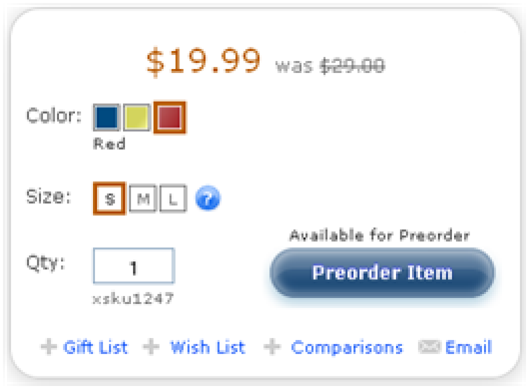
Caching Inventory Data

Commerce Reference Store sets the `cache-mode` attribute of the `inventory` item type to `simple`, with a timeout of 5 minutes. This ensures that the inventory information available on product detail pages is never more than a few minutes out of date, but avoids the frequent database lookups that would occur if caching were disabled. Caching can result in out-of-date inventory data at checkout time, but sites typically ensure they have up-to-date inventory information at checkout by making remote calls to an external inventory system that has real-time data.

If you build a site based on Commerce Reference Store, you may need to change the cache mode to reflect your site's needs. See the *ATG Repository Guide* for information about the available cache modes.

Inventory Information on Site Pages

Commerce Reference Store 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:



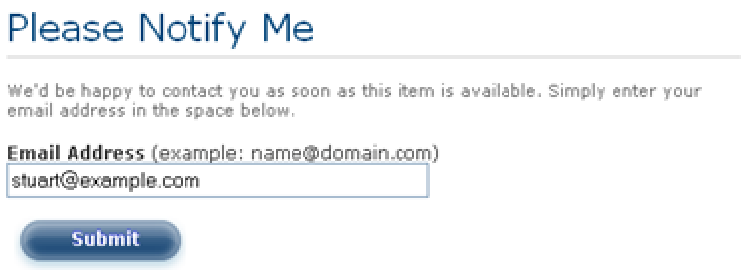
When a SKU is selected that is not in stock, the label on the Add to Cart button changes to reflect this. In the example above, the label has changed to Preorder Item because the selected SKU is preorderable.

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 Commerce Reference Store 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 A, *Sample Data* (page 193) 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)
stuart@example.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 Commerce Reference Store `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 Platform 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 A, *Sample Data* (page 193) for more information about out-of-stock products.

8 Order Handling

A key aspect of any Commerce implementation is the handling and processing of orders. This chapter discusses the order-handling process in 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 Commerce Reference Store-specific subclass of the standard Commerce `atg.commerce.order.purchase.CartModifierFormHandler` class.

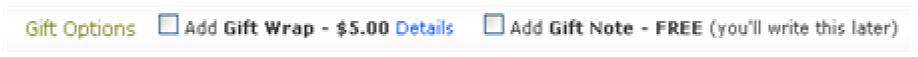
The shopping cart page includes a Checkout button that initiates the standard checkout process. The page also conditionally includes an Express Checkout button. See the Checkout Process (page 108) section for more information.

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, Commerce Reference Store 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 (page 23) in the *Pages and Navigation* (page 21) 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)

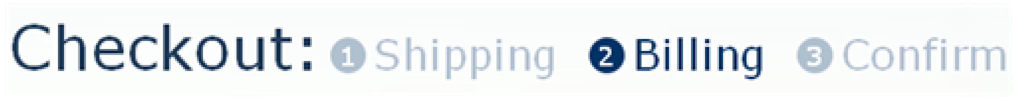
If the customer selects the gift wrap option, it applies to all of the SKUs in the order whose `giftWrapEligible` property is `true`. It is not possible to select the gift wrap option for some SKUs and not for others. The price of gift wrapping is the same regardless of the number of SKUs.

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 Commerce Reference Store-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. Commerce Reference Store changes the class of this component to `atg.projects.store.order.purchase.ShippingInfoFormHandler`, which is a Commerce Reference Store-specific subclass of the 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 Commerce Reference Store is based on the 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 *Promotions* (page 65) 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

Commerce Reference Store configures `ItemPricingEngine` to use precalculators that look up prices in price lists:

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

OrderPricingEngine

Commerce Reference Store uses `OrderPricingEngine` without any additional configuration.

ShippingPricingEngine

Commerce Reference Store 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 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 integrate your store with tax-calculating software, you can instead set `taxProcessor` to a component that makes calls to the tax-calculating software and processes the results.

Payment Options

Commerce Reference Store supports two payment options: store credits and a single credit card. (Commerce Reference Store 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

Commerce Reference Store uses the standard 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 Commerce Reference Store extensions, include the `Store.Fulfillment` module in your storefront EAR file.

Simulating Normal Order-Processing Behavior

When you run Commerce Reference Store 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 (for example, 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 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 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 Commerce `Fulfillment` module, but it is not enabled by default. You can enable it on a currently running Commerce 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/HardgoodShipper
```

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 Platform 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

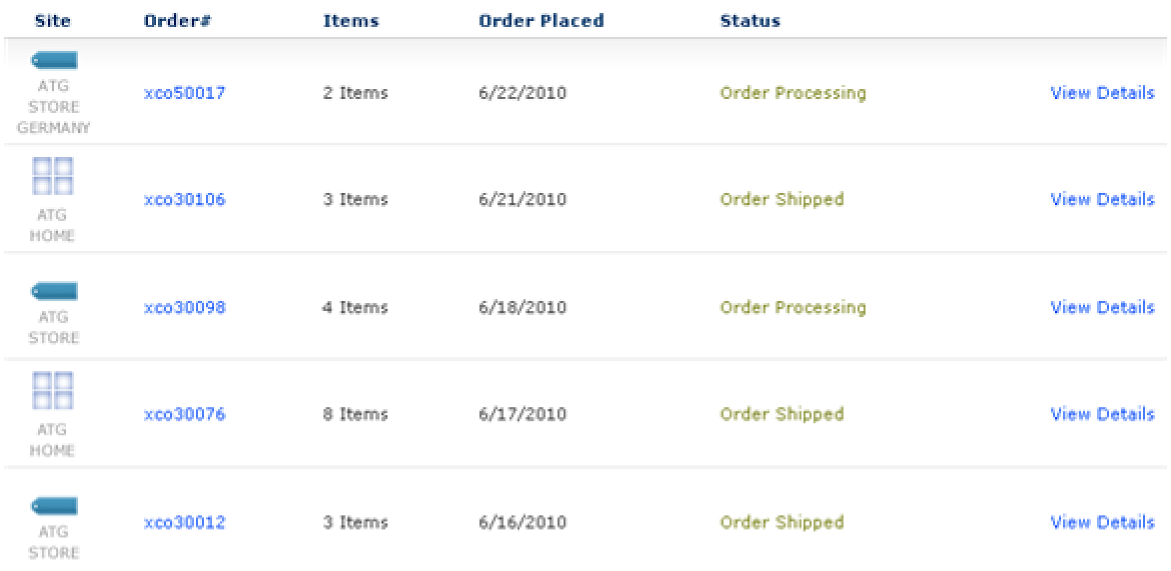
The Fulfillment system adjusts inventory when items are fulfilled. If you are using a different fulfillment system that does not adjust inventory, you can configure the Commerce Reference Store order manager to adjust inventory on checkout.

By default, the Commerce order manager, `/atg/commerce/order/OrderManager`, is of class `atg.commerce.order.SimpleOrderManager`. Commerce Reference Store changes the class of this component to `atg.projects.store.order.StoreOrderManager`, which is a Commerce Reference Store-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 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	6/22/2010	Order Processing	View Details
ATG HOME	xco30106	3 Items	6/21/2010	Order Shipped	View Details
ATG STORE	xco30098	4 Items	6/18/2010	Order Processing	View Details
ATG HOME	xco30076	8 Items	6/17/2010	Order Shipped	View Details
ATG STORE	xco30012	3 Items	6/16/2010	Order Shipped	View Details

The `store.war/myaccount/myOrders.jsp` page 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 A, *Sample Data* (page 193) for a list of sample customers who have orders with various statuses.

9 Using Cartridges

This chapter provides details on the cartridges that are included with the ATG Platform and Commerce Reference Store. It includes the following sections:

Creating Nucleus Components for Cartridges

To use cartridges, Oracle ATG Web Commerce must create Nucleus components for the cartridge handler classes and any classes that the handlers depend on. The `DAF.Endeca.Assembler` module includes Nucleus component configuration for existing Endeca cartridge handlers in the `/atg/endeca/assembler/cartridge/handler` and `/atg/endeca/assembler/cartridge/handler/config` Nucleus paths. The `CommerceReferenceStore.Store.Endeca.Assembler` module includes component configuration for Commerce Reference Store cartridges in those same Nucleus path locations.

Cartridges also rely on Experience Manager configuration as well as application rendering code. Because Experience Manager configuration and page rendering are both application-specific, the files that support all cartridges are included in the `CommerceReferenceStore` modules. See Experience Manager Configuration and JSP Renderers (page 115) for more information.

Experience Manager Configuration and JSP Renderers

Commerce Reference Store stores the XML files that configure Experience Manager in the `<ATG10dir>/CommerceReferenceStore/Store/Storefront/deploy/cartridge_templates` directory. This directory contains XML templates for all of the cartridges that are actively used in Commerce Reference Store. It also contains XML definitions for cartridges that are not used in Commerce Reference Store but could be useful for developers who base an application on Commerce Reference Store.

The JSP renderers for the cartridges that are used in Commerce Reference Store are located in the `store.war/cartridges` directory. The path to each JSP renderer follows this convention:

```
store.war/cartridges/cartridge-type/cartridge-type.jsp
```

For example, the path to the Breadcrumbs cartridge renderer is:

```
store.war/cartridges/Breadcrumbs/Breadcrumbs.jsp
```

Each renderer has its own directory, so that associated content (sub-pages, images, and so on) can be grouped together, making it possible for renderers to be packaged as stand-alone units. Providing a unique name for each renderer makes it easier to distinguish one file from another during various tasks such as editing.

Note: See *Retrieving Renderers* in the *Endeca Integration Guide* for details on how the correct renderer is retrieved for each cartridge.

Category Pages versus Browse Pages

Commerce Reference Store makes a distinction between category pages and browse pages. A category page is triggered when a query is composed of a single dimension value from the Category dimension, for example:

Category=Women

Browse pages are triggered whenever a shopper enters a search term or selects a refinement during the browse experience (in other words, for any condition *except* the category page condition). For example:

[search terms]

[search terms] + Category=Women

[search terms] + Color=Red

Category = Women + Color=Red

Advantages of the Two Page Types

By using a separate category page, an ATG merchandiser maintains the ability to manage the category hierarchy and the order of products in a category using the Business Control Center and Visual Merchandising. Meanwhile, the browse page takes advantage of Endeca's MDEX engine technology.

Commerce Reference Store includes a definition for a category page, called `Category Page`, and another for a browse page, called `Default Browse Page`. See Commerce Reference Store Cartridge Hierarchy (page 119) for more details on these two page definitions.

Behavior of Category Page-specific Cartridges

Some cartridges, such as `HeaderBanner-ATGCategory` and `ProductSpotlight-ATGCategoryRelatedProducts`, are specific to the category page. If you place these cartridges on a browse page, their content will not be rendered. For more information on how Commerce Reference Store determines if a request if for a category page, see Identifying Category Page Requests (page 121).

Experience Manager Templates in Commerce Reference Store

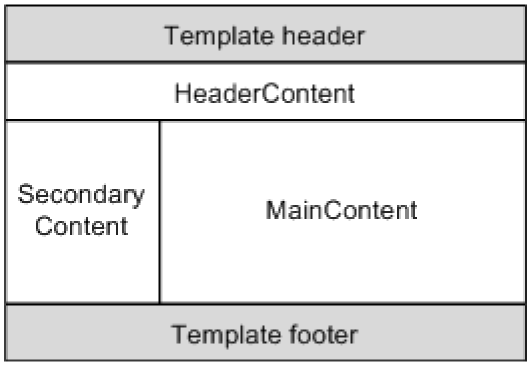
Commerce Reference Store includes two Experience Manager page templates, `OneColumnPage` and `TwoColumnPage`. `TwoColumnPage` is used out of the box while `OneColumnPage` is not. This section describes both.

TwoColumnPage Template

The `TwoColumnPage` template has three sections:

- `HeaderContent`: The `HeaderContent` section resides near the top of the page and extends across the entire page width. This section accepts cartridges of type `HeaderContent`.
- `SecondaryContent`: The `SecondaryContent` section provides a panel down the left hand side of the page and it accepts cartridges of type `SecondaryContent`.
- `MainContent`: The `MainContent` section appears to the right of the `SecondaryContent` section and takes up the bulk of the window. This section accepts cartridges of type `MainContent`.

The following illustration shows the layout of the `TwoColumnPage` template sections:



The XML for the `TwoColumnPage` is located in `<ATG10dir>/CommerceReferenceStore/Store/Storefront/deploy/cartridge_templates/Page-TwoColumnPage.xml`. This XML defines the content type of the `TwoColumnPage` as `Page` and its ID as `TwoColumnPage`.

The renderer that accompanies the `TwoColumnPage` template is located in `store.war/cartridges/TwoColumnPage/TwoColumnPage.jsp`. The renderer uses the `crs:pageContainer` tag to render the standard Commerce Reference Store page components such as header and footer. Then, the renderer takes each section defined in the `TwoColumnPage` template and renders the cartridges held in these areas using the `dsp:renderContentItem` tag. See the *Endeca Integration Guide* for more details on the `dsp:renderContentItem` tag.

```
<crs:pageContainer divId="atg_store_facetGlossaryIntro" contentClass="category"
```

```
                 index="false" follow="false"
                 bodyClass="category atg_store_leftCol">
  <jsp:body>
    <%-- Render the header --%>
    <c:if test="${not empty contentItem.header}">
      <c:forEach var="element" items="${contentItem.header}">
        <dsp:renderContentItem contentItem="${element}"/>
      </c:forEach>
    </c:if>
    <%-- Render the main content --%>
    <div>
      <c:forEach var="element" items="${contentItem.main}">
        <dsp:renderContentItem contentItem="${element}"/>
      </c:forEach>
    </div>
    <%-- Render the secondary content --%>
    <div>
      <c:forEach var="element" items="${contentItem.secondaryContent}">
        <dsp:renderContentItem contentItem="${element}"/>
      </c:forEach>
    </div>

