ORACLE®

ATG WEB COMMERCE

Version 10.0.2

Multisite Administration Guide

Oracle ATG
One Main Street
Cambridge, MA 02142
USA

ATG Multisite Administration Guide

Document Version

Doc10.0.2 MULTISITE 04/15/2011

Copyright

Contents

Part I: Multisite Overview

The term *multisite* refers to running multiple Web sites from a single ATG instance, with the sites sharing certain resources. For example, consider a clothing manufacturer that has two brands, a bargain brand and a luxury brand, and wants to run separate sites for the two brands.

In the simplest case, these sites may share only user profiles, so a customer registered on one site is also registered on the other. In more complex implementations, the sites may share shopping carts, so a customer can include items from both sites in a single order. The bargain-brand site might also include upsell recommendations for (and links to) items on the luxury brand site, or the sites might share gift lists.

This section provides an overview of multisite concepts and ATG's implementation of multisite functionality. Its intended audience consists of business users, developers, and system administrators who are designing and developing multisite applications.

The chapters in this section include:

Note that most of the topics discussed in this overview are covered in greater depth in the other sections of this document.

1 Introduction to Multisite

This chapter introduces key multisite concepts, and discusses the following topics:

Multisite Approaches

There are a variety of multisite approaches, which are distinguished by the relationships between the sites and the types of resources the sites share. The multisite approach that makes sense for your environment depends, of course, on the nature of your business. Some of the more common approaches include:

- affiliated sites
- microsites
- country sites

These approaches are described below.

Note that these terms are not necessarily used universally, and a given term may have different meanings depending on the context. Also, these approaches are not rigid, nor are they mutually exclusive. A given multisite implementation may include aspects of any or all of these approaches. The terms are useful for making general distinctions, but the critical questions for any multisite implementation are what resources are shared across sites and what resources are not.

Affiliated Sites

Affiliated sites are "sibling" sites that have largely independent catalog content that differs by theme or brand. A clothing manufacturer that wants to have separate sites for two different brands might use affiliated sites. In this example, there would typically be little or no overlap between the items sold at the two sites. The items might be similar across sites (for example, both sites might sell blue jeans), but each site would have its own products and SKUs.

Typically, affiliated sites are tied together fairly loosely. They may include cross-site links, and have shared user profiles and shopping carts, but catalog items are not shared.

Microsites

In contrast to affiliated sites, which have a sibling relationship, a microsite is a relatively small site that is a child of a larger parent site. In many cases, users may not even be aware that the microsite is a different site; it may appear to be a special area of the parent site.

Microsites can be used in a variety of ways. A microsite can promote a specific product line, seasonal items, or items associated with a specific event. For example, a site that sells flowers might have an associated microsite that is available only around Mother's Day.

A microsite typically shares many resources with its parent site. Generally, a microsite will share user profiles and shopping carts with the parent site, and most if not all of its catalog content will be drawn from the parent site's catalog.

Country Stores

Country stores, like affiliated sites, have a sibling relationship, but their catalog content differs by country or locale. For example, consider a store that has separate sites for customers in the United States and Canada. The two sites might sell mostly the same products and SKUs. But the sites would need to differ in several ways:

- The US site might be in English only, while the Canadian site would be in English and French.
- Prices on the US site would be in US dollars. Prices on the Canadian site would be in Canadian dollars.
- Although the sites sell mostly the same items, there might be some differences: items that are popular in one country and not the other, country-specific versions of certain items, etc.

Country stores typically do not share shopping carts with each other, because of issues with currency conversion and shipping.

ATG Multisite Support

The ATG platform includes a number of features to facilitate creation of multisite applications. These features are of two main types:

- Data sharing mechanisms that allow sites to share commerce and personalization resources, such as catalog items, shopping carts, orders, search results, promotions, scenarios, and targeters.
- Administrative tools for creating and managing sites.

Data Sharing

A key aspect of multisite support is the sharing of data. For example, if two sites share a gift list, a customer can include products from both sites on his or her gift list. This is much more convenient for the customer than having separate gift lists for each site.

ATG provides a number of mechanisms for sharing data across sites. Using ATG's multisite support, your sites can share:

- Personalization tools, such as scenarios and targeters
- Catalog data, such as categories, products, and SKUs
- Standard commerce facilities, such as shopping carts, promotions, orders, gift lists, and product comparisons
- Search results

These elements can be shared across all sites, a subset of sites, or be unique to each site.

Data sharing is managed by a variety of mechanisms. Primary among these is the use of *site groups*.

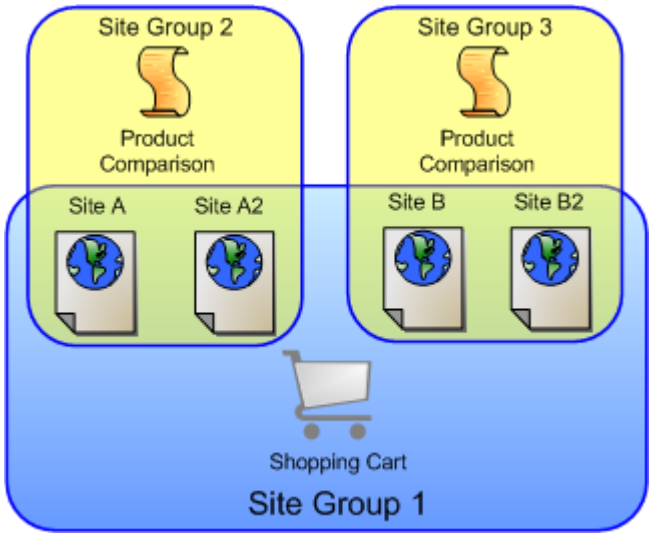
Site Groups

A site group comprises a subset of the sites in a multisite environment. Site groups provide a way to logically group sites to manage data sharing between those sites. A *sharing group* consists of a site group and the data the sites share.

For example, suppose your multisite environment initially consists of two affiliated sites, Site A and Site B, that share a shopping cart. These sites would be defined as a site group. When you define this site group, you specify that the sites share a shopping cart. These sites plus the shopping cart constitute a sharing group.

Now suppose each of these sites has an associated microsite. Site A's microsite is Site A2. Site B's microsite is Site B2. In each case, the microsite shares additional resources with the parent site, such as product comparisons. So you would need to create two other site groups: one containing A and A2, and one containing B and B2. You would put all four sites in the first site group, so they can all share a shopping cart.

The following diagram illustrates the situation:



Note: By default, sharing of product comparisons is tied to sharing of shopping carts. This example assumes for illustrative purposes that they are managed independently. See the Shopping Cart section of the Multisite Support in ATG Products chapter for more information.