  </jsp:body>
</crs:pageContainer>
```

OneColumnPage Template

The `OneColumnPage` template has two sections: `HeaderContent` and `MainContent`. These sections function very similarly to their counterparts in the `TwoColumnPage` template in that the `HeaderContent` section resides near the top of the page and extends across the entire page width while the `MainContent` section appears below the `HeaderContent` section and extends across the entire page. Most importantly, these sections accept the same cartridge types, and therefore, set up the same cartridge containment strategy for pages that use the `OneColumnTemplate`. Note that, out of the box, Commerce Reference Store does not use the `OneColumnPage` template.

Content Types and Their Cartridges

Commerce Reference Store has three content types that correspond to the three sections of the `TwoColumnPage` template: `HeaderContent`, `SecondaryContent`, and `MainContent`. The lists below show the cartridges that exist for each content type. Asterisks mark the cartridges that are custom built for the ATG platform. The other cartridges are included with the Endeca platform.

`HeaderContent` cartridges include:

- `ContentSlot-Header`
- `HeaderBanner-ATGCategory*`

`SecondaryContent` cartridges include:

- `ContentSlot-Secondary`
- `Breadcrumbs`

- `GuidedNavigation`
  - `RefinementMenu`
  - `PriceSlider*`
- `PromotionalContent-ATGSlot*`
- `PromotionalContent-ATGTargeter*`

`MainContent` cartridges include:

- `ContentSlot-Main`
- `HorizontalRecordSpotlight`
- `ProductList-ATGCategoryChildren*`
- `ProductSpotlight-ATGCategoryRecommendations*`
- `ProductSpotlight-ATGCategoryRelatedProducts*`
- `ProductSpotlight-ATGSlot*`
- `ProductSpotlight-ATGTargeter*`
- `ResultsList`

In addition to the three content types defined for the `TwoColumnPage` template, Commerce Reference Store has one additional content type, `SearchBox`. There is one cartridge that uses this content type, also called `SearchBox`. Commerce Reference Store uses this cartridge to render the search box that appears in the header on each page (see SearchBox (page 138) for more details).

Note that Commerce Reference Store defines more cartridges than it actually uses. Specifically, Commerce Reference Store defines the `PromotionalContent-ATGTargeter`, `ProductSpotlight-ATGSlot`, `ProductSpotlight-ATGTargeter`, and `ProductSpotlight-ATGCategoryRelatedProducts` cartridges but it does not use them on its pages out of the box. These cartridges are provided for the convenience of developers who use Commerce Reference Store as a starting point for their own custom development.

Commerce Reference Store Cartridge Hierarchy

In Experience Manager, Commerce Reference Store defines a `browse` page, accessed via the `/browse` URL, as its starting point for page-based queries. The `browse` page has a single Endeca content slot for items of type `Page`. Commerce Reference Store configures this slot to get its content from the `/content/Web/Web Browse Pages` content collection.

The `Web Browse Pages` content collection contains two items: `Category Page` and `Default Browse Page`. The `Category Page` item is triggered when the query is composed solely of a dimension value from the Category dimension. The `Default Browse Page` is triggered for all other conditions. Both pages use the `TwoColumnPage` template with its three sections--`HeaderContent`, `SecondaryContent`, and `MainContent`--however, the cartridges included in the three sections are slightly different.

Note: Any mechanism that generates links for the `browse` page must use the `/browse` URL (note that a different URL is configured for the browse page for mobile applications). Commerce Reference Store cartridge handlers use the `/atg/endeca/assembler/cartridge/manager/DefaultActionPathProvider` component provided with the ATG Platform to generate these links properly. See *Controlling How Cartridges Generate URLs* in the *Endeca Integration Guide* for more information on this component.

Category Page Configuration

This section describes how the Category Page populates the three sections of the `TwoColumnPage` template.

The `HeaderContent` section includes a `HeaderBanner-ATGCategory` cartridge instance, named `Category Header Banner`, which renders the category title, a background image, and a promotional content item.

The `MainContent` section includes:

- A `ProductList-ATGCategoryChildren` cartridge instance, called `Category Products Grid`, that renders the child products of the current category.
- A `ContentSlot-Main` cartridge instance, called `Featured Items`, that renders a set of featured products for the current category. This Endeca content slot gets its content from the `/content/Shared/Spotlights` content collection. The `Spotlights` content collection has a single item, called `Featured Items`, that is returned for all locations. The `Featured Items` item gets its contents from a `HorizontalRecordSpotlight` cartridge. This cartridge is configured to return the top four products in the current category, as defined by margin.
- A `ProductSpotlight-ATGCategoryRecommendations` cartridge instance, named `Recommendations`, that renders the products that Oracle Recommendations has returned for the current category.

The `SecondaryContent` section includes:

- A `Breadcrumbs` cartridge instance, named `Breadcrumbs`, that renders the breadcrumbs that track a shopper's search and navigation choices.
- A `ContentSlot-Secondary` cartridge instance, named `Guided Navigation`, that renders a set of refinement links. This Endeca content slot gets its content from the `/content/Shared/GuidedNavigation` content collection. The `GuidedNavigation` content collection has a single item, called `Default Guided Navigation`, that is returned for all locations. The `Default Guided Navigation` item gets its contents from:
    - `RefinementMenu` cartridge instances for each dimension that appears in the left hand rail (`Category`, `Size`, `Brand`, `Color`, `Wood Finish`, `Feature`).
    - An instance of the `PriceSlider` cartridge, called `Price`, that renders a slider that filters records based on their `activePrice` property.
- Two `PromotionalContent-ATGSlot` cartridge instances, named `Promotional Content 1` and `Promotional Content 2`, which reference ATG slots. These ATG slots provide promotional content items that inform the shopper about Commerce Reference Store promotions.

Default Browse Page Configuration

The `Default Browse Page` uses a similar structure to the `Category Page` for populating the sections of the `TwoColumnPage` template, as described below:

- The `HeaderContent` section of the `Default Browse Page` does not include any additional cartridges (in other words, it does not render a header banner like the `Category Page` does).
- The `SecondaryContent` section of the `Default Browse Page` uses the same configuration as the `Category Page`.

- The `MainContent` section includes a single `ContentSlot-Main` cartridge instance, called `Results List`, that renders the products that are appropriate for the shopper's current navigation state. This Endeca content slot gets its content from the `/content/Shared/Results List` content collection. The `Results List` content collection has a single item, called `Default Results List`, that is returned for all locations. The `Default Results List` item gets its contents from a `ResultsList` cartridge that has been configured to return 12 records per page.

Identifying Category Page Requests

To determine if a request is for a category page, Commerce Reference Store includes a helper component, `/atg/endeca/assembler/cartridge/StoreCartridgeTools`. This component determines if the current request is for a category page and, if so, it returns the current category's ID. The `StoreCartridgeTools` component is of class `atg.projects.store.assembler.cartridge.StoreCartridgeTools`, which has two methods:

- `isUserOnCategoryPage()` - Determines whether the user is currently on the category page by checking whether the current navigation state contains the `CategoryOnly` user segment. (See Tracking a Customer's Catalog Navigation (page 121) for more details on the `CategoryOnly` user segment.)
- `getCurrentCategoryId()` - Returns the ID for the category the user is currently viewing. This method first checks whether the user is currently on the category page, using the `isUserOnCategoryPage()` method, and, if so, it retrieves the last browsed category ID from the component specified in the `StoreCartridgeTools.catalogNavigation` property.

To do its tasks, the `StoreCartridgeTools` component has the following properties configured:

- `catalogNavigation` - The `CatalogNavigation` component where the last browsed category is stored. Set by default to `/atg/endeca/assembler/cartridge/manager/NavigationStateBuilder.catalogNavigation`, which resolves to `/atg/store/catalog/CatalogNavigation`.
- `userState` - The `UserState` component that holds the user segments list. Set by default to `/atg/endeca/assembler/cartridge/manager/NavigationStateBuilder.userState`, which resolves to `/atg/endeca/assembler/cartridge/manager/user/LiveUserState`.
- `categoryNavigationUserSegment` - The name of the user segment that indicates a request is for a category page. Set by default to `/atg/endeca/assembler/cartridge/manager/NavigationStateBuilder.userSegment`, which resolves to `CategoryOnly`.

Because it references other request-scoped components, specifically `/atg/endeca/assembler/cartridge/manager/user/LiveUserState`, the `StoreCartridgeTools` component is also configured to be request-scoped.

Tracking a Customer's Catalog Navigation

Several of Commerce Reference Store's features, such as the Continue Shopping and breadcrumbs features, require knowing the most recent category a customer has browsed. To enable tracking of a customer's catalog navigation, Commerce Reference Store implements the `/atg/store/catalog/CatalogNavigation` component, of class `atg.projects.store.catalog.CatalogNavigationService`. This session-based component stores information about the customer's current category and that category's position in the Category dimension hierarchy. The `CatalogNavigation` component provides:

- The ID of the current category.
- The IDs of the ancestor categories for the current category.
- The top-level category associated with the current category.
- The full category hierarchy path from the current category up to its top-level parent.

The data stored in the `CatalogNavigation` component is updated each time the customer makes a request for a category page.

To determine if the request is for a category page, first the URL is parsed. This is the responsibility of the `/atg/endeca/assembler/cartridge/manager/NavigationStateBuilder` component, which is of class `atg.endeca.assembler.navigation.ExtendedNavigationStateBuilder` class. The `ExtendedNavigationStateBuilder` class is an extension of the `endeca.infront.navigation.url.UrlNavigationStateBuilder` class. The `UrlNavigationStateBuilder` class parses the URL and creates an initial `com.endeca.infront.navigation.NavigationState` object. The `ExtendedNavigationStateBuilder` class then modifies that `NavigationState` object.

To specify how the `NavigationState` object should be modified, the `NavigationStateBuilder` component has a `navigationStateProcessors` property that consists of a list of `NavigationStateProcessor` components. These components must be of a class that implements the `atg.endeca.assembler.navigation.NavigationStateProcessor` interface. This interface defines a `process()` method that contains logic that modifies the `NavigationState` object in some manner. The `NavigationStateBuilder` component invokes each `NavigationStateProcessor`, executing the logic contained within.

Commerce Reference Store implements a new processor component, `/atg/endeca/assembler/cartridge/manager/StoreNavigationStateProcessor`, and adds it to the `NavigationStateBuilder.navigationStateProcessors` property. The `StoreNavigationStateProcessor` component is of class `atg.projects.store.assembler.navigation.StoreNavigationStateProcessor` and it determines if the request is for a single dimension value ID. If it does, the `StoreNavigationStateProcessor` component retrieves the dimension value ID from the `NavigationState` object and attempts to find a match for it in the dimension value cache. If the ID exists in the cache, it indicates that this is a Category dimension value ID (as opposed to a dimension value ID from another dimension) because the cache only contains Category dimension value IDs.

If the `StoreNavigationStateProcessor` component finds a match, it does the following:

- It updates the `CatalogNavigation` component with the information it finds in the cache for the specified dimension value ID.
- It adds the `CategoryOnly` user segment to the `com.endeca.infront.navigation.UserState` object, to indicate that this is a category page request.

Note that categories that can be accessed via multiple paths in the Category hierarchy have multiple dimension value IDs, one for each path. Therefore, the ID that is parsed from the URL will indicate not only the Category dimension value itself, but also its hierarchical path. This ensures that when the `StoreNavigationStateProcessor` component maps the dimension value ID to the category ID in the dimension value cache, it also maps to the correct hierarchy.

Custom Cartridges

This section describes custom cartridges included with Commerce Reference Store.

HeaderBanner-ATGCategory

The `HeaderBanner-ATGCategory` cartridge displays the header banner defined for the currently selected category on category pages.

Note: This cartridge should only be used on category pages. If you place it on a browse page, it will not render any content. See Category Pages versus Browse Pages (page 116) for more details on the differences between the two page types.

Template

The XML template for the `HeaderBanner-ATGCategory` cartridge, located in `<ATG10dir>/CommerceReferenceStore/Store/Storefront/deploy/cartridge_templates/HeaderContent-HeaderBanner-ATGCategory.xml`, sets the cartridge's content type to `HeaderContent` and its ID to `HeaderBanner-ATGCategory`. There are no editable properties defined for the `HeaderBanner-ATGCategory` cartridge; all of its content is dynamically generated by the handler.

Content Item and Handler

The `atg.projects.store.assembler.cartridge.CategoryHeaderBannerContentItem` class represents the content item that the `HeaderBanner-ATGCategory` cartridge returns to the renderer. The `CategoryHeaderBannerContentItem` class has three properties:

- `headerTitle`: Contains the display name for the current category, for example, Women, Men, Shirts, and so on.
- `backgroundBannerURL`: Contains the URL for the hero image for the top-level category associated with the current category (for example, if the current category is Dresses, the top-level category is Women). This image is rendered behind both the `headerTitle` and the promotional content item specified by the `promotionalContentId` property.
- `promotionalContentId`: Contains the ID of the promotional content item for the top-level category associated with the current category. The promotional content item gets rendered in the right hand side of the header banner.

The `/atg/endeca/assembler/cartridge/handler/CategoryHeaderBanner` component, which is of class `atg.projects.store.assembler.cartridge.handler.CategoryHeaderBannerHandler`, is the cartridge handler responsible for creating and populating the `CategoryHeaderBannerContentItem` object. The `CategoryHeaderBannerHandler` class is an extension of the `com.endeca.infront.cartridge.NavigationCartridgeHandler` class with an overridden `process()` method. The `process()` method retrieves the current navigation state from the `/atg/endeca/assembler/cartridge/manager/NavigationState` component and creates a new `CategoryHeaderBannerContentItem` object. A call to the `CategoryHeaderBannerHandler.loadBannerItemDetails()` method populates the `CategoryHeaderBannerContentItem` object with data from the catalog repository and returns it to the JSP renderer.

To populate the content item, the `loadBannerItemDetails()` method uses the `/atg/endeca/assembler/cartridge/StoreCartridgeTools` component to retrieve the currently chosen category and its associated top-level category from the `atg.projects.store.catalog.CatalogNavigationService` class. The `loadBannerItemDetails()` method sets the content item's `headerTitle` property to the current category's `displayName` value. It also sets the `backgroundBannerURL` and `promotionalContentId` content item properties using the values specified for the top-level category's `heroImage` and `feature` properties.

Note: A category's `feature` and `myFeature` properties work together to ensure that a promotional content item is always returned for a category. For more information on these properties, see the *Product Catalog* (page 85) chapter.

Note: If the `CatalogNavigationService` class returns null for the current category and top-level category IDs, it means that the `CategoryHeaderBanner` has been incorrectly placed on a browse page and the cartridge should not be rendered. The `CategoryHeaderBanner` component tests for this situation and handles it appropriately.

To do its tasks, the `CategoryHeaderBanner` component has two configurable properties:

- `catalogTools`: This property specifies which `CatalogTools` component the `CategoryHeaderBanner` component should use to access product catalog data. Out of the box, this property is set to `/atg/commerce/catalog/CatalogTools`.
- `storeCartridgeTools`: This property specifies which `StoreCartridgeTools` component the `CategoryHeaderBanner` component should use to access the `CatalogNavigationService` cache. Out of the box, this property is set to `/atg/endeca/assembler/cartridge/StoreCartridgeTools`.

The `CategoryHeaderBanner` component also inherits a number of properties from the `/atg/endeca/assembler/cartridge/handler/NavigationCartridgeHandler` component, through a `$basedOn` property configuration. Specifically, `CategoryHeaderBanner` inherits a reference to the `/atg/endeca/assembler/cartridge/manager/NavigationState` component that provides information on the shopper's current navigation state.

JSP Renderer

The `store.war/cartridges/HeaderBanner-ATGCategory/HeaderBanner-ATGCategory.jsp` page renders the content in the `CategoryHeaderBannerContentItem` object. This renderer retrieves the `headerTitle` and `headerBannerURL` values from the `CategoryHeaderBannerContentItem` object and uses these values to render the display name and background image for the banner. `HeaderBanner-ATGCategory.jsp` also retrieves the `promotionalContentId` value and passes it to the `/atg/commerce/promotional/PromotionalContentLookup` component to render the promotional content item. This component is of class `atg.commerce.catalog.custom.CatalogItemLookupDroplet` and it specifies the catalog repository as the repository to be searched and `promotionalContent` as the item type to search for.

When a promotional content item that includes a link is clicked, it takes the shopper to the page specified in the promotional content item's `associatedCategory` property. The link for this page must be constructed using an Endeca URL. To satisfy this requirement, the template used to render promotional content items, `store.war/promo/gadgets/linkedImage.jsp`, uses the `<crs:promotionalContentWrapper.tag>`. This tag calls the `/atg/commerce/endeca/cache/DimensionValueCacheDroplet` servlet bean to retrieve the correct URL for the `associatedCategory` and then it associates that URL value with the link reference.

Note: See the *ATG Endeca Integration Guide* for information about the `DimensionValueCacheDroplet` servlet bean and the cache.

ProductList-ATGCategoryChildren

The `ProductList-ATGCategoryChildren` cartridge displays the child products for the currently selected category on category pages.

Note: This cartridge should only be used on category pages. If you place it on a browse page, it will not render any content. See Category Pages versus Browse Pages (page 116) for more details on the differences between the two page types.

Template

The XML template for the `ProductList-ATGCategoryChildren` cartridge, located in `<ATG10dir>/CommerceReferenceStore/Store/Storefront/deploy/cartridge_templates/MainContent-ProductList-ATGCategoryChildren.xml`, sets the cartridge's content type to `MainContent` and its ID to `ProductList-ATGCategoryChildren`. The XML template defines one editable property, `recordsPerPage`, that defines the number of child products to show for the current category.

Content Item and Handler

The `atg.projects.store.assembler.cartridge.ProductListCategoryChildrenContentItem` class represents the content item that the `ProductList-ATGCategoryChildren` cartridge returns to the renderer. The `ProductListCategoryChildrenContentItem` class has three properties:

- `categoryId`: The ID of the current category.
- `recordsPerPage`: The number of records to be displayed.
- `totalNumRecs`: The total number of child products in the category.

Note that a `ProductListCategoryChildrenContentItem` object does not contain the actual child products. It is the JSP renderer's responsibility to retrieve the correct child products from the catalog repository based on the `categoryId` returned in the content item.

The `/atg/endeca/assembler/cartridge/handler/ProductListCategoryChildren` component, which is of class `atg.projects.store.assembler.cartridge.handler.ProductListCategoryChildrenHandler`, is the cartridge handler responsible for creating and populating the `ProductListCategoryChildrenContentItem` object. The `ProductListCategoryChildrenHandler` class is an extension of the `com.endeca.infront.cartridge.NavigationCartridgeHandler` class with an overridden `process()` method. The `process()` method retrieves the current navigation state from the `/atg/endeca/assembler/cartridge/manager/NavigationState` component and creates a new `ProductListCategoryChildrenContentItem` object. A call to the `ProductListcategoryChildrenHandler.loadCategoryDetails()` method populates the `ProductListCategoryChildrenContentItem` object with data from the catalog repository and returns it to the JSP renderer.

To do its tasks, the `ProductListCategoryChildren` component has two configurable properties:

- `catalogTools`: This property specifies which `CatalogTools` component the `ProductListCategoryChildren` component should use to access product catalog data. Out of the box, this property is set to `/atg/commerce/catalog/CatalogTools`.
- `catalogNavigation`: This property references the `/atg/store/catalog/CatalogNavigation` component, which is a Commerce Reference Store-specific component that retrieves the ID for the current category.

The `ProductListCategoryChildren` component also inherits a number of properties from the `/atg/endeca/assembler/cartridge/handler/NavigationCartridgeHandler` component, through a `$basedOn` property configuration. Specifically, `ProductListCategoryChildren` inherits a reference to the `/atg/endeca/assembler/cartridge/manager/NavigationState` component that provides information on the shopper's current navigation state.

JSP Renderer

The `store.war/cartridges/ProductListCategoryChildren/ProductListCategoryChildren.jsp` page renders the content in the `ProductListCategoryChildrenContentItem` object. This renderer retrieves the `categoryId` value from the `ProductListCategoryChildrenContentItem` object and uses this value as a parameter to the `/atg/commerce/catalog/CategoryLookup` servlet bean to retrieve the category and its child products from the Catalog repository. The child products are then passed to the `/atg/store/sort/RangeSortDroplet` servlet bean, along with the `recordsPerPage` value returned in the content item, to sort and render the child product list.

PromotionalContent-ATGSlot, PromotionalContent-ATGTargeter, ProductSpotlight-ATGSlot, and ProductSpotlight-ATGTargeter

The `PromotionalContent-ATGSlot` and `PromotionalContent-ATGTargeter` cartridges display promotional content items in the left hand rail of both the category and browse pages. The `ProductSpotlight-ATGSlot` and `ProductSpotlight-ATGTargeter` display products in the main content area of both the category and browse pages. The content for these cartridges is provided by ATG slots and targeters that have been defined in the ATG Control Center and the Business Control Center, respectively. All four cartridges have a similar structure, described below.

Template

The XML templates for the `PromotionalContent-ATGSlot`, `PromotionalContent-ATGTargeter`, `ProductSpotlight-ATGSlot`, and `ProductSpotlight-ATGTargeter` cartridges are located in equivalently named files in the `<ATG10dir>/CommerceReferenceStore/Store/Storefront/deploy/cartridge_templates` directory. Each XML file defines the following properties:

- `componentPath`: This property defines either a partial or a full path to the ATG slot or targeter that is providing the content. If a partial path is provided in Experience Manager, the cartridge handler prepends this value with the correct ATG registry path.
- `itemCount`: This property determines the number of items the cartridge will render. For the `PromotionalContent` cartridges, this value is fixed at 1 and is not editable in Experience Manager. For the `ProductSpotlight` cartridges, this value defaults to 4 and is editable in Experience Manager (out of the box, Commerce Reference Store is designed to handle four products on a single line in the main content area; if more than four products are returned, the products wrap to a new line).

Content Item and Handler

The `atg.projects.store.assembler.cartridge.TargetedItemsContentItem` class represents the content item that the `PromotionalContent-ATGSlot`, `PromotionalContent-ATGTargeter`, `ProductSpotlight-ATGSlot`, and `ProductSpotlight-ATGTargeter` cartridges return to the renderer. The `TargetedItemsContentItem` class has one property, called `items`, that contains the items that were returned by the ATG slot or targeter. The number of items contained in this property is limited by the `itemCount` value defined in the Experience Manager configuration.

The `atg.projects.store.assembler.cartridge.handler.TargetedItemsHandler` class creates and populates the `TargetedItemsContentItem` object. Four components use this class, one for each cartridge type:

- `/atg/endeca/assembler/cartridge/handler/PromotionalContentSlot`
- `/atg/endeca/assembler/cartridge/handler/PromotionalContentTargeter`
- `/atg/endeca/assembler/cartridge/handler/ProductSpotlightSlot`

- `/atg/endeca/assembler/cartridge/handler/ProductSpotlightTargeter`

The `/atg/endeca/assembler/NucleusAssemblerFactory.handlerMapping` property includes the following mappings to associate each ATG slot or targeter cartridge with the correct handler component:

```
PromotionalContent-ATGSlot=\
     /atg/endeca/assembler/cartridge/handler/PromotionalContentSlot,\
PromotionalContent-ATGTargeter=\
     /atg/endeca/assembler/cartridge/handler/PromotionalContentTargeter,\
ProductSpotlight-ATGSlot=\
     /atg/endeca/assembler/cartridge/handler/ProductSpotlightSlot,\
ProductSpotlight-ATGTargeter=\
     /atg/endeca/assembler/cartridge/handler/ProductSpotlightTargeter,\
```

The four handler components all have the following properties:

- `pathPrefix`: A prefix that is prepended, if necessary, to the `componentPath` defined by the XML template in order to create a fully-qualified path to the ATG slot or targeter. Each component sets this prefix by linking to the properties listed in the table below.
- `sourceMap`: The path to the `TargetingSourceMap` component that is used during the targeting operations. (See the *ATG Personalization Programming Guide* for more information on `TargetingSourceMap` components.)
- `itemDescriptor`: The item descriptor type that should be returned, for example, `promotionalContent` or `product`.
- `repository`: The repository that contains the item descriptor type that should be returned. Item descriptor types are not unique across repositories, so the handler must compare both the item descriptor type and the repository type when calculating which items to return.

The `sourceMap` property is the same for all four components, `/atg/targeting/TargetingSourceMap`. Each component has its own values for the `pathPrefix` and `itemDescriptor`, as shown in the table below:

Component	**pathPrefix**	**itemDescriptor**
`PromotionalContentSlot`	`/atg/registry/SlotRegistry.componentRootPath`	`promotionalContent`
`PromotionalContentTargeter`	`/atg/registry/TargeterRegistry.componentRootPath`	`promotionalContent`
`ProductSpotlightSlot`	`atg/registry/SlotRegistry.componentRootPath`	`product`
`ProductSpotlightTargeter`	`/atg/registry/TargeterRegistry.componentRootPath`	`product`

The `TargetedItemsHandler` class is an extension of the `com.endeca.infront.cartridge.NavigationCartridgeHandler` class with overrides for the `process()` method. The `process()` method determines if the `componentPath` provided by Experience Manager is a complete path and, if not, it prepends the `componentPath` with the value in `pathPrefix`. After a complete path is determined, the ATG slot or targeter component is resolved and the number of items specified in the `itemCount` property are retrieved from the appropriate ATG repository (if no `itemCount` property is supplied, all of an ATG slot or targeter's items are returned).

JSP Renderer

Four renderers are included in Commerce Reference Store, one for each cartridge:

- `store.war/cartridges/PromotionalContent-ATGSlot/PromotionalContent-ATGSlot.jsp`
- `store.war/cartridges/PromotionalContent-ATGTargeter/PromotionalContent-ATGTargeter.jsp`
- `store.war/cartridges/ProductSpotlight-ATGSlot/ProductSpotlight-ATGSlot.jsp`
- `store.war/cartridges/ProductSpotlight-ATGTargeter/ProductSpotlight-ATGTargeter.jsp`

The `PromotionalContent` JSP pages iterate through the `TargetedItemsContentItem.items` property and include the `item.template.url` page for each item. The `ProductSpotlight` JSP pages iterate through the `TargetedItemsContentItem.items` property and include the `store.war/global/gadgets/promotedProductRenderer.jsp` page for each item.

The "Featured Items" label that appears above the product spotlight row in the main content area, along with its translated versions, is stored in the `StoreText` repository in the `productTargeter_spotlightTitle` item and may be edited in the Business Control Center.

ProductSpotlight-ATGCategoryRelatedProducts

The `ProductSpotlight-ATGCategoryRelatedProducts` cartridge displays any related products, or cross-sells, defined for the currently selected category on category pages. Out of the box, this cartridge is not used by Commerce Reference Store but Commerce Reference Store's deployment template does make it available for selection in Experience Manager.

Note: This cartridge should only be used on category pages. If you place it on a browse page, it will not render any content. See Category Pages versus Browse Pages (page 116) for more details on the differences between the two page types.

Template

The XML template for the `ProductSpotlight-ATGCategoryRelatedProducts` cartridge, located in `<ATG10dir>/CommerceReferenceStore/Store/Storefront/deploy/cartridge_templates/MainContent-ProductSpotlight-ATGCategoryRelatedProducts.xml`, sets the cartridge's content type to `MainContent` and its ID to `ProductSpotlight-ATGCategoryRelatedProducts`. The XML template defines one editable property, `numberOfRecords`, that specifies the number of cross-sell products to show for the current category. By default, `numberOfRecords` is set to 4.

Content Item and Handler

The `atg.projects.store.assembler.cartridge.CategoryRelatedProductsContentItem` class represents the content item that the `ProductSpotlight-ATGCategoryRelatedProducts` cartridge returns to the renderer. The `CategoryRelatedProductsContentItem` class has three properties:

- `categoryId`: The ID of the current category.
- `numberOfRecords`: The number of cross-sell products to be displayed.
- `relatedProducts`: The list of related products for the current category.

The /atg/endeca/assembler/cartridge/handler/CategoryRelatedProducts component, which is of class atg.projects.store.assembler.cartridge.handler.CategoryRelatedProductsHandler, is responsible for creating and populating the CategoryRelatedProductsContentItem object. The CategoryRelatedProductsHandler class is an extension of the com.endeca.infront.cartridge.NavigationCartridgeHandler class.

The CategoryRelatedProducts component uses the methods provided by the /atg/endeca/assembler/cartridge/StoreCartridgeTools component to get the ID for the current category. Once it has the ID, CategoryRelatedProducts can retrieve the list of related products for the correct category from the catalog repository. To do this, CategoryRelatedProducts uses its relatedProductsProperties property to determine which category item properties contain related products (out of the box, this property is set to a single repository item property, relatedProducts). CategoryRelatedProducts then retrieves the related products from those properties. The number of products retrieved is limited by the numberOfRecords value specified in the cartridge configuration.

CategoryRelatedProducts may also apply an ATG filter to the list of products it retrieves, for example, to filter out products that are not appropriate for the current site context. The filter to apply is configured by the CategoryRelatedProducts.filter property. Out of the box, CategoryRelatedProducts.filter is set to /atg/store/collections/filter/CatalogItemFilter, which filters out products based on their start and end dates.

The CategoryRelatedProducts component has the following configurable properties:

- relatedProductsProperties: Lists the category repository item properties that contain related products. Defaults to relatedProducts.
- catalogTools: Specifies which CatalogTools component the ProductListCategoryChildren component should use to access product catalog data. Defaults to /atg/commerce/catalog/CatalogTools.
- filter: An optional property that specifies a filter to apply to the related products list. Defaults to /atg/store/collections/filter/CatalogItemFilter.
- storeCartridgeTools: References the StoreCartridgeTools component that CategoryRelatedProducts uses to retrieve the ID of the current category. Defaults to /atg/endeca/assembler/cartridge/StoreCartridgeTools.

JSP Renderer

The store.war/cartridges/ProductSpotlight-ATGCategoryRelatedProducts/ProductSpotlight-ATGCategoryRelatedProducts.jsp page renders the content in the CategoryRelatedProductsContentItem object. This renderer retrieves the list of products from the content item's relatedProducts property. If this list is empty, either because no related products exist for the category or because the ProductSpotlight-ATGCategoryRelatedProducts cartridge was inadvertently placed on a browse page instead of a category page, then the products and the label that introduces them are not rendered.

The "Related Items" label that appears above the product spotlight row in the main content area, along with its translated versions, is stored in the StoreText repository in the product_spotlight_categoryRelatedProductsTitle item and may be edited in the Business Control Center.

ProductSpotlight-ATGCategoryRecommendations

The ProductSpotlight-ATGCategoryRecommendations cartridge renders products returned by Oracle Recommendations on Demand for the current category on category pages.

Note: This cartridge should only be used on category pages. If you place it on a browse page, it will not render any content. See Category Pages versus Browse Pages (page 116) for more details on the differences between the two page types.

Note: The `ProductSpotlight-ATGCategoryRecommendations` cartridge provides the rendering mechanism for the products returned by Oracle Recommendations. The Oracle Recommendations integration also includes functionality for page instrumentation and making calls to the Oracle Recommendations engine. This functionality is described in the *Oracle Recommendations On Demand Integration* (page 167) chapter. This section assumes you are familiar with the information in that section.

Template

The XML template for the `ProductSpotlight-ATGCategoryRecommendations` cartridge, located in `<ATG10dir>/CommerceReferenceStore/Store/Storefront/deploy/cartridge_templates/MainContent-ProductSpotlight-ATGCategoryRecommendations.xml`, sets the cartridge's content type to `MainContent` and its ID to `ProductSpotlight-ATGCategoryRecommendations`. The XML template defines one editable property, `numberOfRecords`, that defines the number of recommendations to show for the current category. By default, `numberOfRecords` is set to 4.

Content Item and Handler

The `atg.projects.store.assembler.cartridge.CategoryRecommendationsContentItem` class represents the content item that the `ProductSpotlight-ATGCategoryRecommendations` cartridge returns to the renderer. The `CategoryRecommendationsContentItem` class has two properties:

- `categoryId`: The ID of the current category.
- `numberOfRecords`: The number of recommended products to be displayed.

The `/atg/endeca/assembler/cartridge/handler/CategoryRecommendations` component, which is of class `atg.projects.store.assembler.cartridge.handler.CategoryRecommendationsHandler`, is responsible for creating and populating the `CategoryRecommendationsContentItem` object. The `CategoryRecommendationsHandler` class is an extension of the `com.endeca.infront.cartridge.NavigationCartridgeHandler` class.

The `CategoryRecommendations` component uses the methods provided by the `StoreCartridgeTools` component to get the ID for the current category. The `CategoryRecommendations` component then stores this ID, along with the value specified in Experience Manager for the number of records to display, in the `CategoryRecommendationsContentItem` object. It is the renderer's responsibility to take this information stored in the content item, make an appropriate request to the Oracle Recommendations engine, and render the results. The `CategoryRecommendations` component has one configurable property, `storeCartridgeTools`, which references the `StoreCartridgeTools` component that `CategoryRecommendations` should use. Out of the box, this property is set to `/atg/endeca/assembler/cartridge/StoreCartridgeTools`.

JSP Renderer

The `store.war/cartridges/ProductSpotlight-ATGCategoryRecommendations/ProductSpotlight-ATGCategoryRecommendations.jsp` page is the renderer for the `ProductSpotlight-ATGCategoryRecommendations` cartridge. This page is responsible for inserting a container `<div>` element with content that specifies information about the recommendations to be displayed, including the number of products to show and any products to be excluded from the list of recommendations.

The `ProductSpotlight-ATGCategoryRecommendations.jsp` page checks for the existence of the `/atg/store/recommendations/StoreRecommendationsConfiguration` component, which is part of the `Store.Recommendations` module. If this component is present, it indicates Oracle Recommendations is running, and the `ProductSpotlight-ATGCategoryRecommendations.jsp` page should render the `<div>` element.

Calculating the Products to Exclude

When creating the `<div>` element, the `ProductSpotlight-ATGCategoryRecommendations.jsp` page starts with the data stored in the `CategoryRecommendationsContentItem` object that indicates the number of products to display and the current category ID. The page must also calculate the list of products to exclude. This functionality is required to avoid showing products in the Recommended Items area that are already displayed elsewhere on the page. There are several other cartridges on a category page that can potentially show products, specifically `ProductSpotlight-ATGCategoryRelatedProducts`, `HorizontalRecordSpotlight`, and `ProductList-ATGCategoryChildren`.

To determine the products to exclude, the `ProductSpotlight-ATGCategoryRecommendations.jsp` page invokes the `/atg/endeca/assembler/cartridge/CrossCartridgeItemsLookupDroplet` servlet bean. This servlet bean retrieves the root content item object from the current request's `rootContentItem` attribute and passes it to the `/atg/endeca/assembler/cartridge/CrossCartridgeItemsLookup` component's `traverse()` method. The `traverse()` method returns a list of products that are already being displayed on the page by other cartridges. The renderer then inserts that list into the `<div>` element.

The `CrossCartridgeItemsLookupDroplet` servlet bean is a request-scoped component of class `atg.projects.store.assembler.cartridge.CrossCartridgeItemsLookupDroplet`. It has one property, `contentItemTreeIterator`, that specifies the component that should be used to traverse the content item tree to find already-displayed products. This property is set to `/atg/endeca/assembler/cartridge/CrossCartridgeItemsLookup`.

The `CrossCartridgeItemsLookup` component is of class `atg.projects.store.assembler.cartridge.CrossCartridgeItemsLookup`, which is an implementation of the `atg.projects.store.assembler.cartridge.ContentItemTreeIterator` abstract class. The `ContentItemTreeIterator` abstract class implements a `traverse()` method that takes a content item as an input parameter and then traverses that content item, looking for descendent content items. For each content item detected, `ContentItemTreeIterator` calls its `process()` method. The `ContentItemTreeIterator` class is able to detect and handle circular references, terminating content item traversal and throwing an exception when a circular reference occurs.

The `CrossCartridgeItemsLookup` class inherits the traversing logic from the `ContentItemTreeIterator` abstract class and implements a `process()` method that checks whether a content item is of a specified type and, if so, it processes the content item accordingly. The `CrossCartridgeItemsLookup` class has two properties that define the content item types to look for:

- `typeToPropertyNameMap` – A mapping between cartridges that display products and the property names in the associated content items where the displayed products are stored.
- `contentItemsToCache` – A list of cartridges whose associated content items should be cached during content item tree traversing.

If a cartridge's type is specified in the `typeToPropertyNameMap` mapping, the `process()` method tries to retrieve the products displayed by the cartridge. Displayed products are retrieved from the property configured in the `typeToPropertyNameMap` mapping. It is expected that products are represented either by a product ID, a `com.endeca.infront.cartridge.model.Record` object or an `atg.repository.RepositoryItem` object. Once all the displayed products are retrieved from the content item hierarchy, their IDs are stored in a single shared `displayedItems` property that forms the basis of the items to exclude list.

If a cartridge's type is specified in the `contentItemsToCache` list, the `process()` method caches the associated content item into a `contentItemCache` map. This caching is required for cartridges whose content items do not directly contain displayed products or product IDs. For example, the content item for the `ProductList-ATGCategoryChildren` cartridge contains information about the products to be displayed, such as the ID of the parent category and the number of products to display, but it does not contain any actual product data. In this case, the content item data is cached so that the `ProductSpotlight-ATGCategoryRecommendations` cartridge renderer can use it to calculate which product IDs are already being displayed by the `ProductList-ATGCategoryChildren` cartridge.

The `CrossCartridgeItemsLookup` component is configured as follows:

```
# Mapping between cartridge type and property name containing products
typeToPropertyNameMap=\
  ProductSpotlight-ATGCategoryRelatedProducts=relatedProducts,\
  HorizontalRecordSpotlight=records

# List of IDs for cartridges that do not directly contain product data but instead
# contain data that allow the product IDs to be calculated. This information is
# cached so that the renderer can use it to calculate which product IDs are
# already being displayed.
contentItemsToCache=ProductList-ATGCategoryChildren
```

Rendering the Returned Products

Part of the page instrumentation that occurs for Oracle Recommendations is the insertion in the `<head>` element of a call to the Oracle Recommendations JavaScript library:

```
<script type="text/javascript" src="//static.atgsvcs.com/js/atgsvcs.js"></script>
```

After the HTML page loads, the `atgsvcs.js` script executes and sends the information from the container `<div>` about the product recommendations to be returned by the Oracle Recommendations engine (the container `<div>` information is sent along with Oracle Recommendations clickstream code; see the *Oracle Recommendations On Demand Integration* chapter for more details on clickstream code). Oracle Recommendations returns a response containing data about the products to display, and executes JavaScript renderer code that manipulates the DOM of the page to alter the HTML to display the recommendations.

Modifying the Recommended Items Label

The "Recommended Items" label that appears above the product spotlight row in the main content area, along with its translated versions, is stored in the `StoreText` repository in the `product_spotlight_categoryRelatedProductsTitle` item and may be edited in the Business Control Center.

PriceSlider

The `PriceSlider` cartridge allows a Commerce Reference Store shopper to refine the current results list based on the active price of the skus associated with the currently displayed products. The active price is defined as the price that is in effect for the current user on the current site. The `PriceSlider` has two controls that indicate a minimum and maximum price. Items whose active price falls within these two values are included in the results list. A shopper can click and drag the minimum or maximum control to alter the price range to be displayed; alternatively, a shopper can drag the entire range up or down. Upon mouse release, a POST request is sent to the server to retrieve the new results list and the page is refreshed.

The `PriceSlider` applies an Endeca range filter to the current result set. The range filter is applied through an `Nf` URL parameter that gets added to the POST request. The `Nf` parameter takes the form of `Nf=<record property name>|<function>+<value>`. Because the price slider has two controls, a minimum and maximum, the `Nf` parameter takes two different values and uses the between function. For example:

```
Nf=sku.activePrice|BTWN+105+1148
```

Note: For more details on the active price, see the Handling Price Data (page 147) section.

Template

The XML template for the `PriceSlider` cartridge, located in `<ATG10dir>/CommerceReferenceStore/Store/Storefront/deploy/cartridge_templates/Navigation-PriceSlider.xml`, must be available for use within the `GuidedNavigation` cartridge, so its content type is set to `Navigation` while its ID is set to `PriceSlider`. The `Navigation-PriceSlider.xml` file defines one property for the `PriceSlider` cartridge, `priceProperty`, that is not editable in Experience Manager. This property specifies the record property that the slider should be based on which, in this case, is `sku.activePrice`.

Content Item and Handler

The `atg.projects.store.assembler.cartridge.PriceSliderContentItem` class represents the content item that the `PriceSlider` cartridge returns to the renderer. The `PriceSliderContentItem` class has these properties:

- `filterCrumb`: This property references the `com.endeca.infront.cartridge.model.RangeFilterBreadcrumb` object that is created by the `PriceSliderHandler` whenever price slider settings have been applied for the current query. Continue reading below for more information on this object and how it is used.
- `priceProperty`: This property references the record property that the slider is based on which, in this case, is `sku.activePrice`.
- `sliderMin`: Specifies the lower end point for the price slider.
- `sliderMax`: Specifies the upper end point for the price slider.
- `enabled`: Indicates whether the price slider should be rendered. When this property is set to `true`, the slider is rendered. When it is set to `false`, it is not rendered. The `PriceSliderHandler` sets this property based on the number of results returned for the query. If 0 or 1 result is returned, `PriceSliderHandler` sets this property to false.

The `/atg/endeca/assembler/cartridge/handler/PriceSlider` component, which is of class `atg.projects.store.assembler.cartridge.handler.PriceSliderHandler`, is responsible for creating and populating the `PriceSliderContentItem` object. The `PriceSliderHandler` class is an extension of the `com.endeca.infront.cartridge.NavigationCartridgeHandler` class and it overwrites the `preprocess()` and `process()` methods.

The `PriceSliderHandler.preprocess()` method creates an MDEX request but does not execute it. The `process()` method executes the MDEX request to get the search results. It also retrieves the current `com.endeca.infront.navigation.NavigationState` object using the `/atg/endeca/assembler/cartridge/manager/NavigationState` component referenced by the `PriceSliderHandler` component's `navigationStateProperty`. The `NavigationState` object contains the navigation choices that have led to the current result set, including any range filters that have been applied and the properties those range filters have been applied to. If an existing range filter has been applied to the `sku.activePrice` property, it is an indicator that the price slider has already been applied to the current result set and the rendering of the price slider must take into account the existing settings. To do this, the `PriceSliderHandler` creates a `com.endeca.infront.cartridge.model.RangeFilterBreadcrumb` object and populates it with data from the range filter. Specifically, the `RangeFilterBreadcrumb` includes:

- The name of the property the range filter was applied to, the `sku.activePrice` property.

- The operation name, in this case, BTWN for the between operation.
- The upper and lower bounds of the range filter.

The PriceSliderHandler then sets the PriceSliderContentItem.filterCrumb property to reference this RangeFilterBreadcrumb object, thereby making the existing slider settings available to the JSP renderer.

The PriceSliderHandler also sets the slider end points by retrieving the priceSliderMaximumValue and priceSliderMinimumValue properties for the currently searched site (or sites) from the Site repository. The priceSliderMaximumValue and priceSliderMinimumValue properties are site-specific and can be configured through the Site Administration UI. If the current results come from more than one site, the PriceSliderHandler uses the lowest priceSliderMinimumValue and the highest priceSliderMaximumValue from the sites. If no values have been set, PriceSliderHandler uses the defaults of 0 and 1500 for the minimum and maximum values, respectively.

Finally, the PriceSliderHandler evaluates the number of results returned for the query and sets the PriceSliderContentItem.enabled property to true if there is more than one result and false if there is 0 or 1 result.

The PriceSlider component has the following configurable properties:

- storeSitePropertiesManager: Specifies the component that PriceSliderHandler uses to retrieve the names of the site properties that store the slider minimum and slider maximum values. PriceSliderHandler then queries the sites, uses these names, to get the values. Out of the box, this property is set to /atg/multisite/StoreSitePropertiesManager.
- searchedSites: Specifies the component that PriceSliderHandler uses to determine which sites have contributed records to the current results set.
- siteManager: Specifies the component that PriceSliderHandler uses to retrieve sites from the Site repository.

The PriceSlider component also inherits a number of properties from the /atg/endeca/assembler/cartridge/handler/NavigationCartridgeHandler component, through a $basedOn property configuration. Specifically, PriceSlider inherits a reference to the /atg/endeca/assembler/cartridge/manager/NavigationState component that provides information on the shopper's current navigation state.

JSP Renderer

The store.war/cartridges/PriceSlider/PriceSlider.jsp page renders the content in the PriceSliderContentItem object. This renderer retrieves the current price slider settings, if any exist, from the PriceSliderContentItem.filterCrumb property. It also retrieves the slider end points from the PriceSliderContentItem.sliderMin and PriceSliderContentItem.sliderMax properties.

Existing Endeca Cartridges

This section describes Commerce Reference Store's use of existing Oracle Endeca cartridges.

RefinementMenu

RefinementMenu cartridges render the set of dimension value refinements that are available in the left hand rail for the current record set. This set of refinement values is driven by the dimension values that are tagged to the records themselves.

In Oracle ATG Web Commerce, applications may incorporate multiple sites and multiple catalogs. This means that each ATG category in the Category dimension may have one or more sites or catalogs associated with it. For example, Category A may be associated with Sites X and Y but not Site Z. In this case, when a customer is browsing on Sites X and Y, Category A should be included in the list of refinement values but when the customer is browsing Site Z, it should not. The `RefinementMenu` cartridge that is included out of the box with Oracle Endeca Commerce does not take site and catalog associations into account when it calculates the refinement values for the Category dimension. For this reason, the DCS module provides an extension to the `RefinementMenu` handler that filters out Category dimension values that are inappropriate for the current site and catalog context.

The DCS module implements the `atg.commerce.endeca.assembler.cartridge.handler.CategoryMenuHandler` class, which is an extension of the `com.endeca.infront.cartridge.RefinementMenuHandler` class with an override for the `process()` method. The `CategoryMenuHandler.process()` method makes an initial call to the `RefinementMenuHandler.process()` method to get the refinement values for the current record set as they would normally be returned by the Assembler. The `CategoryMenuHandler.process()` method then filters the refinement values to remove any values that are not appropriate for the current site and catalog context. To do the filtering, the `process()` method invokes the `/atg/endeca/assembler/cartridge/manager/filter/SiteFilterBuilder` and `/atg/endeca/assembler/cartridge/manager/filter/CatalogFilterBuilder` components, which are of class `atg.endeca.assembler.navigation.filter.SiteFilterBuilder` and `atg.commerce.endeca.assembler.navigation.filter.CatalogFilterBuilder`, respectively. These components calculate the set of sites and catalogs that are specified for the current request. Only those refinement values that have an association with these sites and catalogs are returned.

To do its tasks, the `/atg/endeca/assembler/cartridge/handler/RefinementMenu` component defines the following properties in the DCS module:

- `sitesPropertyName`: The name of the category repository item property that contains the list of sites associated with the category. Set by default to `category.siteId`.
- `catalogsPropertyname`: The name of the category repository item property that contains the list of catalogs associated with the category. Set by default to `category.catalogs.repositoryId`.
- `categoryDimensionName`: The name of the Category dimension. Set by default to `product.category`.
- `separator`: A reference to a `ContactFilter.separator` property. Because the `category.siteId` and `category.catalogs.repositoryId` properties can be multi-valued, their values are generated by the `ConcatFilter` component. To split the properties into their separate values, the `CategoryMenuHandler` must know which separator is being used.
- `siteFilterBuilder`: A reference to the `SiteFilterBuilder` component that should be used to calculate the set of sites that make up the site context for the current request. Set by default to `/atg/endeca/assembler/cartridge/manager/filter/SiteFilterBuilder`.
- `catalogFilterBuilder`: A reference to the `CatalogFilterBuilder` component that should be used to calculate the set of catalogs that make up the catalog context for the current request. Set by default to `/atg/endeca/assembler/cartridge/manager/filter/CatalogFilterBuilder`.

Note about the More Link

It is important to note that the Category dimension value filtering described in this section occurs *after* the MDEX has returned the dimension values and, possibly, a More link if one has been configured for the cartridge. This creates the potential for a situation where the all the returned dimension values are filtered out and only the More link gets rendered. Additionally, the More link itself may or may not be usable, depending on whether the dimension values that were identified as “More” dimension values are also filtered out. For this reason, it is important to consider whether or not configuring a More link is appropriate if you are using the extended version of the `RefinementMenu` cartridge to render the Category dimension values. Configuring a More link is safe if your application does not use multisite and has a single catalog for all shoppers. Also, if you use multisite but have a separate MDEX for each site, the More link is safe as long as all of the records in each MDEX come from the same catalog.

Note that Commerce Reference Store, which uses a single MDEX to support multiple sites, does not configure a More link for its Category dimension because of the potential for filtering issues.

HorizontalRecordSpotlight

Commerce Reference Store uses Oracle Endeca's existing `HorizontalRecordSpotlight` cartridge with some modifications described below.

Template

The XML template for the `HorizontalRecordSpotlight` cartridge is located in `<ATG10dir>/CommerceReferenceStore/Store/Storefront/deploy/cartridge_templates/HorizontalRecordSpotlight.xml`. The `title` and `seeAllLinkText` labels are not editable in Experience Manager because Commerce Reference Store uses localized values for these labels that must be computed at query time.

Also, the `name` attribute of the `PreviewProperty` sub-element of the `recordSelection` editor is set to `product.repositoryId`. Any property to be returned to the renderer must be specified in both the XML template as a `PreviewProperty` and in the `/atg/endeca/assembler/cartridge/handler/config/RecordSpotlightConfig` component's `fieldNames` property (see below for more details on this component).