Administrative Tools

ATG's multisite support is designed to make it as easy as possible to create and administer sites. To facilitate this process, the following tools are provided:

- Site Repository
- Site Administration
- Site categories

Site Repository

Many aspects of multisite support are managed through the Site Repository (`/atg/multisite/SiteRepository`). This repository stores information about each site in a multisite environment -- the site name, default URL, etc. A site listed in this repository is called a *registered* site.

The Site Repository determines the set of properties for a site. A number of properties are included by default, but a system administrator can add site properties by modifying the repository definition.

Note that if you have only a single site, you do not need to register it. If you subsequently add a second site, though, you will need to register both -- you cannot have a mix of registered and unregistered sites. In other words, multisite support requires that you register **all** of your sites. If you have already created an unregistered site and choose to register it, you may need to make some coding changes. See Coding a Multisite Application.

Site Administration

To register and administer sites in a multisite environment, ATG provides the Site Administration utility. Site Administration is available through the ATG Business Control Center, and is designed for business users. Site Administration enables business users to configure new sites, adjust the settings of existing sites, and disable sites that are no longer needed. Users can also create site groups, by specifying the sites in the group and the data they share. The settings specified in Site Administration are stored in the Site Repository.

Site Categories

When you create a site in the Site Administration utility, you select the site category for the site. For example, depending on how your system is configured, you might be able to select a Microsite or Country Store category. It is possible to create any number of categories, which may differ by the types of sites they are intended for.

Site categories are created by system administrators. To create a site category, the system administrator creates a site template item in the Site Repository. The site template item defines the editing environment in Site Administration for sites that use the corresponding site category. The template makes it possible to:

- Control the grouping of properties in Site Administration.
- Specify non-default editors for properties. For example, the standard editor for a text property is a text box where a user can enter an arbitrary string. A site template can replace the editor for a specific text property with a drop-down menu that has a set of preconfigured options.
- Suppress the display of certain properties, so that they cannot be set.
- Hard-code the values of certain properties.

2 Creating Multisite Applications

Creating a multisite application involves collaboration among business users, system administrators, and site developers:

- System administrators create the Site Repository definition and the site categories that are used to define sites in the Site Administration UI, and also configure the resources that can be shared within site groups.
- Business users create and manage sites using the Site Administration utility. The settings specified in Site Administration are stored in the Site Repository.
- Java and JSP developers code the application in a multisite-aware way.

This chapter provides an overview of creating multisite applications. It includes the following sections:

Configuring Multisite Support

Site properties, site categories, and shared data are essential parts of the site definitions created in the Site Administration utility. These elements are created by system administrators or developers, for use by business users within Site Administration.

Defining Site Properties

The Site Repository has three item types by default:

- `siteTemplate` -- Represents a site category. Stores the name and description of the site category, and the ID of the `itemMapping` item in the View Mapping Repository that contains the view mappings for the site category.
- `siteGroup` -- Represents a site group. Stores the name of the site group and references to the sites and shareable types (see below) that make up the group.
- `siteConfiguration` -- Stores the values of the site properties for an individual site. The available properties of this item type vary depending on the modules included in the application.

The default repository definition includes a number of properties for the `siteConfiguration` item type. The core (platform) properties include general properties for specifying a site name and description, the site category, URLs for accessing the site, etc. Other modules add properties through XML combination. For example, ATG Commerce adds these properties:

- `defaultListPriceList`
- `defaultSalePriceList`
- `defaultCatalog`

Other applications may add other properties or item types.

You can add properties to the `siteConfiguration` item type that are specific to your environment. For example, ATG Commerce Reference Store (CRS) adds a number of CRS-specific properties, such as `newProductThresholdDays` for specifying the number of days after being added that a product is considered new.

Using Site Categories

Site categories define the editing environment for the Site Administration utility. A site category is associated with a set of view mappings, which can specify non-default editors for certain properties or suppress their display altogether. For example, the standard editor for a text property is a text box where a user can enter an arbitrary string. Using view mapping, you could replace the editor for a specific text property with a drop-down menu that has a set of preconfigured options.

Unless specified otherwise in the site category's view mappings, each site property appears in Site Administration with the default editor for the property's data type. For example, a text property appears with a text box, a Boolean appears with radio buttons, etc.

By default, the Site Repository includes a single site category called the Default Category. The Site Administration utility provides a basic user interface (available only to administrators) for creating additional site categories. This interface is accessed through the Manage Site Categories workflow.

To create a site category, you fill in the name and description fields, and then select the item mapping ID for the view mappings to use. Note that this requires that you have already created the view mappings and imported them into the View Mapping Repository.

Defining Shared Resources

When creating a site group in the Site Administration utility, a business user selects the resources that can be shared within the group. These are typically items that hold data for the sites, such as shopping carts and product comparisons.

A list of the items available for sharing is displayed in Site Administration when a site group is being defined. To share an item within the site group, the business user selects the checkbox next to the item.

To enable an item or group of items to be shared within a site group (and to make the item or group of items appear in this list), you create a `ShareableType` component. A `ShareableType` component is a globally scoped Nucleus component of class `atg.multisite.ShareableType` that represents one or more shared data items. The shared items can be Nucleus components, Java objects, or other resources.

For shared Nucleus components, the `ShareableType` component should be of class `atg.multisite.NucleusComponentShareableType`. This is a subclass of `ShareableType`, which includes a `paths` property for specifying the pathnames of one or more shared Nucleus components.

Each `ShareableType` component must be registered with the `/atg/multisite/SiteGroupManager` component by adding it to that component's `shareableTypes` property:

```
shareableTypes+=/atg/multisite/ShoppingCartShareableType
```

Configuring Sites and Site Groups

Configuring and managing sites is designed to be as easy as possible. Site Administration provides a simple user interface (accessed through the Manage Site Assets workflow) that enables business users to quickly create sites and site groups, and to modify their settings.

The Site Administration utility has two main functions:

- To create a new site, you select a site category, and fill in the values of the site configuration properties.
- To create a site group, you specify the sites in the group, and the resources that the sites share.

The available options in Site Administration are configured by developers or system administrators, as discussed in Configuring Multisite Support. For more information about using Site Administration to create sites and site groups, see Administrative Tools.

Coding a Multisite Application

Coding a multisite application differs from coding a non-multisite application in a number of ways. This section discusses some the issues a multisite application must take into account:

- Sharing resources
- Site context
- URL management
- Page development

Sharing Resources

A key aspect of multisite support is the sharing of resources such as shopping carts and gift lists. From a coding point of view, however, the issue is not so much sharing as *restricting* sharing. By default, all resources are shared across sites. You code your application in a way that *limits* the sharing of resources.

Another way to think of this is as making resources *site-aware*. A site-aware resource tracks the site it is being used on, and behaves differently depending on the site.

For example, suppose you have three sites, A, B, and C. By default, all three sites share a shopping cart. To make the shopping cart site-aware, you create a shareable type component and associate it with the shopping cart. (Actually, this has already been done for you, but you can ignore that for the sake of this example.) Now the default behavior is for none of the sites to share a cart; each site has a separate instance of the cart.

Now that you have made the shopping cart site-aware, you can enable restricted cart sharing by creating sharing groups. For example, suppose you create a sharing group consisting of sites A and B and the shopping cart. Sites A and B now share one instance of the shopping cart, while site C has a separate cart instance.

Note that not all resources can be site-aware. For example, user profiles are not site-aware. User profiles (and therefore logins) are always shared across all sites in the application.

ATG provide a number of mechanisms for making resources site-aware, which are discussed below.

Site-Specific Session-Scoped Components (Shareable Types)

As discussed above, shareable types designate resources that are shared within site groups. If the shared resource is a Nucleus component, the application may need multiple instances of the same component in a single session. In the example above, the application needs two instances of the shopping cart in each session, one shared by site A and B, and the other used only by site C. That way, a site customer moving from site A to site B will see the same shopping cart in both places, but when moving to site C, the customer will see a different cart.

To simplify site development, switching between instances of shared Nucleus components is designed to happen automatically. Underlying system services are responsible for tracking the current site and using the correct instance of the shopping cart. In your own code, you refer to the cart as if it were a single entity.

Site-Specific Property Value Overrides

Another mechanism for making components site-aware enables you to override property values on a per site basis. For example, suppose you have multiple Commerce sites that each charge a different fixed price for shipping. To manage this, you could use a single `atg.commerce.pricing.FixedPriceShippingCalculator` component to determine the shipping cost, and override the component's `amount` property (which sets the shipping cost) for each site.

Site-specific overrides are handled through the `atg.service.subclasser.SiteSourcedPropertyGetterSubClasser` class. The ATG platform includes a component of this class, `/atg/multisite/SiteSourcedPropertyGetterSubClasser`. `SiteSourcedPropertyGetterSubClasser` overrides the value of a specific component property (such as `FixedPriceShippingCalculator.amount`) by setting it to the value of a property of the `siteConfiguration` repository item. This allows the component property to take on a different value for each site.

To override the value of the `amount` property of your `FixedPriceShippingCalculator` component:

1. Add a property named `siteShippingAmount` (for example) to the `siteConfiguration` item type in the Site Repository.

2. In the `FixedPriceShippingCalculator.properties` file, add the following special properties:

```
$instanceFactory=/atg/multisite/SiteSourcedPropertyGetterSubClasser
$overridePropertyToValuePropertyMap=\
       amount=siteShippingAmount
```

3. In the Site Administration utility, set the value of the site's `siteShippingAmount` property to the fixed shipping cost on the site.

Site-Specific Personalization

Targeters, user segments, and content groups all support site-specific rules. For example, suppose you have a user segment that by default includes all customers whose age is between 30 and 40 years old. You could add a site-specific override rule specifying that on site A the segment includes all customers whose age is between 30 and 50. As the result of this override, a customer who is 43 years old would be included in this segment when browsing site A, but not when browsing any other site.

In addition to site-specific rules, targeters and content groups also make it possible to show on one site content that is associated with other sites. For example, when customers browse products on site A, a targeter might display a set of related products from site B.

Scenario conditions and actions are also site-aware. For example, an action that fills a slot with items from a targeter can specify a site, causing the targeter to execute a site-specific rule.

Site-Specific Repository Data (Site Membership)

The ATG platform also includes a mechanism for making repository items site-aware. This is done by using a *context membership* property to designate which sites the item applies to. This mechanism is used in the catalog repository. The catalog, category, product, and SKU item types all have a `siteIds` property that holds an array of the site IDs of the sites that the item is part of. So, for example, if a product is included on site A and C, but not site B, the value of this property would be the site IDs of sites A and C.

Context membership is used by internal querying mechanisms of the repository, enabling it to issue queries such as "return all products that are included in site A." Note that other repositories may have `siteIds` properties to track which sites an item is associated with, but these are not context membership properties unless explicitly configured as such in the repository definition.

Site Context

The ATG platform includes a site context system for associating requests with specific sites. Tracking the site context is essential for ensuring the correct site-aware data is made available and the proper site-aware behavior is exhibited. All of the mechanisms discussed in the Sharing Resources section rely on the site context system.

Key elements in the site context system include:

- The `SiteContextPipelineServlet` is a servlet in the request-handling pipeline. It examines each incoming request and applies various rule checks to determine the site to associate the request with. For example, it can associate a request with a site based on the request URL, as described in URL Management.

- A `SiteContext` object tracks information about a site for the current request. For a given request, there can be `SiteContext` objects for one or more sites. At any given point in the handling of the request, one `SiteContext` is current. The `SiteContextManager` component manages `SiteContext` objects, making different `SiteContext` objects current as needed.
- A `SiteSession` object tracks information about a site for the current session. For a given session, there can be separate `SiteSession` objects for some or all of the sites. The `SiteSessionManager` component manages `SiteSession` objects.

You can use the Site Context API to control the site context in your application. For example, you can write rule filters for the `SiteContextPipelineServlet` that modify the logic for associating a request with a site. You can also change the current site in the course of handling a request. For example, suppose a page on site A includes a targeter displaying recommended products from site B. To do this, the targeter needs to execute within the context of site B. (See Page Development for information about how this is accomplished.)

JMS Events

Many JMS message classes now have a `siteId` property for specifying the site where an event is fired. Event senders use the current site context to set the value of this property. Systems that receive these messages can then use this information to affect system behavior. For example, `siteId` can be used to set the site in an action parameter for a scenario that is triggered by the event. Site information in events can be logged and loaded into the Data Warehouse for use in reporting.

URL Management

A multisite application must be able to quickly determine which site a given request is associated with. The ATG platform includes a very flexible URL management system for ensuring that each request is associated with the correct site. In addition, it provides ways of handling the potentially large number of different URLs that are required for multisite production, staging, and preview machines.

As discussed in the Site Context section, when a new request is received, the `SiteContextPipelineServlet` examines the URL to determine the site to associate it with. There are two main approaches to making the determination, one based on URL paths, and the other based on domain names.

Path-Based Strategy

In a path-based strategy, the domain name is the same for all sites, and the site is keyed off the context root in the URL. For example, suppose a sports equipment company, `sports.com`, has separate sites for baseball, football, and basketball equipment. The URLs for these three sites might begin with:

```
http://www.sports.com/baseball/
http://www.sports.com/football/
http://www.sports.com/basketball/
```

To handle this case, you would configure the URL management system to associate the request with a site based on the context root, so that a request whose URL includes `/football/` in the path would be associated with the football site.

The advantage of this approach is that it is makes it easy to manage cookies. Because the sites all have the same domain, they automatically share cookies (and thus sessions).