```
<editors:RecordListEditor propertyName="recordSelection" label="Spotlight Records">
     <PreviewProperty name="product.repositoryId"/>
</editors:RecordListEditor>
```

Handler

The `/atg/endeca/assembler/cartridge/handler/RecordSpotlightHandler` component is of class `com.endeca.infront.cartridge.RecordSpotlightHandler`. The configuration for the `RecordSpotlightHandler` component is included in the `DAF` module and it looks like this:

```
$class=com.endeca.infront.cartridge.RecordSpotlightHandler
$scope=prototype

# Import properties
$basedOn=RecordBasedNavigationCartridgeHandler

contentItemInitializer=config/RecordSpotlightInitializer
```

Config

The `/atg/endeca/assembler/cartridge/handler/config/RecordSpotlightConfig` component, which is of class `com.endeca.infront.cartridge.RecordSpotlightConfig`, defines one property, `fieldnames`, that specifies which record fields are passed through to the content item. In this case, the field that is passed through is the `product.repositoryId` field, which allows the renderer to locate and render the actual products.

```
# Specifies which record fields will be passed through from the records to the
# resulting ContentItem
fieldNames=product.repositoryId
```

Note: Any property returned to the renderer must be specified in the `RecordSpotlightConfig.fieldNames` property and also set as a `PreviewProperty` in the XML template.

Initializer

The `/atg/endeca/assembler/cartridge/handler/config/RecordSpotlightInitializer` component, which is of class `com.endeca.infront.cartridge.RecordSpotlightInitializer`, sets the default configuration component to be used by the handler. As such, the `defaults` property in this component references the `/atg/endeca/assembler/cartridge/handler/config/RecordSpotlightConfig` component.

JSP Renderer

The `store.war/cartridges/HorizontalRecordSpotlight/HorizontalRecordSpotlight.jsp` page renders the content in the `HorizontalRecordSpotlight` content item. This content item contains a list of `com.endeca.infront.cartridge.model.Record` objects. Each `Record` object represents a product that has been returned by the handler and contains a `Map` called `attributes` that contains a value corresponding to the `product.repositoryId` key. `HorizontalRecordSpotlight.jsp` iterates over the list of `Record` objects, extracting the corresponding `product.repositoryId` values. After extracting the `product.repositoryId` values, `HorizontalRecordSpotlight.jsp` uses the `/atg/commerce/catalog/ProductLookup` component to retrieve the actual product items from the repository.

The “Featured Items” label that appears above the product spotlight row in the main content area, along with its translated versions, are stored in the `StoreText` repository in the `horizontal_record_spotlightTitle` item. The “See All” text and its translations are stored in the `horizontal_record_spotlightSeeAllText` item. Both may be edited in the Business Control Center.

ResultsList

The ATG Platform and Commerce Reference Store use Oracle Endeca’s existing `ResultsList` cartridge with some extensions to the cartridge’s handler class, described below.

The `atg.endeca.assembler.cartridge.handler.ResultsListHandler` class overrides the `com.endeca.infront.cartridge.ResultsListHandler` class and includes an additional `sorters` property of type `atg.Nucleus.ServiceMap`. The keys of this `ServiceMap` are labels for the sorting options and the values are the components that perform the actual sorting. Out of the box, the `CommerceReferenceStore` module sets the `ResultsList` component `sorters` property as follows:

```
sorters=\
  Option1=/atg/endeca/assembler/cartridge/sort/Relevance,\
  Option2=/atg/endeca/assembler/cartridge/sort/NameAscending,\
  Option3=/atg/endeca/assembler/cartridge/sort/NameDescending,\
  Option4=/atg/endeca/assembler/cartridge/sort/PriceAscending,\
  Option5=/atg/endeca/assembler/cartridge/sort/PriceDescending
```

The `atg.endeca.assembler.cartridge.handler.ResultsListHandler.setSorters()` method transforms the `sorters` `ServiceMap` into a `List` of `com.endeca.infront.cartridge.model.SortOptionConfig` components. It then passes that `List` when it calls the `com.endeca.infront.cartridge.model.SortOptionConfig.setSortOptions()` method to set the sort options. This technique of creating a `ServiceMap` and then using it to create a `List` of components is necessary because Nucleus cannot set `Lists` of components directly.

SearchBox

In Commerce Reference Store, there is an instance of the `SearchBox` cartridge called `Search Box`, located in `/content/Shared/Global Search Configuration/Search Box` in Experience Manager. The `<crs:pageContainer.tag>` retrieves the contents of the `Search Box` cartridge to render the search box that appears in the header portion of each page. To do this, the `<crs:pageContainer.tag>` calls the `/atg/endeca/assembler/droplet/InvokeAssembler` servlet bean and passes in the full path to the cartridge:

```
<dsp:droplet name="InvokeAssembler">
  <dsp:param name="contentCollection"
             value="/content/Shared/Global Search Configuration/Search Box"/>
  <dsp:oparam name="output">
    <dsp:getvalueof var="searchBox"
                    vartype="com.endeca.infront.assembler.ContentItem"
                    param="contentItem" />
  </dsp:oparam>
</dsp:droplet>
```

Other Existing Endeca Cartridges

The `DAF.Endeca.Assembler` module includes configuration, in the `/atg/endeca/assembler/cartridge/handler/` Nucleus folder, for additional existing Oracle Endeca cartridge handlers from the `com.endeca.infront.cartridge` package. The table below provides a list of these cartridge handlers (note that some of these cartridges handlers are used by Commerce Reference Store and some are not):

Nucleus Component	**Cartridge Handler Class**
`Breadcrumbs`	`BreadcrumbsHandler`
`ContentInclude`	`ContentIncludeHandler`
`ContentSlot`	`ContentSlotHandler`
`DimensionSearchResults`	`DimensionSearchResultsHandler`
`NavigationCartridgeHandler`	`NavigationCartridgeHandler`
`NavigationContainer`	`NavigationContainerHandler`
`RecordBasedNavigationCartridgehandler`	`RecordBasedNavigationCartridgeHandler`
`RecordDetail`	`RecordDetailsHandler`
`RecordSpotlight`	`RecordSpotlightHandler`
`RefinementMenu`	`RefinementMenuHandler`

Nucleus Component	Cartridge Handler Class
`SearchAdjustments`	`SearchAdjustmentsHandler`

Registering Cartridges with the NucleusAssemblerFactory

Any time a cartridge handler has a name that does not match its cartridge type exactly, it must be registered in the `/atg/endeca/assembler/NucleusAssemblerFactory.handlerMapping` property. For example, The `CategoryHeaderBanner` component is registered in the `NucleusAssemblerFactory.handlerMapping` property as the handler for cartridges with an ID of `HeaderBanner-ATGCategory`. The following mappings cartridge type and handler are defined in Commerce Reference Store

```
handlerMapping+=\
  ContentSlot-Main=/atg/endeca/assembler/cartridge/handler/ContentSlot,\
  ContentSlot-Header=/atg/endeca/assembler/cartridge/handler/ContentSlot,\
  ContentSlot-Secondary=/atg/endeca/assembler/cartridge/handler/ContentSlot,\
  PageSlot=/atg/endeca/assembler/cartridge/handler/ContentSlot,\
  PromotionalContent-ATGSlot=\
     /atg/endeca/assembler/cartridge/handler/PromotionalContentSlot,\
  PromotionalContent-ATGTargeter=\
     /atg/endeca/assembler/cartridge/handler/PromotionalContentTargeter,\
  ProductSpotlight-ATGSlot=\
     /atg/endeca/assembler/cartridge/handler/ProductSpotlightSlot,\
  ProductSpotlight-ATGTargeter=\
     /atg/endeca/assembler/cartridge/handler/ProductSpotlightTargeter,\
  HeaderBanner-ATGCategory=\
     /atg/endeca/assembler/cartridge/handler/CategoryHeaderBanner,\
  DimensionSearchAutoSuggestItem=\
     /atg/endeca/assembler/cartridge/handler/DimensionSearchResults,\
  ProductSpotlight-ATGCategoryRelatedProducts=\
     /atg/endeca/assembler/cartridge/handler/CategoryRelatedProducts,\
  ProductSpotlight-ATGCategoryRecommendations=\
     /atg/endeca/assembler/cartridge/handler/CategoryRecommendations,\
  HorizontalRecordSpotlight=\
     /atg/endeca/assembler/cartridge/handler/RecordSpotlight,\
  ProductList-ATGCategoryChildren=\
     /atg/endeca/assembler/cartridge/handler/ProductListCategoryChildren
```

Note: The `PriceSlider` cartridge handler has the same name as the `PriceSlider` cartridge, so it does not need to be registered in the `handlerMapping` property.

Using Sites and ATG User Segments as Triggers in Experience Manager

Experience Manager gives business users the ability to choose layouts and content based on a shopper's location within an application. This choice is embodied in prioritized triggering rules. One of the criteria that can be incorporated into triggering rules is the Endeca user segments. The ATG-Endeca integration takes advantage of the Endeca user segment functionality to enable the use of sites, site groups, and ATG user segments in Experience Manager triggers. To incorporate this functionality, the following happens:

- A set of Endeca user segments that correspond to the ATG sites, site groups, and user segments are manually added in Experience Manager. This allows the ATG sites, site groups, and user segments to be used in triggers configured by business users.
- Experience Manager business users configure triggering rules using the manually added user segments.
- Every time the ATG Platform calls the Assembler, it passes a list of ATG user segments and site context information (the current site and its site groups) for the current request and shopper.
- The passed information allows the Assembler to return the correct content for the request, based on the configuration set in Experience Manager.

The following sections provide more detail on these general steps.

Adding Sites and Segments with Experience Manager

The Endeca user segments that are available as trigger options in Experience Manager come from a manually administered list. Business users must manually add new sites, site groups, and segments with Endeca before referring to them in triggering rules.

When adding the new user segments, business users must use the correct syntax. ATG user segment names, site IDs, and site group IDs are unique within type but can, in theory, collide across types. Because these names are passed as a single list to Experience Manager, a prefix scheme is used to ensure a unique identifier for each segment passed. For more details on these prefixes, see the next section, Constructing the Segment List (page 140).

Constructing the Segment List

Both the `DAF` and `DPS` modules contain logic for adding entries to the segment list that is sent to the Assembler. The `DAF` module is responsible for adding sites and site groups to the list, while DPS adds ATG user segments to the list.

The `DAF` module includes the request-scoped `/atg/endeca/assembler/cartridge/manager/user/LiveUserState` component. This component is of class `atg.endeca.assembler.navigation.LiveUserState`. The `LiveUserState` class is an extension of the `com.endeca.infront.navigation.UserState` class that overrides the `getUserSegments()` method with a call to a `LiveUserState.computeSegments()` method that computes the list of sites and site groups to be added to the segment list. Segment names are added to the `userSegments` property via the `addUserSegments()` method.

The `DAF` module contains the following configuration for the `LiveUserState` component:

```
sitePrefix=site
siteGroupPrefix=sitegroup
prefixDelimiter=.
```

This configuration specifies that, when the `LiveUserState` component adds a site to the segment list, it prefixes the site's ID (as defined in the Site Manager UI) with `site.`, for example:

```
site.MySite
```

A similar situation exists for site groups. Site group IDs are prefixed with `sitegroup.`, for example:

```
sitegroup.MySiteGroup
```

It is important to remember that any manually added segments in Experience Manager must also follow these same naming conventions.

The `DPS` module extends the `LiveUserState` class with an override to the `computeUserSegments()`method that retrieves the current shopper's ATG user segments and adds them to the segment list as well. The `DPS` module adds a `userSegmentPrefix` property to the `LiveUserState` component's configuration. Out of the box, this property is null, meaning no prefix is added to ATG user segment names before they are added to the segment list.

Note: Commerce Reference Store includes a special user segment, `CategoryOnly`, that functions as the trigger mechanism for category pages. This user segment is added programmatically and does not appear in the ATG user segments in the Business Control Center. For more information, see the ATG Commerce Reference Store Overview (page 1).

10 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. More sophisticated search techniques can be used to navigate the catalog based on criteria such as price, color, or manufacturer.

Search in Commerce Reference Store is implemented through integration with Oracle Endeca Commerce. This chapter discusses the search-related features in Commerce Reference Store and how they are implemented. More detailed information about the cartridges used in Commerce Reference Store to access Endeca-driven content is found in the *Using Cartridges* (page 115) chapter.

This chapter includes the following topics:

Indexing the Product Catalog

The indexing process for Commerce sites is configured through the `/atg/commerce/search/ProductCatalogOutputConfig` and other related components described in the *ATG Endeca Integration Guide*. These components are configured to create Endeca data, dimension value, and schema records, which are submitted to Oracle Endeca Commerce for indexing.

The XML definition file used for the `ProductCatalogOutputConfig` component in Commerce Reference Store is an XML combination of the definition file included in the `DCS.Endeca.Index` module, the `DCS.Endeca.Index.SKUIndexing` module, and Commerce Reference Store-specific extensions added in the `Store.Endeca.Index` module. The Commerce Reference Store extensions add Commerce Reference Store-specific catalog properties to the index. For internationalized sites, the `Store.Endeca.International.Index` module also adds configuration for including content in multiple languages.

Each product and SKU in the Commerce Reference Store catalog repository has a `siteIds` property whose value is a `Set` of the sites the item 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 records generated for a SKU include a separate `sku.siteId` property value for each site listed in the `siteIds` property. For example, the records for a SKU found on ATG Store US and ATG Store Germany include:

```
<PROP NAME="sku.siteId">
  <PVAL>
    storeSiteUS
  </PVAL>
</PROP>
<PROP NAME="sku.siteId">
  <PVAL>
    storeSiteDE
  </PVAL>
</PROP>
```

This information is used to determine which results to return when a query is issued from a site, as discussed in Implementing a Multisite Search Control (page 147).

Guided Navigation

As discussed in the *Using Cartridges* (page 115) chapter, a category page is displayed when a query includes only a Category dimension value. If a query includes either a search term or a value for a dimension other than Category, Commerce Reference Store displays a browse page (search results). This is true even if the query also includes a Category dimension value.

Both pages display available facets in the left column. Selecting a facet value on either page refines the search, and the results are then displayed on a browse page.

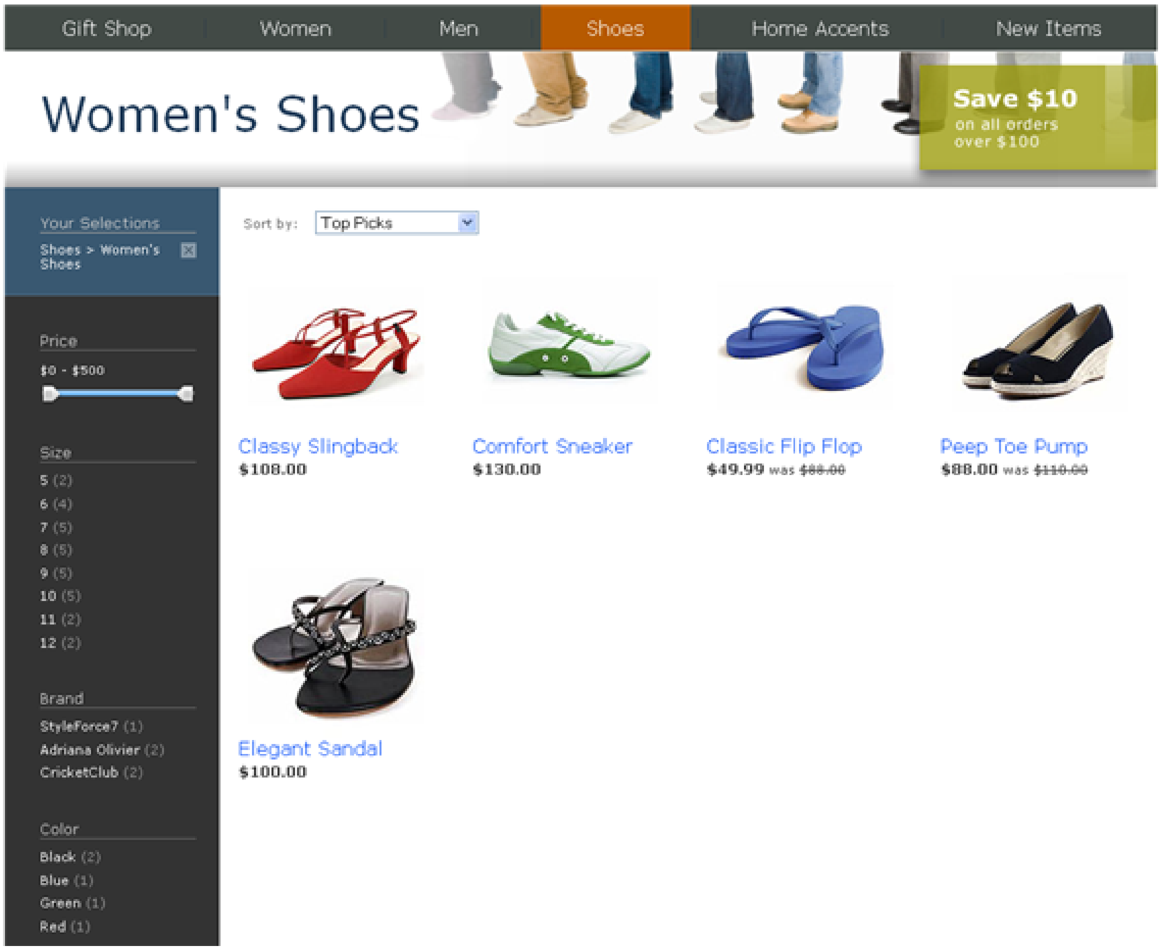
Commerce Reference Store defines a number of facets for guided navigation, based on dimension values stored in the MDEX. The table below lists the facets, the names of the Endeca dimensions they correspond to, and the ATG properties that the dimensions are created from. Note that the ATG property names are relative to the `product` item type, which is the top-level item in the definition file of the `ProductCatalogOutputConfig` component.

Facet Name	**Endeca Dimension Name**	**ATG Property Name**
Size	`clothing-sku.size`	`childSKUs.size`
Color	`clothing-sku.color`	`childSKUs.color`
Wood Finish	`furniture-sku.woodFinish`	`childSKUs.woodFinish`
Brand	`product.brand`	`brand`
Feature	`product.features.displayName`	`features.displayName`
Category	`product.category`	`parentCategories`

The Category facet (the `product.category` dimension) is hierarchical. All of the other facets are non-hierarchical dimensions.

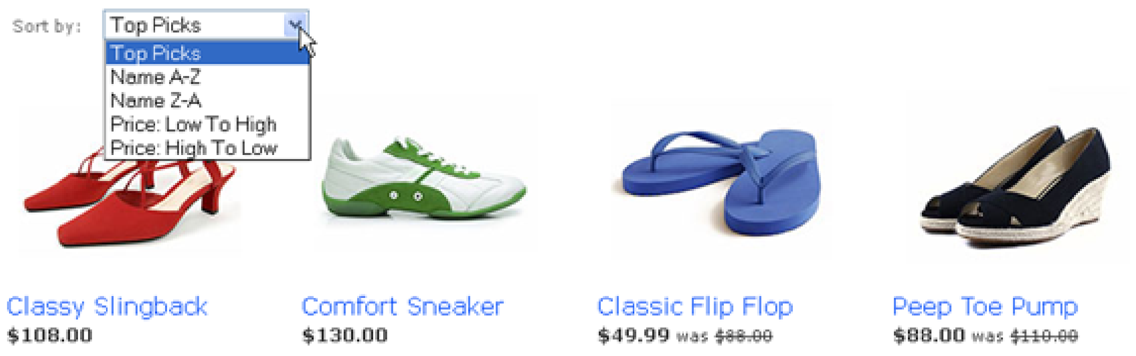
Displaying the Facets and Results

As a customer navigates the product catalog, Commerce Reference Store displays facet values and selections on the left column of category pages and browse pages, and the products that match the facet selections in the main body of the page. In addition, the left column displays a slider for filtering results by price. For example:



Sorting Options

When Commerce Reference Store displays a category page, the list of child products is displayed by the `ProductList-ATGCategoryChildren` cartridge. As discussed in the *Using Cartridges* (page 115) chapter, this cartridge retrieves the list of products from the ATG catalog repository, not the MDEX. By default, the products are sorted using the Top Picks option, which displays them in the order they appear in the category in ATG Merchandising:



The other sorting options available on the drop-down menu are implemented using the `/atg/store/sort/RangeSortDroplet` servlet bean.

When Commerce Reference Store displays a browse page (search results), the list of products is displayed by the `ResultsList` cartridge. In this case, the records associated with the appropriate products are retrieved from the MDEX. The same menu of five sorting options appears on this page, but since the data is coming from the MDEX, the options are implemented in Oracle Endeca Commerce using instances of the `com.endeca.infront.cartridge.model.SortOptionConfig` class. The default sorting option for the page can be specified by editing the `ResultsList` cartridge in Experience Manager.

The Top Picks option (which appears in Experience Manager as Default) works as follows:

- If the current query does not include a search term (that is, it includes only dimension values), sort by values of the `sku.margin` property, in descending order.
- If the current query includes a search term, apply the selected relevance ranking.

Relevance Ranking

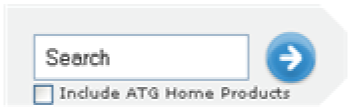
Relevance ranking is a formula used by Oracle Endeca Commerce for ordering search results based on the number of times a search term is found in a record, which properties it is found in, how exact the match is, etc. For example, based on the ranking strategy being used, a record might be considered more relevant if the search term is found multiple times in certain key properties than if it is found only once in a less critical property.

Commerce Reference Store provides two relevance ranking options that can be selected in Experience Manager, labeled Margin Bias and Price Bias. These options are fundamentally similar, and differ only in which property is used to break ties. If two records have the same relevance ranking scores, the one with the higher value for `sku.margin` appears first if Margin Bias is used, while the one with the higher value for `sku.activePrice` appears first if Price Bias is used.

Note that to apply a relevance ranking, a search interface must be used. Therefore, Commerce Reference Store includes a search interface that configures various search behaviors. See the Oracle Endeca Commerce documentation for more information about relevance ranking and search interfaces.

Implementing a Multisite Search Control

Commerce Reference Store 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, on the ATG Store US site, the control looks like this:



This search control appears on virtually all Commerce Reference Store pages. It is created by the `Search Box` cartridge, which is placed on the page using the `/atg/endeca/assembler/droplet/InvokeAssembler` servlet bean. (This servlet bean is invoked by the `<crs:pageContainer>` tag, as discussed in the *Using Cartridges* (page 115) chapter. `InvokeAssembler` enables cartridges to be included on ATG-driven pages.) The renderer for the `Search Box` cartridge, `store.war/cartridges/SearchBox/SearchBox.jsp`, creates the text input field and checkboxes for specifying the sites to search. The control includes a checkbox for each site that shares a shopping cart with the current site.

The `SearchBox.jsp` renderer also invokes the `/atg/endeca/assembler/SearchFormHandler` component, which is of class `atg.projects.store.assembler.SearchFormHandler`. This form handler determines the sites to search based on which checkboxes are selected, and stores the site IDs in the `/atg/endeca/assembler/SearchedSites` component. The `/atg/endeca/assembler/cartridge/manager/filter/SiteFilterBuilder` component then creates a record filter to restrict the records returned to only those associated with those sites.

See the *ATG Endeca Integration Guide* for more information about record filtering.

Handling Price Data

Commerce Reference Store assigns a pair of price lists to each customer, with one price list containing the list prices for each SKU in the catalog, and the other price list containing sale prices for the SKUs that are currently on sale (and empty values for SKUs that are not on sale). If a SKU has a sale price, the sale price is selected; otherwise the list price is selected. The resulting value is referred to as the *active price*.

Price lists are assigned to a customer through the `priceList` and `salePriceList` properties of the customer's user profile. Commerce Reference Store sets these profile properties based on the `defaultListPriceList` and `defaultSalePriceList` properties of the current site's `siteConfiguration` item.

For example, on the `siteConfiguration` item for the ATG Store US site, the value of the `defaultListPriceList` property is `listPrices`, and value of the `defaultSalePriceList` property is `salePrices`. Therefore, when a customer visits this site, the profile properties are set as follows:

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

The following logic is used to determine the prices used for guided navigation:

- When a SKU is indexed, a separate record is created for each price list pair.

- In each record, the `product.priceListPair` property is set to the price list pair the record applies to, and the value of the `sku.activePrice` property is set based on the logic described above.
- After the records are generated and indexed, a record filter is used during querying that returns only the records associated with the price list pair for the current customer. The `sku.activePrice` values are used for guided navigation..

See the *ATG Endeca Integration Guide* for more information about how this logic is implemented.

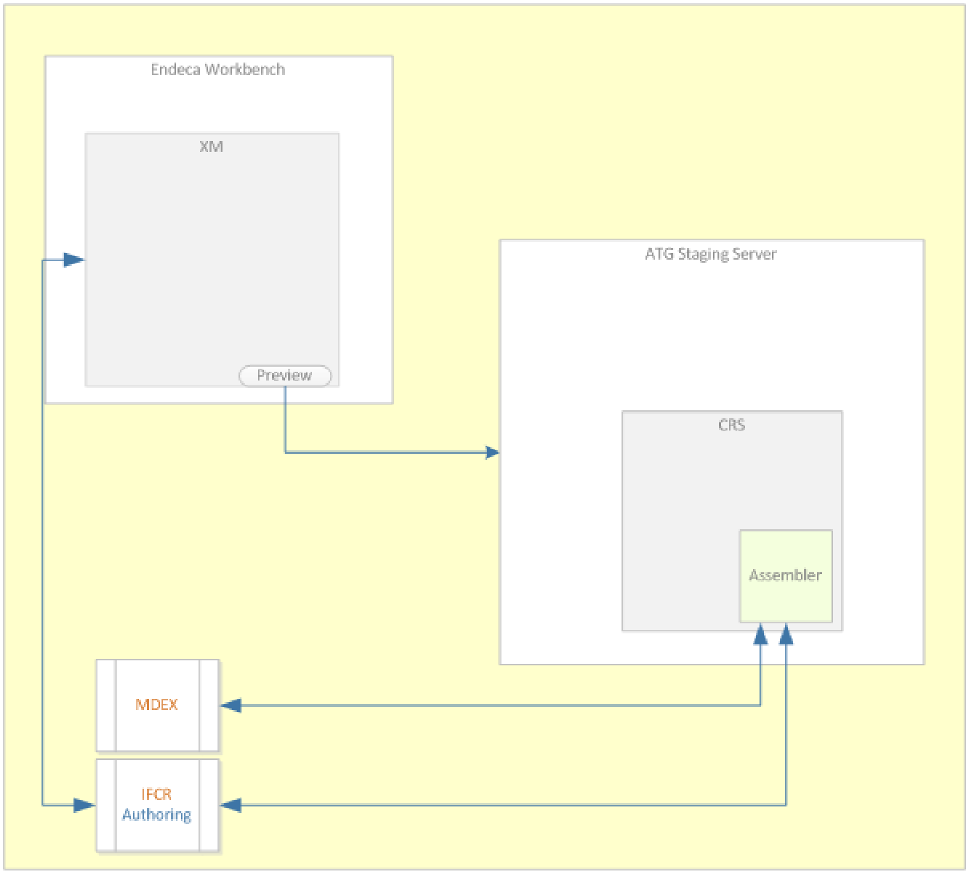
11 Preview

Oracle ATG Web Commerce and Oracle Endeca Commerce both provide mechanisms that allow merchandisers to preview site changes from an authoring environment, before these changes are committed to production sites. Commerce Reference Store includes configuration that supports both of these preview mechanisms.

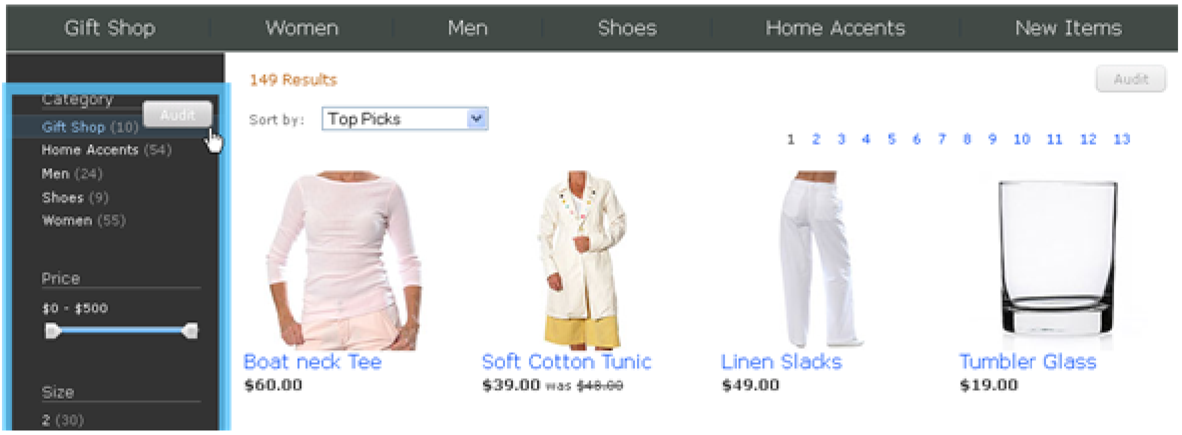
This chapter describes Commerce Reference Store support for preview. It includes these sections:

Endeca Experience Manager Preview

Experience Manager preview for Commerce Reference Store allows you to make changes to Endeca-driven content in Experience Manager and view your changes on a staging server. The architecture looks like this:



Choosing preview from within Experience Manager launches a browser window that displays the Commerce Reference Store application. An Audit button appears beside each cartridge. For example, this page includes Audit buttons for the Guided Navigation and Results List cartridges:



Clicking an Audit button opens a popup that displays information about the associated cartridge.

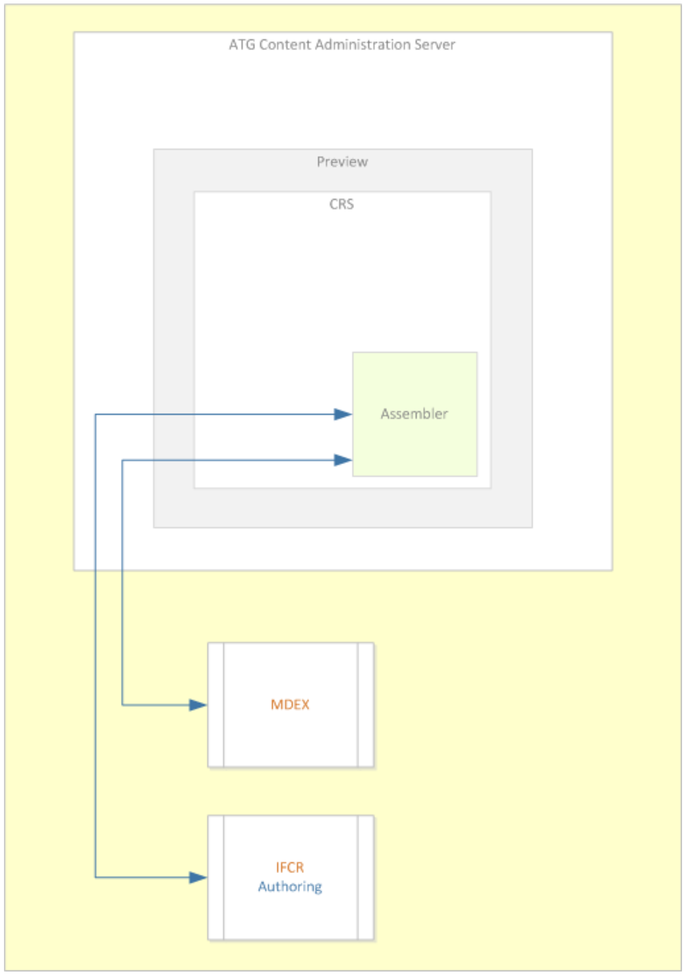
See the Oracle Endeca Commerce documentation for more information about Experience Manager preview.

Enabling Support for Experience Manager Preview

The `endecapreview` ATG configuration layer includes configuration that enables Commerce Reference Store and other ATG applications to work appropriately with Experience Manager preview. CIM gives you the option of enabling this configuration layer on the staging server.

ATG Business Control Center Preview

ATG Business Control Center preview allows you to preview and modify ATG content displayed on Commerce Reference Store pages, including the ATG content on Endeca-driven pages (for example, products displayed on category and browse pages). The architecture looks like this:



See the *ATG Business Control Center User's Guide* for information about how to launch Business Control Center preview. Further information is available in the *ATG Merchandising Guide for Business Users*.

Configuring Business Control Center Preview

Business Control Center preview is enabled through the `preview` ATG configuration layer. This layer includes configuration that enables the preview application to include categories and products that have not yet been indexed. CIM enables this configuration layer on the Content Administration server if you select Preview as an add-on.

Features and Limitations of Preview

This section summarizes features and limitations of using Experience Manager preview and Business Control Center preview with Commerce Reference Store.

Project Context

When you are working in a project in the Business Control Center, you must deploy your changes (typically to a staging environment) in order for Experience Manager preview to reflect them. Experience Manager does not have access to content in a project that has not been deployed.

New Categories and Products

When you create a new category in the Business Control Center, the category name appears in the preview menu bar. You can navigate to the category page for the new category, even if the change has not been deployed and the MDEX has not been updated.

When previewing in the Business Control Center, you can add new products to an existing category page and preview the existing category with the new products. However, they do not affect facet values or counts until the MDEX is updated.

To access new products in Experience Manager, you must first deploy your project changes in the Business Control Center and update the MDEX.

When you add a value to a non-hierarchical dimension (for example, Brand or Color) either on a new product or by modifying an existing product, you must deploy your changes and update the MDEX before the new dimension value will be available.

New Site

When creating a new site in the Business Control Center, you must deploy your project and perform a baseline update of the MDEX before you are able to preview the new site. In addition, to be able to view the site in Experience Manager, you must create a segment in Experience Manager with the same name as the new site.

User Segments

To be able to associate rules in Experience Manager with an ATG user segment, you must manually add a segment of the same name in Experience Manager.

When previewing from Experience Manager, the list of user segments you select (including segments representing sites) is passed to the Assembler. The behavior of Experience Manager triggering rules reflects the segment selections.

When previewing from Experience Manager, the segment selections from Experience Manager are not used to trigger ATG personalization features such as targeters. Rather, the segments associated with the current ATG user profile (either an anonymous profile or a registered user) trigger ATG personalization behavior.

When previewing from the Business Control Center, the segments associated with the ATG user profile (either an anonymous profile or a registered user) trigger both ATG personalization behavior and Experience Manager rules.

Site Context

To set the site context in Experience Manager preview, select the segment representing the site. If you do not select a site when launching preview from Experience Manager, the ATG Store site is used by default.

If you change the site segment selected in Experience Manager, preview responds by displaying the newly selected site.

Date and Time

When previewing in Experience Manager, you can select a date and time, and preview will display a view reflecting that date and time.

When previewing in the Business Control Center, you can select a date and time, and preview will display a view reflecting that date and time.

Visual Merchandising

When previewing from the Business Control Center, you can right click on the category menu to activate the Visual Merchandising context menu for the selected category asset. You can also right click on categories listed in guide navigation area (left column) of category and search results pages to activate the Visual Merchandising context menu.

When previewing from the Business Control Center, you can right click on the following product displays to activate the Visual Merchandising context menu for the selected product assets:

- Products displayed under Featured Items
- Products displayed under We Also Suggest
- Products displayed in the product grid on the category page
- Products displayed in search results

See the Tags for Visual Merchandising (page 35) section of the *Pages and Navigation* (page 21) chapter for information about Commerce Reference Store JSPs that are tagged for use with Visual Merchandising.

12 Using Fluoroscope to Inspect Pages

Commerce Reference Store provides a user interface for accessing Fluoroscope, a tool for viewing site HTML pages that reveals key JSP elements involved in rendering those pages, such as page includes, servlet beans, scenario events and actions, and form fields.

This chapter describes how to use Fluoroscope with Commerce Reference Store pages. It includes the following sections:

Enabling Fluoroscope

To enable Fluoroscope, include the `Store.Fluoroscope` module in your storefront EAR file.

Fluoroscope works through *sensors*, which detect specific elements (for example, a scenario action or a servlet bean) encountered in a JSP when rendering an HTML page. These sensors fire events that are detected by event listeners, which then insert data from the events into the HTML markup for the page. The Commerce Reference Store Fluoroscope UI provides tools for examining this data.

Commerce Reference Store makes four Fluoroscope sensors available:

- `DropletTagSensor` -- Detects the presence of `<dsp:droplet>` tags.
- `PageTagSensor` -- Detects the presence of `<dsp:include>` tags.
- `FormTagSensor` -- Detects the presence of `<dsp:form>` tags.
- `ScenarioManagerSensor` -- Detects scenario events and actions.

Limitations: Fluoroscope is a development tool that should be used only in a development environment. Do not include the `Store.Fluoroscope` module in a production environment, as it has a significant impact on performance.

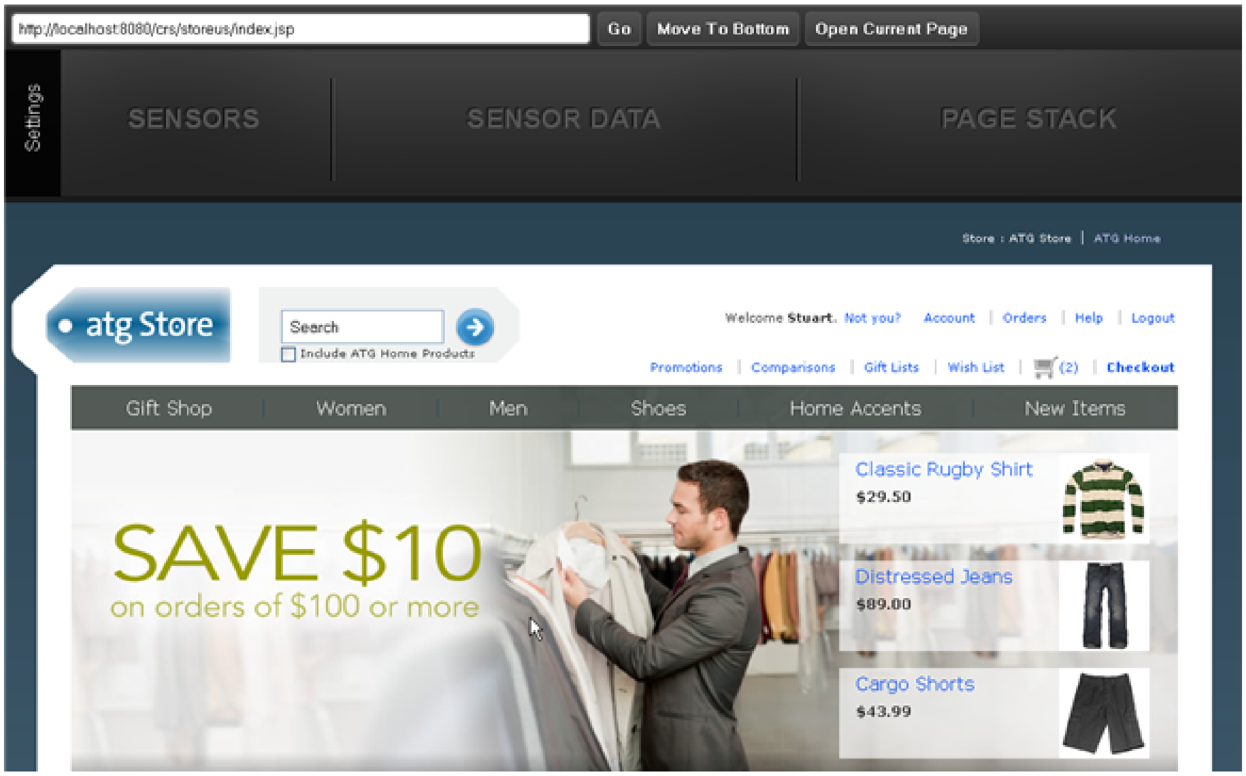
Fluoroscope requires Internet Explorer 9 or Firefox, with JavaScript enabled. Do not use Internet Explorer 9 in Compatibility View.

Accessing Fluoroscope

You can access Fluoroscope through the following URL:

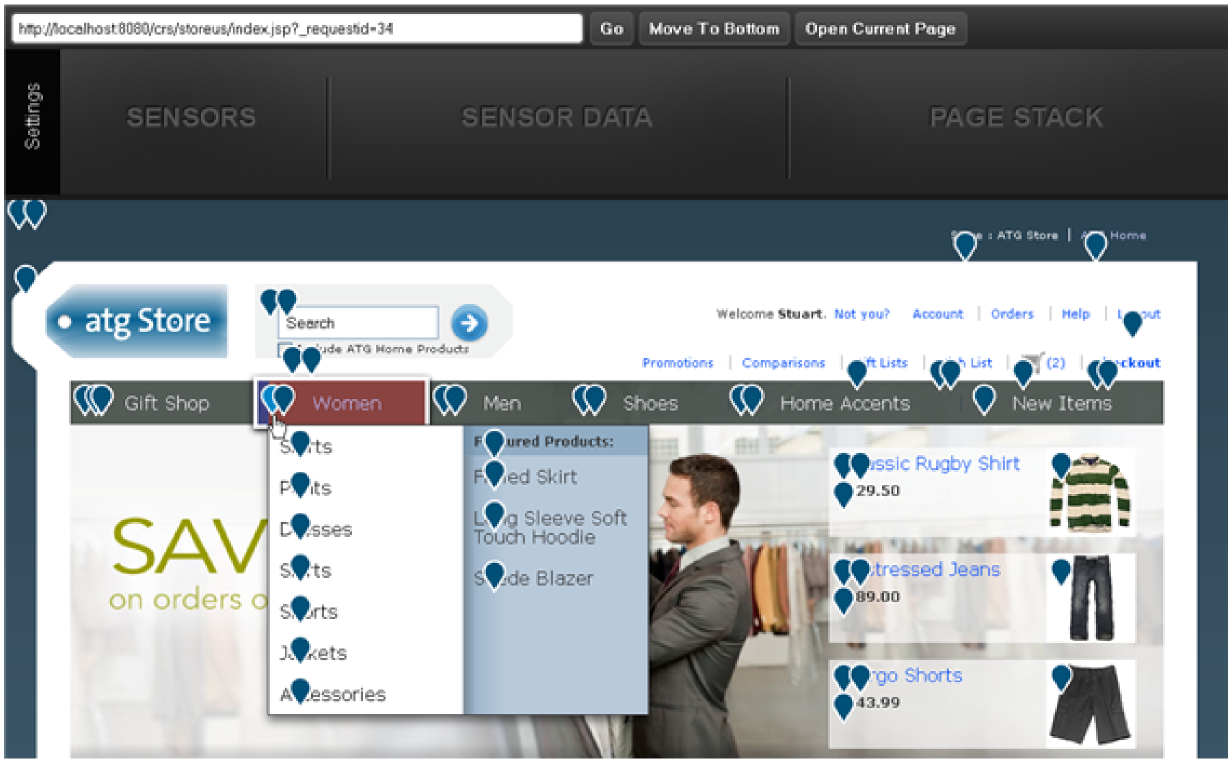
```
http://hostname:port/inspect
```

This displays the home page of ATG Store, but with a toolbar at the top for configuring the behavior of Fluoroscope and displaying information about elements in the page:


http://localhost:8080/crs/storeus/index.jsp
Go
Move To Bottom
Open Current Page
Settings
SENSORS
SENSOR DATA
PAGE STACK
Store : ATG Store | ATG Home
atg Store
Search
Include ATG Home Products
Welcome Stuart. Not you? Account | Orders | Help | Logout
Promotions | Comparisons | Gift Lists | Wish List | (2) | Checkout
Gift Shop
Women
Men
Shoes
Home Accents
New Items
SAVE $10
on orders of $100 or more
Classic Rugby Shirt
$29.50
Distressed Jeans
$89.00
Cargo Shorts
$43.99

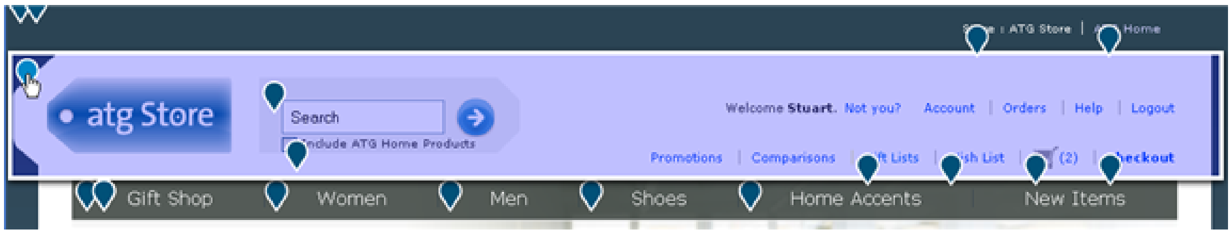
To enable the sensors, hover the cursor over the Setting area on the left edge of the toolbar. This reveals a set of checkboxes for the four sensors described above. Select the checkboxes for the sensors that you want to enable, and then click Save.

Once you have enabled one or more sensors, the page displays action markers at the positions in the page where event data from those sensors has been embedded in the HTML markup:

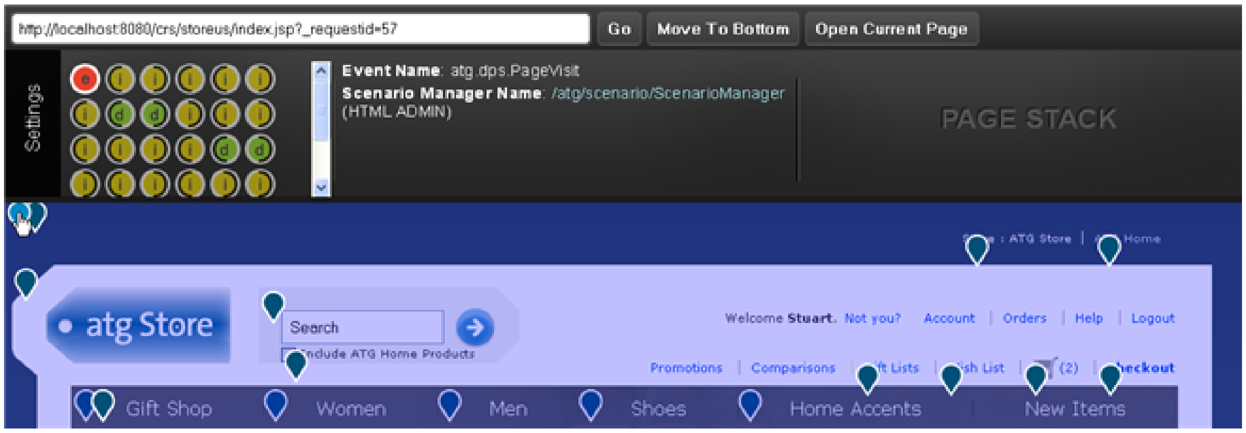


Viewing Event Information

When you hover your cursor over an action marker, a blue overlay appears on the area of the page that the action marker applies to. For example:



When you click on an action marker, the left side of the toolbar displays a list of the events associated with that marker. For example:


http://localhost:8080/crs/storeus/index.jsp?_requestid=57
Go
Move To Bottom
Open Current Page
Settings
PAGE STACK
atg Store
Search
Include ATG Home Products
Welcome Stuart. Not you? Account | Orders | Help | Logout
Promotions | Comparisons | Gift Lists | Wish List | (2) | Checkout
Gift Shop
Women
Men
Shoes
Home Accents
New Items

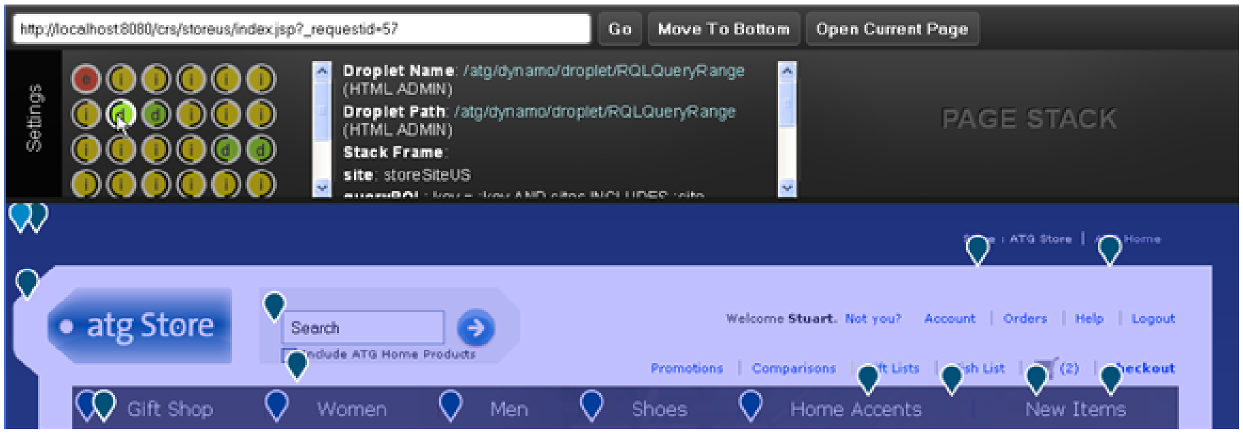
Event Name: atg.dps.PageVisit
Scenario Manager Name: /atg/scenario/ScenarioManager (HTML ADMIN)

The icons for the events may include:

- ⓓ and ⓓ -- Opening and closing `<dsp:droplet>` tags
- ⓘ and ⓘ -- Opening and closing `<dsp:include>` tags
- ⓕ and ⓕ -- Opening and closing `<dsp:form>` tags
- ⓔ -- Scenario event
- ⓐ -- Scenario action

The order in which the icons are displayed reflects their nesting relationship, which can be quite complex (for example, multiple levels of nested servlet beans containing multiple levels of page includes).

The central area of the toolbar initially displays information associated with the first icon in the list. You can display information about a different icon by clicking on it. For example:


http://localhost:8080/crs/storeus/index.jsp?_requestid=57
Go
Move To Bottom
Open Current Page
Settings
PAGE STACK
atg Store
Search
Include ATG Home Products
Welcome Stuart. Not you? Account | Orders | Help | Logout
Promotions | Comparisons | Gift Lists | Wish List | (2) | Checkout
Gift Shop
Women
Men
Shoes
Home Accents
New Items

Droplet Name: /atg/dynamo/droplet/RQLQueryRange (HTML ADMIN)
Droplet Path: /atg/dynamo/droplet/RQLQueryRange (HTML ADMIN)
Stack Frame:
site: storeSiteUS

The sections that follow summarize the data displayed for each type of Fluoroscope event.

Servlet Beans

For each `<dsp:droplet>` tag, Fluoroscope displays:

- The Nucleus pathname of the servlet bean. This pathname is a hyperlink; clicking it displays the page for the servlet bean in the Component Browser of the Dynamo Server Admin.
- Information about the parameters passed to the servlet bean.

Page Includes

For each `<dsp:include>` tag, Fluoroscope displays:

- The current context root, the servlet path and filesystem path of the JSP, and the request URI.
- The page parameters passed to the JSP.
- A list of the pathnames of the JSPs in the page stack. Each entry in the list is a hyperlink. Hovering over a link displays an overlay on the area of the page that the JSP represents; clicking a link displays information about that JSP on the right side of the toolbar.

Forms

For each `<dsp:form>` tag, Fluoroscope displays:

- The form name and context.
- The names of all of the form fields.
- A list of the Nucleus component properties bound to form fields.

Scenarios

For each scenario event, Fluoroscope displays the event name (the `JMSType`) and the Nucleus pathname of the Scenario Manager for the event.

For each scenario action, Fluoroscope displays:

- The action name.
- The Nucleus pathname of the Scenario Manager for the action.
- A list of the parameters passed to the action.
- A list of the current action context parameters.

Navigation

You can use normal site navigation within Fluoroscope. For example, you can select a category from the menu bar, and the category page for that category will be displayed, along with the Fluoroscope markers for that page.

The Fluoroscope toolbar includes a field where you can enter the URL for a page on the site. When you click Go, Fluoroscope displays that page within the Fluoroscope user interface.

The toolbar also includes an Open Current Page button. Clicking this button displays the current page, outside of Fluoroscope, in a separate browser window or tab.

13 Commerce Service Center Integration

This chapter describes how to integrate Commerce Reference Store with Oracle ATG Web Commerce Service Center, a customer-service application used by call center agents to assist customers on commerce sites. It discusses how to set up Commerce Service Center so agents can perform support tasks for Commerce Reference Store storefronts. In addition, it describes how to configure Commerce Reference Store to work with the integration between Commerce Service Center and Oracle Live Help On Demand, which enables Commerce Reference Store customers to initiate phone calls with Commerce Service Center agents.

This chapter includes the following sections:

Running Commerce Service Center with Commerce Reference Store

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

The `Store.DCS-CSR` module adds custom configuration to Commerce Service Center for working with Commerce Reference Store. It also includes several landing page handler components that determine which site page is displayed on the agent's desktop when a customer clicks a Click to Call link. The *ATG Commerce Service Center Installation and Programming Guide* includes information about configuring landing page handler components.

Integrating Commerce Reference Store with Click to Call

Commerce Reference Store pages can incorporate links to Click to Call. When a customer clicks one of these links, a popup window prompts for the customer's phone number. When the phone number is submitted, Click to Call initiates a call between a Commerce Service Center agent and the customer, and displays the customer's order information and profile on the agent's desktop.

To configure Commerce Reference Store to use Click to Call, include the `DCS.ClickToConnect` module in your storefront EAR file.

To access Click to Call, you need an Oracle Live Help On Demand account. You use this account to configure rules that control where links are placed and when and how they are displayed. Commerce Reference Store includes a sample read-only Click to Call account that is preconfigured to work with the Commerce Reference Store sites to demonstrate Click to Call functionality. When you configure Commerce Reference Store using CIM and include the Click to Call integration, you are given the option of selecting this account. If you do, CIM sets the `accountId` property of the `/atg/clicktoconnect/Configuration` component to the ID of the sample account.

Note that if you build your own site based on Commerce Reference Store and want to use Click to Call, you need to obtain a Click to Call account and use it to configure the rules for your site. You must also set properties of the `/atg/clicktoconnect/Configuration` component for your site to use your account.

When a customer accesses a Commerce Reference Store page, Click to Call links are rendered through the following process:

- A JSP renders HTML `<div>` elements to serve as placeholders for the links. The type of link to render is specified through the `id` attribute of the `<div>`.
- The ADC processor pipeline inserts calls to the Click to Call JavaScript libraries in the `<head>` tag of the page, and variables used by the libraries in the body of the page.
- When the page loads, the JavaScript calls inserted by the pipeline are executed. The JavaScript libraries are loaded, and they execute functions that create the Click to Call links.

Inserting <div> Elements

Click to Call links appear on many Commerce Reference Store pages, including product detail pages, the order history page, the shopping cart page, and the Contact Us page. Different pages are configured to use different images as link icons.

Links are associated with `<div>` elements, which are inserted in Commerce Reference Store pages by the `store.war/navigation/gadgets/clickToCallLink.jsp` gadget:

```
<dsp:page>
  <dsp:importbean bean="/atg/dynamo/droplet/ComponentExists"/>
  <%--
    If the Click To Call feature is disabled, this entire div can be safely
    bypassed. We can test if Click To Call is installed by the existence
    of the clicktoconnect Configuration component.
  --%>
  <dsp:droplet name="ComponentExists" path="/atg/clicktoconnect/Configuration">

    <dsp:oparam name="true">
      <%--
        In order to display the ClickToCall button we need to add an anchor in the
        place we want the ClickToCall button to be rendered. This anchor is the
        HTML div below. Its id attribute must match the 'Relative to Layer ID'
        property of the WebCare link
      --%>
      <dsp:getvalueof var="pageName" param="pageName"/>
      <div id="atg_store_c2c_${pageName}">&nbsp;</div>
    </dsp:oparam>
  </dsp:droplet>
</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 renders the `<div>` element and sets the `id` attribute to a value that indicates the page it appears on.

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

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

When the Click to Call link is created, the `<div>` `id` is used to look up the link and its attributes in WebCare.

Inserting Calls to the Click to Call JavaScript Libraries

The ADC processor pipeline is responsible for inserting calls to the Click to Call JavaScript libraries. To enable the ADC processor pipeline, 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 ADC processor pipeline. The filter is part of the `ADC` module, which is automatically included in the assembled EAR file when you specify the `DCS.ClickToConnect` module.

The processor pipeline inserts in the `<head>` tag of the HTML page calls to scripts for importing the Click to Call JavaScript libraries:

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

In addition, the pipeline inserts a script in the <body> tag of the page that specifies variables that are used by Click to Call. For example:

```
<script type="text/javascript">
<!--
if(ATGSvcs != null) {
  ATGSvcs.setUOID(200106298304);
}
var _atg_estara_call_token = "c2c4.5.0.SOYqEhmXS70R3LhAt1hmUPJRYXg";
var _atg_estara_locale = "en_US";
var _atg_estara_site = "storeSiteUS";
//-->
</script>
```

The _atg_estara_call_token variable is used to associate the customer's session with the information displayed to the customer service agent. The other variables make it possible to vary Click to Call behavior based on the current locale and site.

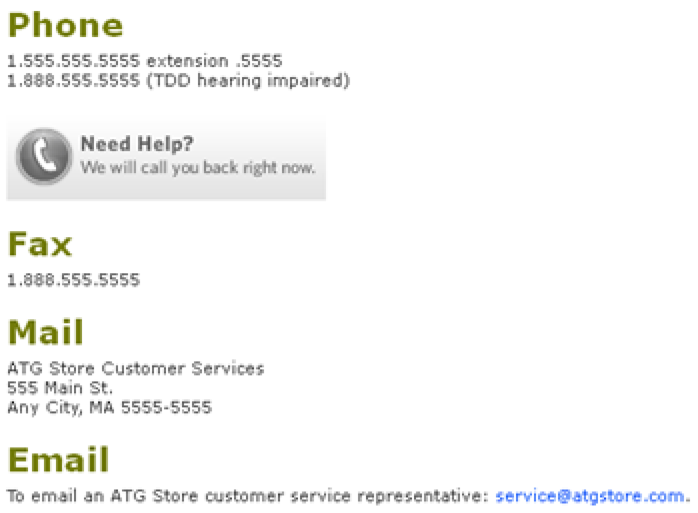
Executing the JavaScript Calls

When the HTML page loads, the JavaScript calls that were written by the ADC processor pipeline are executed. These calls import the Click to Call libraries and set the Click to Call variables.