One difficulty with this approach is that an individual Web application (WAR file) can have only one context root. To make it possible for all of these sites to be part of the same Web application, ATG uses a technique called *virtual context roots*. Using virtual context roots, you map the sites' context roots (for example., `/baseball/`, `/football/`) to the actual context root of the Web application. Even though the virtual context root is not the true context root of the application, it still appears in the browser when a customer accesses the site.

Domain-Based Strategy

In a domain-based strategy, each site has a different domain name, and a request is associated with a site based on the domain name. This strategy is trickier to implement because by default application servers do not share cookies across domains.

There are a couple of variants of this approach, depending on whether the subdomain changes or the parent domain does.

If the subdomain changes, the domain names might be:

```
http://baseball.sports.com/
http://football.sports.com/
http://basketball.sports.com/
```

Note that using different subdomains is distinct from redirection. Some merchants use URLs like this but actually redirect to a single domain. For example, `baseball.sports.com` might redirect to `www.sports.com/baseball`, and the actual URL management strategy used would be path-based. In contrast, using different subdomains means that `baseball.sports.com` is the actual domain name, and it is what customers see in their browsers.

If the parent domain changes, the domain names might be:

```
http://www.baseball.com/
http://www.football.com/
http://www.basketball.com/
```

In the case where the parent domain is always the same, but the subdomain varies, you can configure your application server to use cookies that specify only the parent domain. For the case where the parent domain changes, the ATG platform provides a mechanism called session recovery. (You can also use session recovery if only the subdomain changes, but you do not want the application server to use cookies that specify only the parent domain.) Session recovery uses a special cookie-tracking mechanism to allow all of your sites to share a single session.

Sticky Site Parameters

The URL management system also provides a third mechanism, sticky site parameters, in which the site is specified explicitly through a query parameter, and the value persists throughout the session (or until it is explicitly overridden). This mechanism is primarily intended to be used in development and preview environments to override the site selection made through one of the other two mechanisms. In production environments, this mechanism should be disabled.

Page Development

The Java Server Pages (JSPs) for a multisite application must be coded in a site-aware way. A key aspect of making pages site-aware involves changing the current site context so, for example, a certain area of a page on site A can display products from site B.

To change the site context on a portion of a page, you use the `atg.droplet.multisite.SiteContextDroplet`. This servlet bean takes a site ID as an input parameter, changes the site context to the site with that site ID, and renders output using the new site context. It then restores the site context to what it was before the droplet was invoked, so the remainder of the page is rendered using the original site context.

In addition to `SiteContextDroplet`, the `atg.droplet.multisite` package includes a number of other servlet beans for building site-aware JSPs. These include:

- `GetSiteDroplet` -- Given a site ID, returns the corresponding `Site` object. Properties of the site can then be displayed or used in by other page code.
- `SiteLinkDroplet` -- Given a site ID and a path, generates a link to another site.
- `SitesShareShareableDroplet` -- Tests to see if a given shareable type is shared between the current site and one or more additional sites.
- `SiteIdForItemDroplet` -- Given a repository item, returns a site that the item is associated with (i.e., one of the sites in the item's `siteIds` property). The actual site returned depends on the values of input parameters that are used to give relative weighting to the sites in the `siteIds` property.

Site Scope

In addition to the multisite-specific servlet beans mentioned above, several other servlet beans and form handlers have properties for determining the sites they apply to. The sites are specified using one of these properties:

- `siteIds` – an array of site IDs
- `siteScope` – one of the following values:
  - `all` – applies to all sites
  - `current` – applies to the current request's site
  - a shareable type ID – applies to sites that share this shareable type with the current site; for example., `atg.ShoppingCart`

Only one of these properties should be set. If both are set, `siteScope` is ignored. If neither property is set, the component is applied to all sites.

Servlet beans that have these properties also have `siteIds` and `siteScope` input parameters that can be used to override the property values on an individual page. Form handlers with these properties can override the configured values using hidden input fields.

3 Multisite Support in ATG Products

Every product in the ATG Commerce suite supports multiple sites. This chapter highlights key multisite features in each product. It includes the following sections:

Note that the information in this chapter focuses on major features and is not comprehensive. For more information about multisite support in a specific ATG product, see the documentation for that product.

ATG Commerce

Multisite support affects virtually all areas of ATG Commerce. This section describes the multisite features in various parts of ATG Commerce.

Product Catalog

The Product Catalog repository allows you to associate specific catalog items (catalogs, categories, products, and SKUs) with specific sites. Each of these items has a `siteIds` property that lists the sites the item is associated with. If an item is not associated with a specific site, it does not appear in that site's catalog hierarchy.

ATG Merchandising users can set up catalogs and associate them with sites. The `siteIds` properties of categories are populated based on these associations. When the catalog repository is deployed, Catalog Maintenance System services populate the `siteIds` properties of products and SKUs, based on the categories they are associated with.

Setting a User's Catalog and Price Lists

ATG Commerce includes logic for setting a user's catalog and price lists on a given site. To do this, it adds `CatalogProfilePropertySetter` and `PriceListProfilePropertySetter` components to the `profilePropertySetters` property of the `/atg/dynamo/servlet/dafpipeline/ProfilePropertyServlet` component in the DAF servlet pipeline:

```
profilePropertySetters+=/atg/userprofiling/CatalogProfilePropertySetter,\
                        /atg/userprofiling/PriceListProfilePropertySetter
```

For the profile's `catalog` property, the `CatalogProfilePropertySetter` calls the `determineCatalog()` method of the `/atg/commerce/catalog/CatalogTools` component. This method invokes the `/atg/commerce/util/ContextValueRetriever` component. If this component's `useProfile` property is `false` (the default), the following logic is applied:

- If there is a current site (i.e., the application has multisite support enabled), use the value of the `defaultCatalog` property of the `siteConfiguration` item for the current site.
- Otherwise, use the value of the `defaultCatalog` property of the `CatalogTools` component.

Similar logic is used to set the profile's `priceList` and `salePriceList` properties. For price lists, the `PriceListProfilePropertySetter` component calls the `/atg/commerce/pricing/priceLists/PriceListManager` component's `determinePriceList()` method, which calls `ContextValueRetriever`.

Note that ATG Business Commerce can run in a multisite environment, but does not actually support multiple sites. ATG Business Commerce disables `CatalogProfilePropertySetter` and `PriceListProfilePropertySetter` to prevent catalog and price list settings stored in customer profiles from being overridden.

Shopping Cart

ATG Commerce includes a `ShareableType` component, `/atg/commerce/ShoppingCartShareableType`, which is of class `atg.multisite.NucleusComponentShareableType`. This component is configured to manage the sharing of the shopping cart and the product comparisons list:

```
paths=/atg/commerce/ShoppingCart,\
      /atg/commerce/catalog/comparison/ProductList
```

If you want sharing of these components to be managed independently, you can create separate `ShareableType` components for them. To do this:

1. Remove the `ProductList` component from the `paths` property of `ShoppingCartShareableType`.
2. Create a separate component of class `NucleusComponentShareableType` and set its `paths` property to `/atg/commerce/catalog/comparison/ProductList`.
3. Register the new component with the `/atg/multisite/SiteGroupManager` component by adding it to that component's `shareableTypes` property. For example:

   ```
   shareableTypes+=/myShareableTypes/ProductListShareableType
   ```

Orders

When it adds a `CommerceItem` to an order, the `atg.commerce.order.purchase.CartModifierFormHandler` sets the value of the item's `siteId` property. The value is typically the site that was current when the item was added to the order. In some cases, though, it may be a different site; for example, if the `CommerceItem` is a cross-sell associated with a different site, this property will contain the site ID of the associated site.

Order items in the Order Repository have two properties for tracking the sites they are associated with:

- `creationSiteId` – Contains the site ID of the first `CommerceItem` added to the order.
- `siteId` – If the order has not been submitted, this property contains the site ID of the last `CommerceItem` added to the order. `CartModifierFormHandler` updates this property each time a `CommerceItem` is added to the order or the quantity of an item is updated. When the order is submitted, `CommitOrderFormHandler` or `ExpressCheckoutFormHandler` sets this property to the site ID of the site from which the order was submitted.

The `OrderLookup`, `ScheduledOrderLookup`, and `ApprovalRequiredDroplet` servlet beans have `siteIds` and `siteScope` input parameters to determine which sites to look up orders for. (See Site Scope for information about these parameters.)

Promotions and Coupons

Promotions and coupons are site-aware. Site-awareness can be part of the condition or the offer. For example:

- Site-aware condition: A customer who buys $100 worth of merchandise on a given site receives free shipping.
- Site-aware offer: A customer who buys $100 worth of merchandise from one site receives a 15% discount on a related site.

Coupons can include site constraints that control which sites the coupons can be claimed on. Coupon site constraints do not affect where the associated promotions are valid.

Gift Lists

The `/atg/commerce/gifts/GiftlistManager` component has a `siteScope` property that controls how gift lists are shared between sites. Each gift list is associated with a single site, and items from other sites can be added to a gift list if that site is compatible, based on the value of `siteScope`. For example, if `siteScope` is set to the `atg.ShoppingCart` shareable type ID, an item from another site can be added to a gift list only if the other site shares a shopping cart with the gift list's site.

The `/atg/registry/CollectionFilters/GiftlistSiteFilter` component (used by the `/atg/commerce/collections/filter/droplet/GiftlistSiteFilterDroplet` component) provides site-based filtering of gift and wish lists and the items in these lists. This filtering mechanism is especially useful for wish lists; a customer can have only one wish list, so filtering is needed to restrict the items shown by site.

Search

Search results can be returned from all available sites, or can be limited to items from specific sites. This flexibility is available both with ATG's basic search capability (based on `atg.commerce.catalog.SearchFormHandler`) and with ATG Search (see below).

ATG Search and Search Merchandising

ATG Search and Search Merchandising have been enhanced to support multisite environments.

ATG Search

Indexing and querying are site-aware:

- Structured content (repository data) such as the product catalog is made site-aware using the `siteIds` property of the repository items. Unstructured content (file-system data) can be made site-aware using index-only metadata tags.
- Content sets are associated with sites in the Site Administration utility. If a content set contains site-specific data, the index for that content set does not include data for sites that are not associated with the content set.
- The `QueryRequest` object is configured with properties that allow you to return results from all sites or restrict results to only specific sites.

Search Merchandising

Facets, search configurations, and search tests are site-aware:

- Facets can be applied across all sites, within a site, or within a category.
- Search configurations now support site as a dimension. In additionally, a `site` property is defined within Result Prioritization, allowing users to assign weight to a given site.
- Users can select a site to determine site context for a search test.

ATG Service

ATG Service products have been enhanced to support multisite environments.

ATG Commerce Service Center

ATG Commerce Service Center (CSC) supports all of the multisite features in ATG Commerce: site-aware catalog data, orders, promotions, and so on. In addition, CSC itself includes a number of multisite features to help agents support multisite applications. These features include:

- The ability to configure a default site, which is used to set the site context within the Commerce tab. This default applies to all agents, and does not vary by segment or user.
- A site picker that allows an agent to specify the current site context. When a new site is selected, CSC updates the catalog, price lists, and shopping cart to reflect the new site context.
- The ability to specify sites as criteria for constraining catalog browsing and searching.
- Inclusion of the site ID in Click to Call requests. When a request is received, CSC changes the site context to the specified site, so the agent has the same site context as the customer.

ATG Knowledge and ATG Self Service

In earlier releases, ATG Self Service provided multisite functionality through the use of a brand parameter. In ATG 10.0.2 and above, ATG Self Service uses multisite support configured through the Site Administration utility. The use of brand segments as sites for security purposes is still supported.

ATG Commerce Reference Store

ATG Commerce Reference Store (CRS) provides a sample multisite application that includes three sites: ATG Store US, ATG Store Germany, and ATG Home. These sites are related in the following ways:

- ATG Store US and ATG Home share a shopping cart, product comparisons, and gift and wish lists. Some catalog items are available on both sites, but each site also has items not available on the other site.
- ATG Store US and ATG Store Germany have identical catalogs, but use different price lists because ATG Store US prices are in US dollars and ATG Store Germany prices are in Euros. These sites do not share a shopping cart, product comparisons, or gift and wish lists.

The sites are preconfigured in the Site Repository:

- All three sites use the default ATG site category.
- CRS adds application-specific properties, such as various site e-mail addresses, to the `siteConfiguration` item type.

The sites are designed to use the same front-end code and to take advantage of cross-site data sharing and navigation:

- All of the stores use the same JSP pages, with some styling differences.
- When a product from a site other than the current site appears in a gift list, product comparison, shopping cart, etc., the site that the product is from is indicated with the site name and site icon. In displays of cross-sells and related products, these indicators also function as cross-site links.

- The sites use a URL strategy based on virtual context roots. (See URL Management for more information.)

In addition to the product catalog and shopping cart, the application includes a variety of other types of site-aware data and functionality:

- Site-aware promotions, including one that gives free shipping to users checking out from ATG Home, and one that gives a 20% discount on the highest-priced item from ATG Home.
- A targeter that uses site override rules to display different items on each site.
- A user segment with different age criteria on different sites.
- The ability to search for items on the current site only or on all sites in the same sharing group.

Other Products

The following products have minor changes to provide support for multisite applications.

ATG Content Administration

Multisite support requires ATG Content Administration (CA). The Site Repository is versioned and must be deployed.

When you deploy from your CA environment to a staging or production environment, assets for all sites are deployed. You cannot selectively deploy assets by site.