When the libraries are loaded, they execute functions that insert Click to Call links in the <div> tags. These links include JavaScript calls for creating popup windows for initiating phone calls with customer service agents. For example, the following appears on the Contact Us page:

```
<div id="atg_store_c2c_contactUs">
  <a href="javascript:webVoicePop('Template=733654','urid=256029');">
    <img border="0" alt="Click to speak to a representative"
     src="http://as00.estara.com/webcare/public
    /linkimage.php?ulbid=733654&urid=256029&dnc=1323284899.502471921455">
  </a>
</div>
```

This code creates the link that appears on the Contact Us page below the Phone heading:


Phone

1.555.555.5555 extension .5555
1.888.555.5555 (TDD hearing impaired)

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.

For more information about Click to Call, see the *ATG Commerce Service Center Installation and Programming Guide*.

14 Oracle Recommendations On Demand Integration

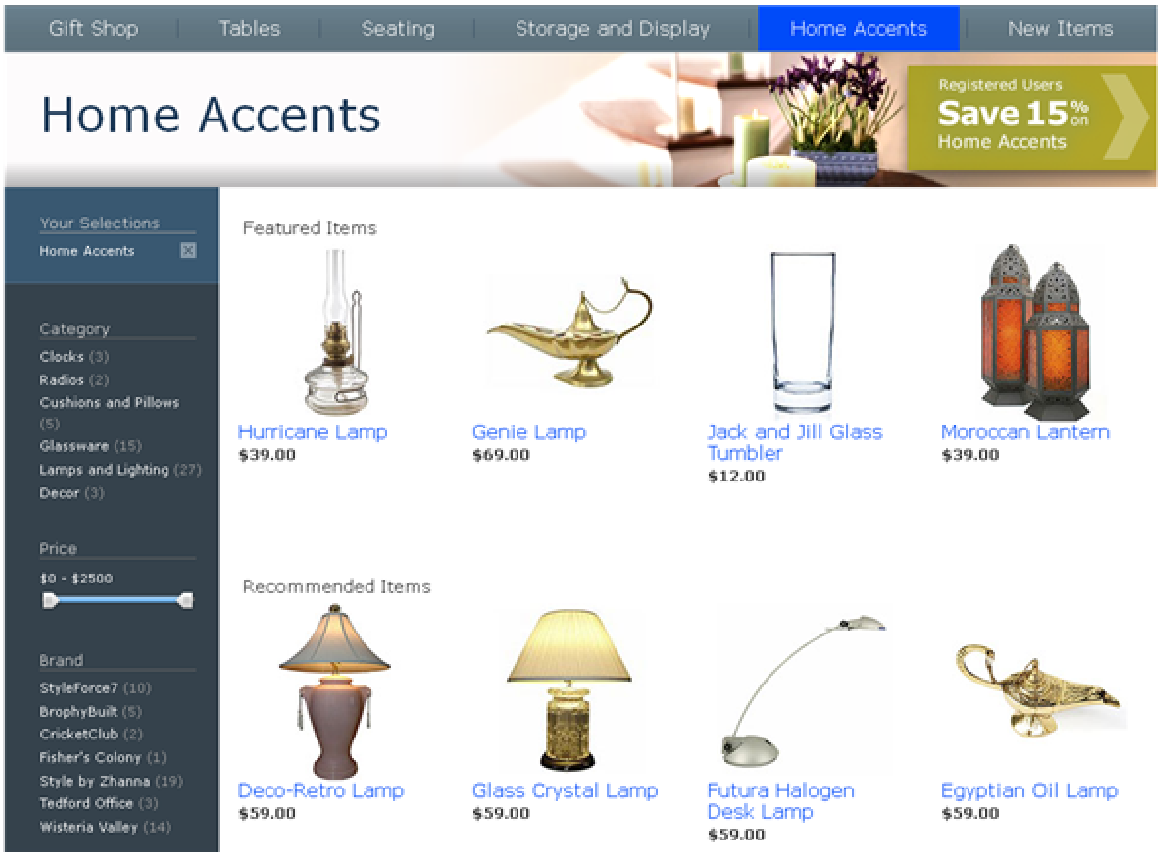
Commerce Reference Store can optionally use Oracle Recommendations On Demand to display recommended products on category pages and product detail pages. The recommendations are based on the shopper's browsing and purchasing history, as well as the browsing and purchasing history of other customers.

This chapter describes the Commerce Reference Store integration with Oracle Recommendations. It includes the following sections:

Running Commerce Reference Store with Oracle Recommendations

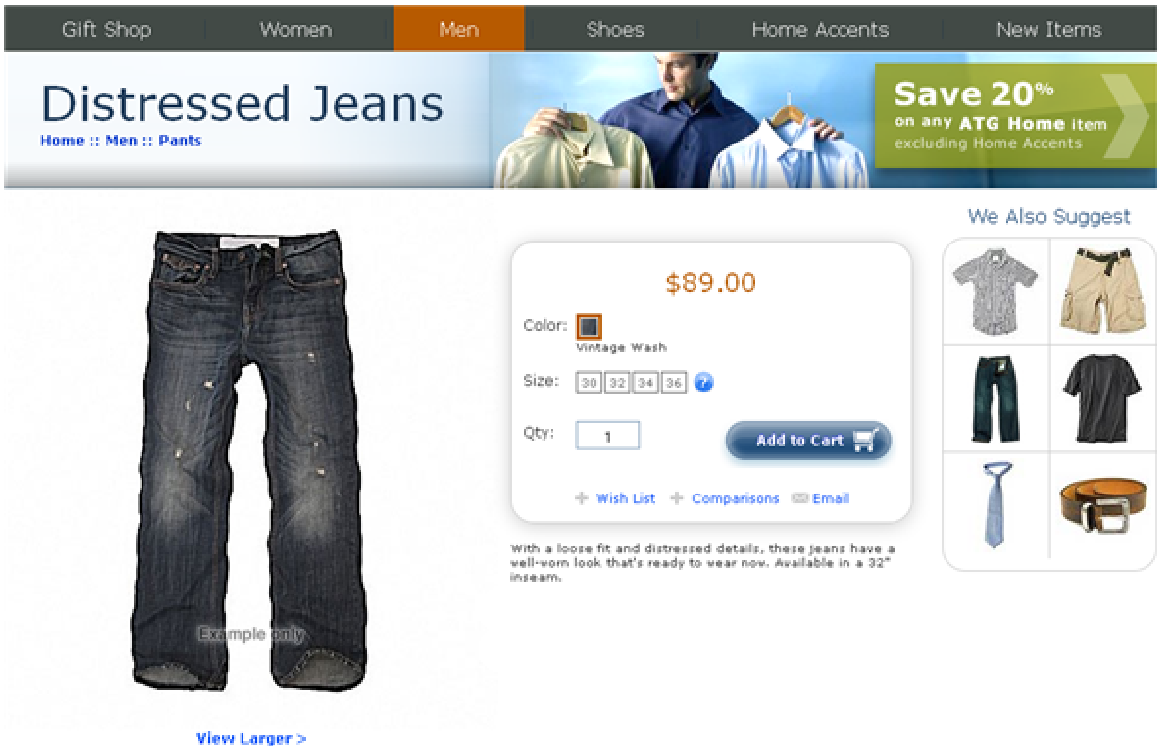
When you run Commerce Reference Store with Oracle Recommendations, the Commerce Reference Store category pages and product detail pages display recommended products that are determined by Oracle Recommendations.

On a category page, up to four products are displayed in a row at the bottom of the page, under the Recommended Items label:



The items under Recommended Items are all provided by Oracle Recommendations. The Recommended Items section is not displayed if Oracle Recommendations is not available.

On a product detail page, up to six products are displayed in a tabular format on the right side of the page, under the We Also Suggest label:



With a loose fit and distressed details, these jeans have a well-worn look that's ready to wear now. Available in a 32" inseam.

The items under We Also Suggest are a mix of recommendations and merchandiser-defined cross-sells. If Oracle Recommendations is not available, the We Also Suggest section is still displayed, but consists only of merchandiser-defined cross-sells.

Configuring the Oracle Recommendations Integration

To use Commerce Reference Store with Oracle Recommendations, include the `Store.Recommendations` module in your storefront EAR file. For internationalized sites, include the `Store.Recommendations.International` module.

To access Oracle Recommendations, you need an account. You use this account to configure rules that control what recommendations are produced, and to upload an XML file containing your product catalog repository data to Oracle Recommendations.

Commerce Reference Store includes a sample read-only account that is preconfigured to work with the Commerce Reference Store sites to demonstrate Oracle Recommendations functionality. The Commerce Reference Store product catalog has already been uploaded to this account. Note that if you build your own site based on Commerce Reference Store and want to use Oracle Recommendations, you need to obtain an account and use it to configure the rules for your site.

The Commerce Reference Store integration with Oracle Recommendations is configured through the `/atg/store/recommendations/StoreRecommendationsConfiguration` component. Key properties of this component include:

retailerId

A unique string that identifies the Oracle Recommendations account. By default, this is set to the Commerce Reference Store demo account.

recommendationsScriptUrl
The URL of the Oracle Recommendations JavaScript library.

cartChangeJMSTypes
The list of JMS event types that trigger inclusion of shopping cart content in the clickstream tracking code.

Commerce Reference Store configures the component like this:

```
recommendationsScriptUrl=//static.atgsvcs.com/js/atgsvcs.js

retailerId=cfa93013

searchTermQueryArgs=Ntt
searchResultsPages=\
    /cartridges/PageSlot/PageSlot.jsp

searchResultsPagesCache=/atg/store/recommendations/SearchResultsPagesCache

priceListManager=/atg/commerce/pricing/priceLists/PriceListManager

catalogNavHistoryPath=/atg/commerce/catalog/CatalogNavHistory

cartChangeJMSTypes=\
  atg.commerce.order.ItemAddedToOrder,\
  atg.commerce.order.ItemQuantityChanged,\
  atg.commerce.order.ItemRemovedFromOrder,\
  atg.commerce.fulfillment.SubmitOrder,\
  atg.commerce.promotion.ScenarioAddedItemToOrder,\
  atg.dps.Login,\
  atg.dps.StartSession,\
  atg.commerce.pricing.PriceChanged,\
  atg.multisite.SiteChanged

includeRootCategory=true
```

Oracle Recommendations Campaigns

You use your Oracle Recommendations account to configure *campaigns*, which are sets of business rules that control which recommendations are generated, based on site activity and other criteria. The account used by Commerce Reference Store is configured with two different campaigns, one for category pages, and one for product detail pages. A container `<div>` element in the page HTML includes instructions for Oracle Recommendations that specify the number of products to recommend, the products to exclude from recommendations, the locations and layout of the recommendations, etc. See Inserting a Container <div> Element (page 174) for more information.

Category Page

On a category page, up to four recommendations are displayed at the bottom of the page, under the Top Sellers label. The recommendations supplied are selected from the most popular products (defined as being in the top 10% of sales) within the current category and its descendant categories. The recommendations include only products on the current site.

The merchandiser-defined cross-sell products displayed below the Featured Items label by the `HorizontalRecordSpotlight` cartridge are explicitly excluded from the recommendations, as are any products displayed on the page by any other cartridges. The list of the products to exclude is passed to Oracle Recommendations through the container `<div>` element.

Product Detail Page

On a product detail page, up to six recommendations are displayed below the We Also Suggest label on the right side of the page. These are a mix of recommendations generated by Oracle Recommendations and merchandiser-defined cross-sells. The cross-sells may include items on sites other than the current site.

The recommendations generated by Oracle Recommendations are restricted to products from a list of specified categories and their descendant categories. The list of these categories is determined by the `RecommendationCategoriesForProduct` servlet bean and is passed to Oracle Recommendations through the container `<div>` element. The categories include the current product's category and its related categories (specified by its `relatedCategories` property).

Commerce Reference Store also passes to Oracle Recommendations a list of the merchandiser-defined cross-sells to include in the set of recommended products. The list contains the current product's related products (specified by its `relatedProducts` property). This list is filtered by the `ItemSiteGroupFilterDroplet` to include only items from the current cart sharing site group.

Embedding Recommendations in Pages

This section describes the process by which Oracle Recommendations data is rendered when a customer accesses a Commerce Reference Store page.

Commerce Reference Store uses the ADC processor pipeline to insert Oracle Recommendations code in site pages. The pipeline is triggered by JMS events that are fired when a customer views a page or performs certain actions.

The ADC processor pipeline inserts in all Commerce Reference Store pages:

- A statement in the `<head>` element of the page that imports the Oracle Recommendations JavaScript library.
- Clickstream tracking code in the `<body>` element.

In addition, on category pages and product detail pages, the Commerce Reference Store JSP code inserts a container `<div>` element that specifies information about the recommendations to be included.

When a page loads, the call to the JavaScript library is executed. A request is sent to Oracle Recommendations that includes the clickstream information and, if there is a container `<div>`, also includes the information in the `<div>`. If there is a container `<div>`, Oracle Recommendations returns a response containing the recommendations, which are passed to the specified JavaScript renderer, which modifies the HTML on the page to display the recommendations.

Note that some of the mechanisms used to trigger the ADC processor pipeline and to render HTML used by Oracle Recommendations differ depending on whether the page is Endeca-driven (i.e., category pages and search result pages) or ATG-driven (all other pages). These differences are discussed in the sections below.

Enabling the ADC Pipeline

To enable the ADC processor pipeline, 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 ADC processor pipeline. The filter is part of the `ADC` module, which is automatically included in the assembled EAR file when you specify the `Store.Recommendations` module.

Importing the Oracle Recommendations JavaScript Library

The ADC processor pipeline inserts in the `<head>` element of the HTML page a call that imports the Oracle Recommendations JavaScript library:

```
<script type="text/javascript" src="//static.atgsvcs.com/js/atgsvcs.js"></script>
```

This call accesses the Oracle Recommendations server, which downloads the library for inclusion in the page.

Inserting Clickstream Tracking Code

In addition to inserting the call to the JavaScript library in the `<head>` element, the ADC pipeline inserts clickstream tracking code in the `<body>` element of the page. This is HTML that is not actually displayed, but which is used to supply Oracle Recommendations with information about the contents of the page for tracking purposes. After the page loads, the `atgsvcs.js` script executes and sends the information in the clickstream code to Oracle Recommendations.

For example, on a product detail page, the tracking code provides information about the product being viewed:

```
<div id="cs-cfg" style="display:none !important;">
  <dl class="cs-cfg">
    <dt>retailerId</dt><dd>fb4a7379</dd>
    <dt>view</dt>
      <dd>
        <dl>
```

```
          <dt>storeId</dt><dd>storeSiteUS</dd>
          <dt>excludeDefaultStore</dt><dd>true</dd>
          <dt>productId</dt><dd>prod20009</dd>
        </dl>
      </dd>
    <dt>-inc-price</dt><dd>true</dd>
    <dt>locale</dt><dd>US</dd>
    <dt>-failover</dt>
      <dd>
        <dl>
          <dt>content</dt><dd>true</dd>
        </dl>
      </dd>
  </dl>
</div>
```

Clickstream tracking code on other pages may include different information. For example, the tracking code for the shopping cart page includes information about the items in the cart, and the tracking code for a browse page (search results) includes information about the search term the customer entered. This information is used by Oracle Recommendations to determine, for example, the likelihood of a certain item being purchased if it was found by entering a specific search term.

Endeca-Driven Pages

To insert clickstream tracking code on an Endeca-driven page, the page code must determine whether the page is a category page or a browse page. To do this, the `catalogItemBrowsed.jsp` gadget, which is included in pages by the `<crs:pageContainer>` tag, invokes the `StoreCartridgeTools.userOnCategoryPage()` method:

```
<dsp:getvalueof var="isUserOnCategoryPage"
  bean="StoreCartridgeTools.userOnCategoryPage"/>
```

The method checks whether the current navigation state includes the `CategoryOnly` user segment. If so, the method returns `true`, and the gadget triggers a `CategoryBrowsed` event that identifies the category being viewed:

```
<c:if test="${isUserOnCategoryPage }">

  <dsp:droplet name="CategoryLookup">
    <dsp:param name="id" bean="CatalogNavigation.currentCategory"/>
    <dsp:oparam name="output">
      <%-- Send 'Category Browsed' event --%>
      <dsp:droplet name="/atg/commerce/catalog/CategoryBrowsed">
        <dsp:param name="eventobject" param="element"/>
      </dsp:droplet>
    </dsp:oparam>
  </dsp:droplet>
</c:if>
```

The ADC pipeline uses the information about the category when it constructs the clickstream tracking code for the page.

If the `userOnCategoryPage()` method returns `false`, but the page URL matches the configured value for the search results page, the page is a browse page. The ADC pipeline is supplied with information about the search term used in the query, which it includes in the clickstream code. This behavior is configured through the `StoreRecommendationsConfiguration` component:

```
searchTermQueryArgs=Ntt
searchResultsPages=\
    /cartridges/PageSlot/PageSlot.jsp
```

`Ntt` is the query parameter used to pass the search term to Endeca, and `/cartridges/PageSlot/PageSlot.jsp` is the URL of the renderer for the top-level content item returned for a `/browse` request. See the *Using Cartridges* (page 115) chapter for more information.

Inserting a Container <div> Element

On product detail pages and category pages, Commerce Reference Store JSP code also inserts a container `<div>` element that specifies information about the recommendations to be displayed.

On category pages, which are Endeca-driven, the container `<div>` is inserted by `store.war/cartridges/ProductSpotlight-ATGCategoryRecommendations.jsp`, which is the renderer for the `ProductSpotlight-ATGCategoryRecommendations` cartridge. See the *Using Cartridges* (page 115) chapter for information about how the `<div>` is inserted.

On product detail pages, the content of the container `<div>` element is generated by the `store.war/browse/gadgets/productRecommendationsContainer.jsp` gadget. This gadget checks for the existence of the `StoreRecommendationsConfiguration` component, and if this component is present, uses the `/atg/store/recommendations/droplet/RecommendationCategoriesForProduct` servlet bean to generate a list of the categories that Oracle Recommendations should draw recommended products from.

The `productRecommendationsContainer.jsp` gadget also includes the following code to filter the product's related products to create the list of merchandiser-defined cross-sells to include in the recommendations:

```
<dsp:droplet name="ItemSiteGroupFilterDroplet">
  <dsp:param name="collection" param="product.relatedProducts"/>

  <dsp:oparam name="output">
    <dsp:getvalueof var="filteredRelatedItems" param="filteredCollection"/>
    <dt>includeRandom</dt>
    <dd>
      <dl>
        <c:forEach var="relatedProduct" items="${filteredRelatedItems}">
          <dt>${relatedProduct.repositoryId}</dt>
        </c:forEach>
      </dl>
    </dd>
  </dsp:oparam>
</dsp:droplet>
```

Executing the Oracle Recommendations JavaScript Call

When the `atgsvcs.js` script executes (after the HTML page loads) and sends the information in the clickstream code to Oracle Recommendations, it also sends the information from the container `<div>` about the product recommendations to be returned. Oracle Recommendations returns a response containing data about the products to display, and executes renderer JavaScript code that manipulates the DOM of the page to alter the

HTML to display the recommendations. For example, the following is a portion of the HTML created by the JavaScript renderer to create the Recommended Items display:

```
<div id="cs-recslot-category" class="cs-slot" style="display: block;">
  <div id="atg_store_prodList">
  <h3>Recommended Items</h3>
  <ul id="atg_store_product" class="atg_store_product">
     <li id="cs-rec-cs_recslot_categoryxprod2089" class="cs-rec odd first">
     <a href="/crs/storeus/browse/productDetailSingleSku.jsp?productId=xprod2089">
       <span class="atg_store_productImage">
       <img src="/crsdocroot/content/images/products/medium/
         HOME_PotbellyBottle_medium.jpg" alt="Potbelly Bottle">
       </span>
      <span class="atg_store_productTitle">Potbelly Bottle</span>
      <span class="atg_store_productPrice">$14.00</span>
    </a>
    </li>
```

For more information about Oracle Recommendations, see the Oracle web site (`www.oracle.com`).