ATG Customer Intelligence

The ATG Customer Intelligence (ACI) Data Warehouse and the metadata model include a site dimension for tracking site information. The site information can then be used in reports. The reports include filter controls for limiting the data to specific sites. Some reports support grouping by site.

ATG Outreach

Users can choose to have a campaign run on all sites running on an ATG instance or on a single site. Campaign events, actions, and reports are site-aware.

ATG Campaign Optimizer

Users can choose to have a test run on all sites running on an ATG instance or on a single site.

Part II: Multisite Roadmap

This section provides a roadmap for the decisions and tasks involved when configuring and coding an application for multisite functionality. It covers both creating new multisite applications as well as modifying existing applications to incorporate multisite functionality. If you have an existing application, this chapter assumes you have already migrated it to the latest ATG release, including updating the schema, so that you are starting with a non-multisite application with zero registered sites.

This section contains the following topics:

Additionally, this section uses examples from ATG Commerce Reference Store (CRS) to illustrate the information it provides. The Commerce Reference Store installation includes all of the CRS-specific source code, in the following locations:

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

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

- More general functionality is found in:

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

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

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

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

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

4 Configuring the Site Administration UI

This chapter describes tasks an IT or developer resource may need to do to prepare the Site Administration UI for use by those who will configure sites. It includes the following sections:

Configuring Site Administration Access Rights

Before they can work in the Site Administration UI, Site Administrators must be given the `siteAdminUser` or BCC administrator role. For detailed instructions on how to do this, see Site Administration Access Rights.

Defining Custom Site Properties

This section discusses custom site properties, what they are and how they are created.

Site Properties Default

The properties you can set for each site depend on the ATG applications and modules you are running. For example, the ATG platform includes site properties that define a site's name, description, ID and so on, while ATG Commerce adds site properties that define the default catalog, price list, and sale price list for a site. If these properties are not sufficient for your application's requirements, you can add custom properties to the site configuration. For example, CRS adds custom properties that store the base CSS filename for each site, the number of days a product is considered new, the e-mail address that order confirmation e-mails come from, and so on. For ATG 10.0.2, site configuration properties are added by:

- The ATG platform
- ATG Commerce
- ATG Search
- ATG Self Service

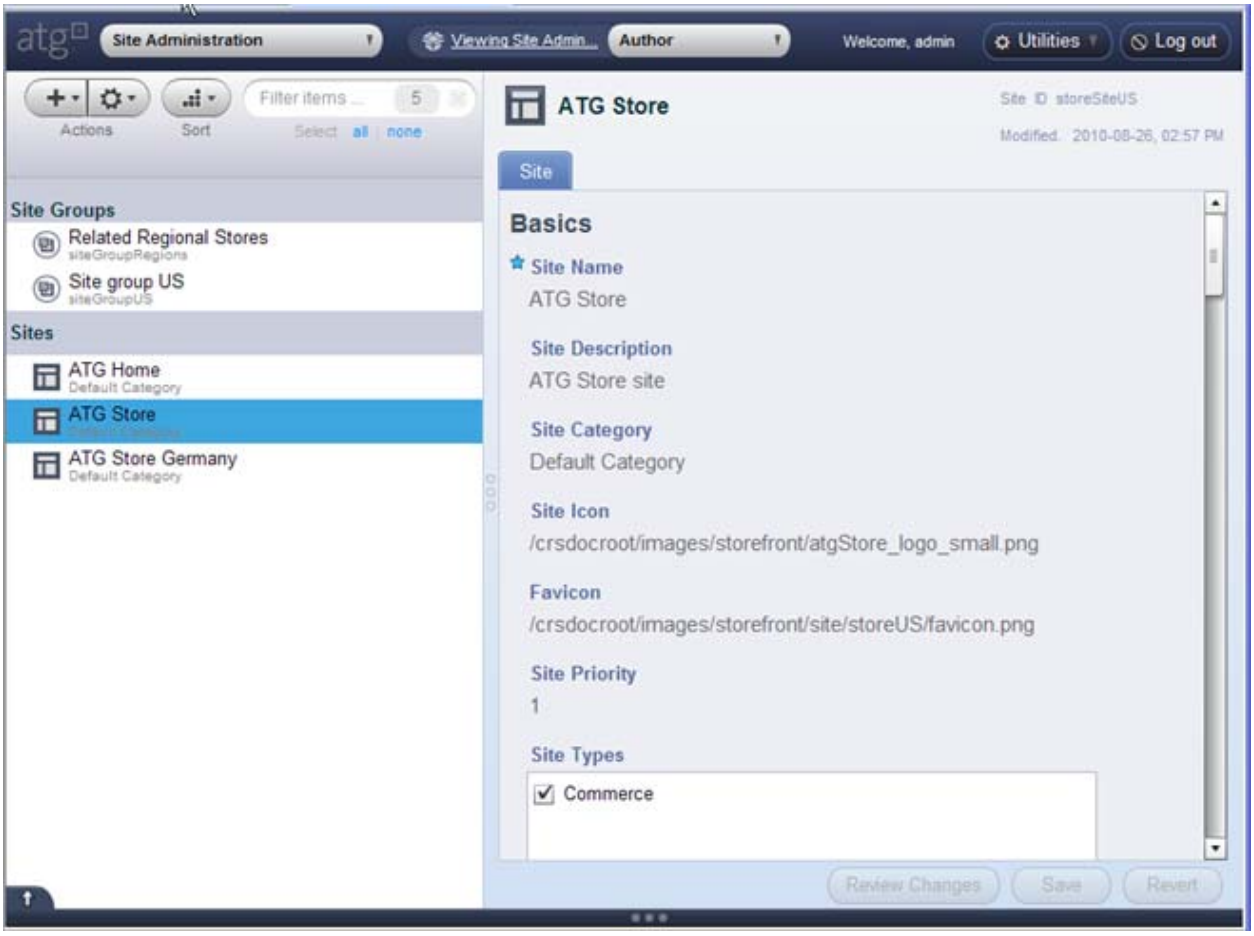
- ATG Commerce Reference Store

For details on each application's properties, refer to the documentation for the individual application.

Adding Custom Properties to the Site Configuration

The method for adding a custom property varies depending on whether or not you want to:

- Specify a non-default editor for the property, for example, a pre-populated list of choices for editing a property of type `string`, rather than the text box that is used by default. Default editors are provided for string, integer, float, date, time stamp, enumerated, enumerated string, Boolean, repository item, and collection data types.
- Define the property as hidden, required, or read only for some of your sites, but not all of them.
- Specify that the property appears on a tab other than the Site tab, or in a grouping with other similar properties (for example, General, Payment, or Shipping properties). The following illustration shows the Site tab for the ATG Store US site in CRS:



If your custom property requires any of these characteristics, you must go through the full process of creating the property along with a custom site category that manages the property's display and editing in Site Administration. If it is acceptable for your property to appear on the Site tab, use the default editor for its type, and have the same settings for hidden, required, or read only across all sites, then the process is simplified through the use of the default site category. This is the approach CRS uses to control the display of its custom properties and it is described below. For more details on creating custom properties that also require custom site categories, see Defining Site Categories.

Adding Custom Properties Using the Default Site Category

The default site category is configured to be inclusive, meaning that it will display, on the Site tab and using default editors, any properties defined for the `siteConfiguration` item type (the exception to this rule is any property that is defined as `hidden` in the repository definition). This configuration allows you to more easily add custom properties, without the overhead of configuring a new site category. To add a custom property to the site configuration, using the default site category, you must:

1. Add the property to the `siteConfiguration` item type in an `/atg/multisite/siteRepository.xml` definition file stored in

<ATG10dir>/home/localconfig. You can add the property to an existing table or create a new table to store the property.

2. If your property values need to be translated, you should create an entry for the property in an existing resource bundle file, such as /atg/projects/store/multisite/StoreSiteRepositoryTemplateResources.properties, and put it in <ATG10dir>/home/locallib. Alternatively, create a new resource bundle, if necessary, for the property, making sure it resides in your CLASSPATH. If your property does not require translating, you can enter literal strings in the repository definition.
3. Update the database to reflect the new property.
4. Rebuild the ATG EAR files for your application and restart the ATG servers.

When you update the database to reflect the new property, you must take the following into account:

- Because all multisite environments use CA, you must update both the publishing and production schemas. If you are using staging or preview servers, you must update their schemas as well.
- If you are using a switch deployment, you must update both the switchingA and switchingB schemas (the switchingCore schema doesn't store site configuration data). See the *ATG Content Administration Programming Guide* for more information on switch deployments.
- If you are adding a new table to store the property, the table must have an ID column in all schemas and this column must be populated with valid site IDs. In other words, this is an auxiliary table that is linked to the site_configuration table through site IDs. In the publishing schema, the table must also have any required ATG Content Administration columns. See the *ATG Content Administration Programming Guide* for more details.
- If you are adding a property to an existing table, you only need to add a column that represents the property to the table in the database, keeping in mind that the table may appear in multiple locations, depending on your environment, as described above.
- If your new property's value is a reference to a new item type, you will have to define a primary table for the new item type in all the relevant schemas. In the publishing schema, this table must include the full complement of ATG Content Administration versioning columns described in *Create and Install the Versioned Schema* in the *ATG Content Administration Programming Guide*.

The following excerpts from the configuration files for CRS show how to add a table, crs_site_attribute, with a number of new site configuration properties. This example assumes an environment that includes CA and has a switching database, so this table must be added to the publishing, switchingA, and switchingB schemas.

Excerpt from /atg/multisite/siteRepository.xml from ATG Commerce Reference Store

This excerpt defines a new table, crs_site_attribute, with CRS-specific site configuration properties.

```
<table name="crs_site_attribute" type="auxiliary" id-column-name="id">

  <property name="resourceBundle" data-type="string"
            column-name="resource_bundle" display-name-resource="resourceBundle"
            category-resource="categoryPresentation">
    <attribute name="resourceBundle"
      value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

  <property name="newProductThresholdDays" data-type="int"
            column-name="prod_threshold"
            display-name-resource="newProductThreshold"
            category-resource="categoryCommerce">
    <attribute name="resourceBundle"
       value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

  <property name="defaultPageSize" data-type="int" column-name="page_size"
            display-name-resource="defaultPageSize"
            category-resource="categoryPresentation">
    <attribute name="resourceBundle"
       value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

  <property name="cssFile" data-type="string" column-name="css_file"
            display-name-resource="cssFile"
            category-resource="categoryAppearance">
    <attribute name="resourceBundle"
       value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

  <property name="defaultCountry" data-type="string"
            column-name="default_country_code"
            display-name-resource="defaultCountry"
            category-resource="categoryLocalization">
    <attribute name="resourceBundle"
       value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

  <property name="emailAFriendEnabled" data-type="boolean"
            column-name="emailafriend" display-name-resource="emailAFriendEnabled"
            category-resource="categoryEmail">
    <attribute name="resourceBundle"
       value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

  <property name="backInStockFromAddress" data-type="string"
            column-name="backinstock_addr"
            display-name-resource="backInStockFromAddress"
```

```
          category-resource="categoryEmail">
    <attribute name="resourceBundle"
       value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

  <property name="newPasswordFromAddress" data-type="string"
          column-name="newpass_addr"
          display-name-resource="newPasswordFromAddress"
          category-resource="categoryEmail">
    <attribute name="resourceBundle"
       value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

  <property name="orderConfirmationFromAddress" data-type="string"
          column-name="orderconfirm_addr"
          display-name-resource="orderConfirmationFromAddress"
          category-resource="categoryEmail">
    <attribute name="resourceBundle"
       value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

  <property name="orderShippedFromAddress" data-type="string"
          column-name="ordershipped_addr"
          display-name-resource="orderShippedFromAddress"
          category-resource="categoryEmail">
    <attribute name="resourceBundle"
      value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

  <property name="promotionEmailAddress" data-type="string"
          column-name="promo_addr" display-name-resource="promotionEmailAddress"
          category-resource="categoryEmail">
    <attribute name="resourceBundle"
       value="atg.projects.store.multisite.StoreSiteRepositoryTemplateResources"/>
  </property>

</table>
```

/atg/projects/store/multisite/StoreSiteRepositoryTemplateResources.properties

This example defines the resource bundle that properties in the `crs_site_attribute` table rely on.

Note: This resource bundle contains more than just the resources required by the `crs_site_attribute` table.

```
# Resource strings for SiteRepository
#
atg_flags=i18n,l10n
```

```
categoryCommerce=Commerce
categoryBasics=Basics
categoryPresentation=Presentation
categoryAppearance=Appearance
categoryEmail=Email
categoryLocalization=Localization

catalogId=Catalog Id
listPricelistId=List price list Id
salePricelistId=Sale price list Id

resourceBundle=Resource Bundle
newProductThreshold=New Product Threshold Days
defaultPageSize=Default Page Size
cssFile=Path to CSS File
largeSiteIcon=Large Site Icon
defaultCountry=Default Country

emailAFriendEnabled=Email a Friend Enabled
backInStockFromAddress=Back in Stock From Address
newPasswordFromAddress=New Password From Address
orderConfirmationFromAddress=Order Confirmation From Address
orderShippedFromAddress=Order Shipped From Address
promotionEmailAddress=Promotion From Address

billableCountries=Billable Country Codes
shippableCountries=Shippable Country Codes

nonBillableCountries=Nonbillable Country Codes
nonShippableCountries=Nonshippable Country Codes
```