15 Internationalization

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

Running the International Module

Internationalization functionality is implemented in Commerce Reference Store through the `Store.EStore.International` module, located in `<ATG10dir>/CommerceReferenceStore`. If you choose the Internationalization add-on when configuring Commerce Reference Store 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 Commerce Reference Store sites that support multiple languages or multiple countries. If you do not install the `International` module, your production instance of Commerce Reference Store 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

Commerce Reference Store 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* (page 9) chapter. It is a good idea to read the *Multisite Features* (page 9) chapter first, before continuing with this chapter. The remainder of this chapter focuses on how translations work in Commerce Reference Store 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 Commerce Reference Store. 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 Commerce Reference Store 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 (page 18) in the *Multisite Features* (page 9) 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 (page 179) 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

Commerce Reference Store 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

Commerce Reference Store 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/Storefront/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 resource bundle location from the site context. If it doesn't
  exist in the site context, get it from the configuration file.
--%>
<c:choose>
  <c:when test="${not empty resourceBundle}">
    <dsp:setLayeredBundle basename="${resourceBundle}"/>
  </c:when>
  <c:otherwise>
    <dsp:setLayeredBundle
         basename="${currentStoreConfiguration.defaultResourceBundle}"/>
  </c:otherwise>
</c:choose>
```

After the `RequestLocale` component determines the correct `locale` for a customer (as described in Determining a Customer's Locale (page 178)), `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.

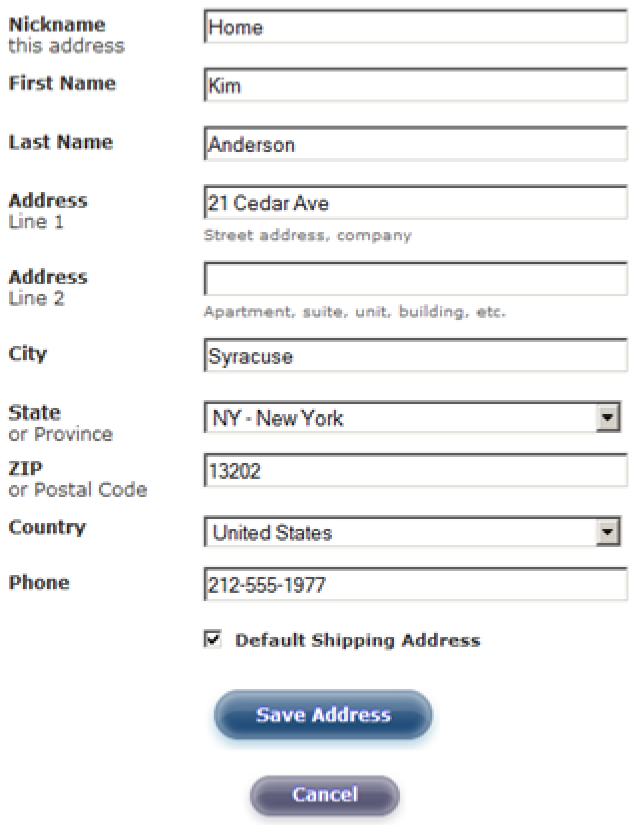
All three sites in Commerce Reference Store have their `resourceBundle` property configured to `atg.projects.store.web.WebAppResources`. Commerce Reference Store also configures the `StoreConfiguration.defaultResourceBundle` property to `atg.projects.store.web.WebAppResources`, so, regardless of where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

Displaying the Strings

Commerce Reference Store 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 address editing form rendered by the `store.war/myaccount/gadgets/addressEdit.jsp` gadget:



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

Commerce Reference Store error messages for the default locale (English) are defined in a number of locations. For the English translations, Commerce Reference Store uses a layered resource bundle model, where some of the error messages come from the ATG platform and Oracle ATG Web Commerce, and then Commerce Reference Store 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.

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

- `/atg/userprofiling/ProfileUserResources.properties`

Commerce Reference Store uses these resource bundles from Oracle ATG Web 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`

Commerce Reference Store layers these resource bundles on top of bundles that exist in Oracle ATG Web 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, Commerce Reference Store 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`.

Commerce Reference Store 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`.

You can 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 Commerce Reference Store.

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 Commerce Reference Store) are stored.

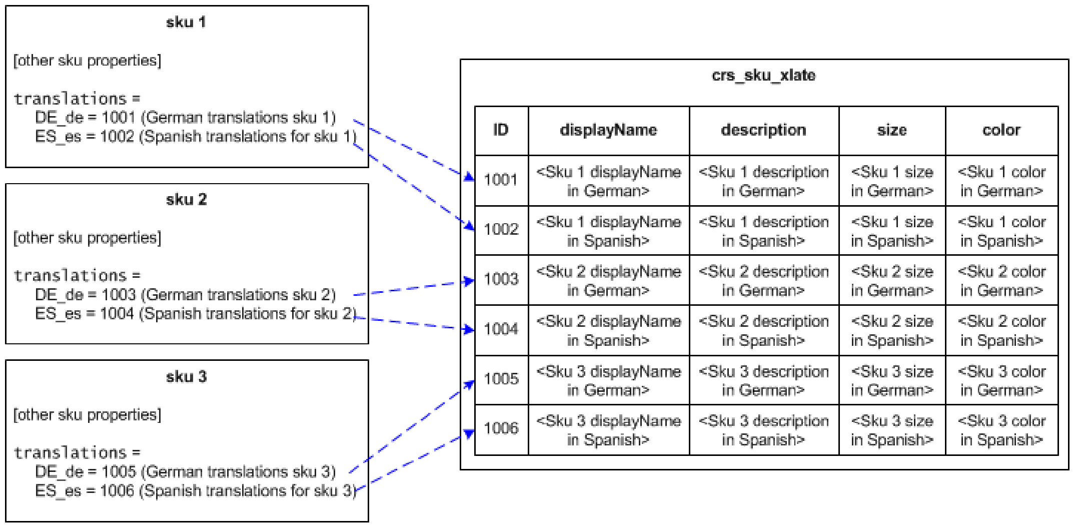
- A `translations` property is added to the item type. The `translations` property is a map whose key is a `locale` and whose value is an item of type <*baseType*>`Translation`, described below. Note that the `locale` key does not have to be a fully qualified locale. In fact, Commerce Reference Store 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 (page 187).

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 illustration shows three `sku` items and six corresponding `skuTranslation` items which contain translated content for two locales, Spanish and German:

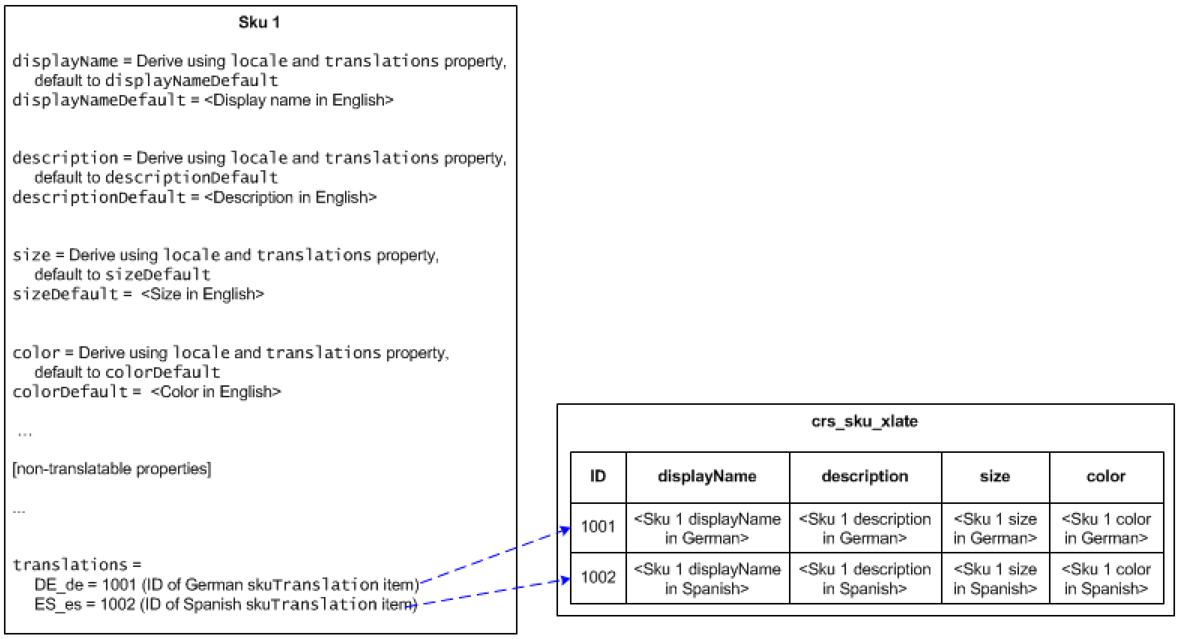

sku 1
[other sku properties]
translations =
DE_de = 1001 (German translations sku 1)
ES_es = 1002 (Spanish translations for sku 1)
sku 2
[other sku properties]
translations =
DE_de = 1003 (German translations sku 2)
ES_es = 1004 (Spanish translations for sku 2)
sku 3
[other sku properties]
translations =
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 has been modified to add four <*translatableProperty*>`Default` properties: `displayNameDefault`, `descriptionDefault`, `sizeDefault`, and `colorDefault`. The `sku` base item has also been modified to add the `translations` property, which contains the map connecting the base `sku` item to its German and Spanish translation items. Finally, the `displayName`, `description`, `size`, and `color` property definitions have been changed 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]

...

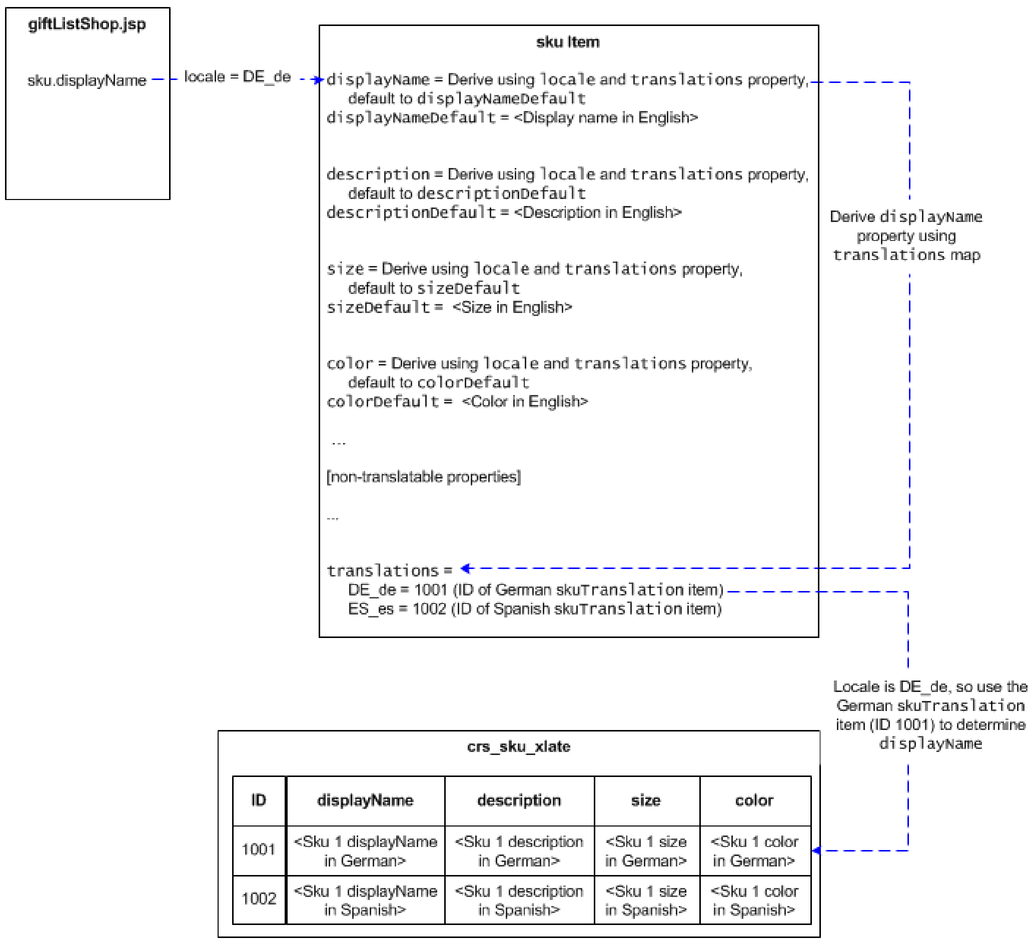
`translations` =
DE_de = 1001 (ID of German sku`Translation` item)
ES_es = 1002 (ID of Spanish sku`Translation` item)

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>

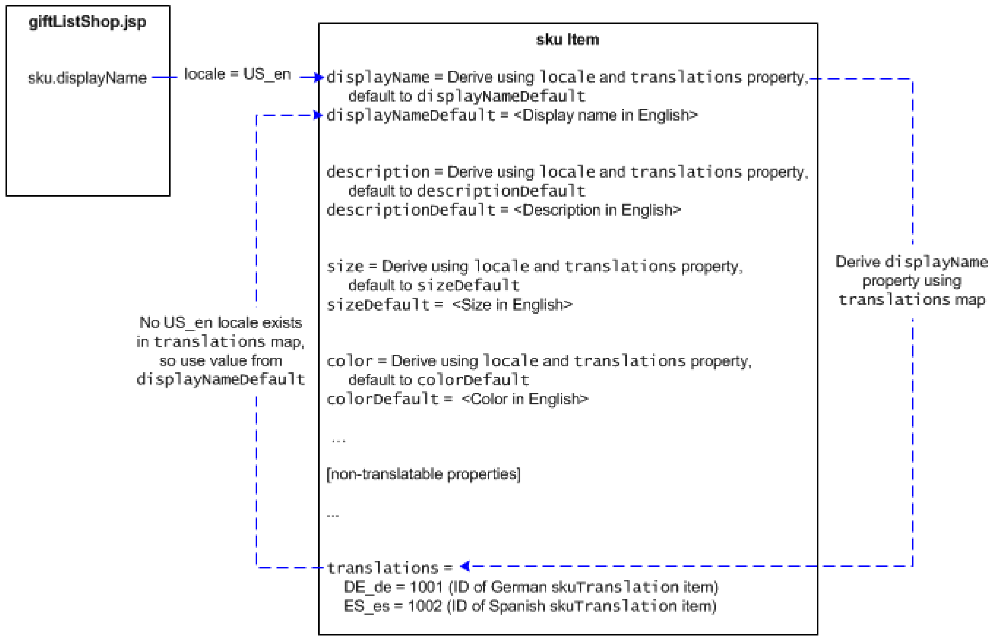
In this illustration, the following happens:

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 (page 178) and, in this example, is `DE_de`.

2. The catalog repository finds the corresponding `skuTranslation` item using the `sku.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 `sku.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 (page 178) and, in this example, is `US_en`.

2. The catalog repository derives the display name for the SKU using the `sku.translations` property map. The repository determines that no `skuTranslation` item exists for the `US_en` locale, so it returns the value from the `sku.displayNameDefault` property, which for Commerce Reference Store is the English translation.

Repository Item Translation Benefits

The approach Commerce Reference Store 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 Commerce Reference Store.

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 Business Control Center.
2. Expand Merchandising, and 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`, Commerce Reference Store 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 Cascading Stylesheets (CSS) (page 24) in the *Pages and Navigation* (page 21) chapter for more details.

Displaying Site and Locale-specific Images

Commerce Reference Store 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/content/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/content/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/content/images/atghome`, `/atgstore`, `/c2call` 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, Commerce Reference Store determines which cascading stylesheets to use, as described in the Cascading Stylesheets (CSS) (page 24) section of the *Pages and Navigation* (page 21) chapter. Commerce Reference Store 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/content/images/atghome` while the `store.css` stylesheet references images in `storedocroot.war/content/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) (page 24) 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 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 (page 109) section of the *Order Handling* (page 107) 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, as shown in the table below:

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. Sample Data

This chapter describes the user profile, catalog, and store location data included with Commerce Reference Store. It includes the following topics:

Sample User Data

This section describes the sample data created for Commerce Reference Store 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

Commerce Reference Store 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, Commerce Reference Store 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 Business Control Center.
2. Expand Personalization, and 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)
Registered Users – Save 15% on 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)
Save 20% on any ATG Home item excluding Home Accents	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: Commerce Reference Store 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

- Get 2 free gifts with any Woman's Jacket
- Free Scarf with all orders over $200

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 Commerce Reference Store 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 Commerce Reference Store 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
          Handbags and Wallets
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 and Oil Lamps
     Décor
```

homeStoreCatalog Hierarchy

```
Gift Shop
     Glassware
     Unique and 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 and 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 and Oil Lamps
Decor
```

Representative Products

This section lists representative products that demonstrate various Commerce Reference Store 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 Sofa 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
Backorderable SKU	ATG Home > Home Accents > Lamps and Lighting > Table Lamps > Cottage Lamp ATG Store > Gift Shop > For Him > 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

Use case	Representative product(s)
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

Commerce Reference Store 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 Merchandising:

1. Start the Business Control Center.
2. Expand Merchandising, and 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 `store.war/company/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 (for example, finding the store closest to a customer’s location).

Appendix B. Disabling Cookies

Commerce Reference Store uses a number of cookies to store information for tailoring site behavior to specific users. If you build a site based on Commerce Reference Store, you may prefer to disable these cookies to comply with local privacy laws and regulations. This appendix lists the cookies used by Commerce Reference Store and related products, explains how to disable them, and discusses the effect of disabling cookies on site behavior.

Note that the information in this appendix is provided to help you implement your strategy for cookie use, but does not constitute legal guidance. You are solely responsible for ensuring that your use of cookies is consistent with your legal and regulatory obligations.

Cookies Used by Commerce Reference Store

This section discusses the three cookies used by the Commerce Reference Store application itself. These consist of two cookies used to associate a customer with his or her user profile, and one cookie that stores information about a customer's preferred language.

Profile Cookies

Commerce Reference Store configures the ATG platform to set two cookies, `DYN_USER_ID` and `DYN_USER_CONFIRM`, for storing user profile information. The `DYN_USER_CONFIRM` cookie is a hash of the `DYN_USER_ID` cookie.

These cookies are used for automatic login, the process of detecting a returning customer and associating the customer with his or her existing user profile. The purpose of using the two cookies is to protect against a customer gaining access to another user's profile by modifying the `DYN_USER_ID` cookie. The user profile associated with `DYN_USER_ID` is used only if it matches the hashed value in `DYN_USER_CONFIRM`. If the cookies do not match, a new profile is created instead.

You can disable profile cookies by setting the `sendProfileCookies` property of the `/atg/userprofiling/CookieManager` component to `false`. This disables both the `DYN_USER_ID` and `DYN_USER_CONFIRM` cookies.

Disabling these cookies prevents automatic login, which means a returning customer is not automatically associated with his or her profile. Automatic login also enables Commerce Reference Store to display the welcome message on the home page and populate the shopping cart with items placed there during previous shopping sessions. Note that automatic login does not allow access to secured areas such as checkout; even if automatic login is enabled, the customer must log in to check out, or use guest checkout.

For more information about managing profile cookies, see *ATG Personalization Programming Guide*.

Preferred Language Cookie

Commerce Reference Store sets the `userPrefLanguage` cookie to store the customer's preferred language. This cookie is Commerce Reference Store-specific, and cannot be disabled without making code changes to Commerce Reference Store. To disable the cookie, modify the `atg.projects.store.servlet.pipeline.StoreProfilePropertySetter` class; change the `setProperties()` method of this class to remove the call to `createUserPrefLangCookie()`. (The source code for this class can be found in the `src/atg/projects/store/servlet/pipeline/StoreProfilePropertySetter.java` file in the `Store.Estore` module.)

Disabling this cookie prevents storing of the customer's preferred language. Each time the customer returns to the site, the site will appear in the default language, and the customer will need to select his or her preferred language again.

Cookies Used by Integrated Products

Commerce Reference Store can optionally integrate with Oracle Live Help On Demand and Oracle Recommendations On Demand. These services, which each set several cookies of their own, are available only through the software as a service (SaaS) model, and do not currently provide a means of disabling the use of cookies. In order to provide these services, your application must be permitted to set cookies.

It is possible, however, to selectively provide these services only to customers who agree to allow cookies to be set, and disable these services for customers who do not allow cookies. To do this, you must provide a way for each customer to specify whether to allow cookies, and implement a function that includes the `//static.atgsvcs.com/js/atgsvcs.js` JavaScript library in pages only if cookies are permitted. If this library is not included, cookies will not be set and the services will not be enabled for the customer.

See the documentation for these On Demand services for more information.

Other Cookies

Other applications you use on your site may set their own cookies. Be sure to consult the documentation for those applications for information about the cookies they set and how to disable them.

Note that any application server you run Commerce Reference Store on sets a cookie called `JSESSIONID`, which stores a unique session ID generated by the application server. The application server can be configured to use a different name for this cookie. The application server can also be configured to avoid the use of this cookie entirely and instead use URL rewriting. See the documentation for your application server for more information.

Note that if you disable the use of cookies by the application server, you should also set the `usingCookies` property of the `/atg/dynamo/servlet/sessiontracking/SessionManager` component to `false`, so that your application is aware that cookies are not being used for session management.

Index