SQL for adding crs_site_attribute to the publishing schema

This example shows the SQL for adding the `crs_site_attribute` table to the publishing schema.

```
CREATE
    TABLE CRS_SITE_ATTRIBUTE
    (
        ASSET_VERSION NUMBER(19) NOT NULL,
        ID VARCHAR2(40) NOT NULL,
        RESOURCE_BUNDLE VARCHAR2(254),
        PROD_THRESHOLD NUMBER(10),
        PAGE_SIZE NUMBER(10),
        CSS_FILE VARCHAR2(254),
        LARGE_SITE_ICON VARCHAR2(254),
        DEFAULT_COUNTRY_CODE VARCHAR2(2),
        EMAILAFRIEND NUMBER(1),
```

```
        BACKINSTOCK_ADDR VARCHAR2(40),
        CUSTSERVICE_ADDR VARCHAR2(40),
        NEWPASS_ADDR VARCHAR2(40),
        ORDERCONFIRM_ADDR VARCHAR2(40),
        ORDERSHIPPED_ADDR VARCHAR2(40),
        PROMO_ADDR VARCHAR2(40),
        CONSTRAINT CRS_SITE_ATTR_P PRIMARY KEY (ASSET_VERSION, ID)
    )
```

SQL for adding crs_site_attribute to the switchingA and switchingB schemas

This example shows the SQL for adding the `crs_site_attribute` table to the switching schemas.

```
CREATE
    TABLE CRS_SITE_ATTRIBUTE
    (
        ID VARCHAR2(40) NOT NULL,
        RESOURCE_BUNDLE VARCHAR2(254),
        PROD_THRESHOLD NUMBER(10),
        PAGE_SIZE NUMBER(10),
        CSS_FILE VARCHAR2(254),
        LARGE_SITE_ICON VARCHAR2(254),
        DEFAULT_COUNTRY_CODE VARCHAR2(2),
        EMAILAFRIEND NUMBER(1),
        BACKINSTOCK_ADDR VARCHAR2(40),
        CUSTSERVICE_ADDR VARCHAR2(40),
        NEWPASS_ADDR VARCHAR2(40),
        ORDERCONFIRM_ADDR VARCHAR2(40),
        ORDERSHIPPED_ADDR VARCHAR2(40),
        PROMO_ADDR VARCHAR2(40),
        CONSTRAINT CRS_SITE_ATTR_P PRIMARY KEY (ID)
    )
```

Working with Site Categories

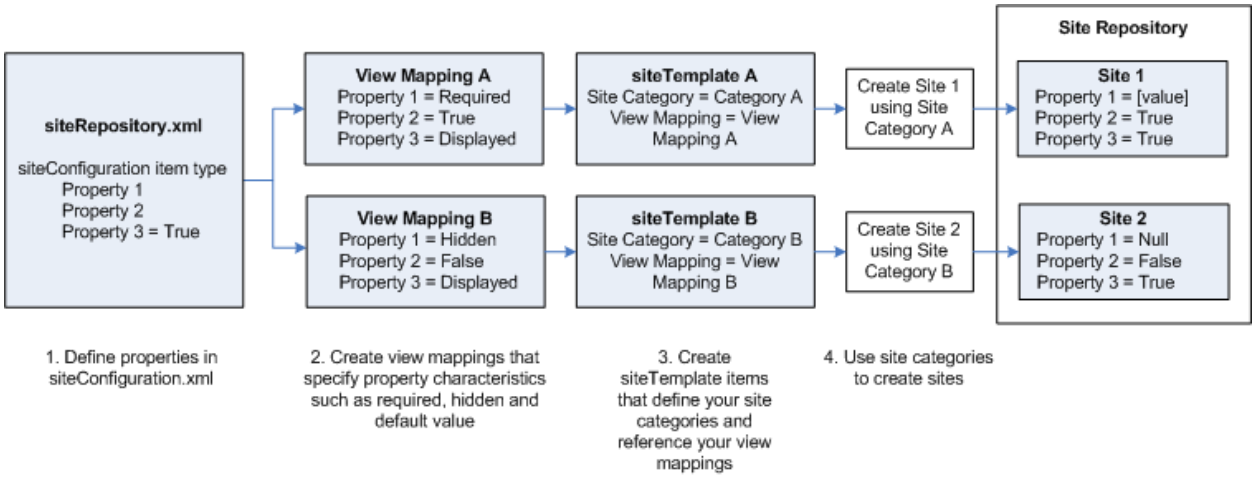
There may be times when the default site category is not sufficient for your needs and you need to create new site categories. You should create a new site category if:

- You want to define custom properties that require a custom site category, as described in Defining Custom Site Properties.
- You want to define a set of characteristics for a particular kind of site, for example, you could have one site category for commerce applications that sell products to customers and another category for customer support applications.

Each site category in Site Administration is backed by a corresponding `siteTemplate` item in the Site Repository that gives the category a name and points to a view mapping. View mappings control the

display of repository item properties in a UI. You can have more than one view mapping refer to the same type of repository item, with each view mapping specifying different display and editing characteristics for the properties of the repository item. For example, one view mapping could mark a property as required while another could mark it as hidden.

It is possible to define characteristics such as required and read only, or to provide a default value for a property, in the `siteConfiguration` XML repository definition file. However, any configuration you specify in the `siteConfiguation` XML definition will apply to all site configurations; that is, all site configuration items in the Site Repository will use that default value. Specifying characteristics like defaults in view mappings allows you to have different defaults for different site categories. When you select a category for your site, you get the corresponding default. For example, consider the following diagram:



In this diagram, the following happens:

1. The `/atg/multisite/siteRepository.xml` defines three properties for the `siteConfiguration` item type, Properties 1, 2, and 3. Property 3 has a default value of `True`. Because it is specified at the XML definition file level, this property will be the same for all sites created in the Site repository.
2. View Mapping A further refines Property 1 to mark it as required, and Property 2 to set its default value to `True`. These values will be reflected in Site Administration. View Mapping A doesn't further refine Property 3 beyond specifying that the property should be visible in Site Administration.

   View Mapping B further refines Property 1 to mark it as hidden, meaning it will not appear in Site Administration, and Property 2 to set its default value to `False`. Like View Mapping A, View Mapping B doesn't further refine Property 3 beyond specifying that the property should be visible in Site Administration.

   Both view mappings configure the display of the site properties in Site Administration. However, because they specify different behavior for the properties, the presentation and behavior of the properties in the Site Administration will also be different.

3. `siteTemplate A` defines Site Category A and specifies that the category should use View Mapping A as its view mapping. `siteTemplate B` defines Site Category B and specifies that the category should use View Mapping B as its view mapping.
4. When you create a site using Site Category A in Site Administration, the following happens:
   - A value for Property 1 is required in order for the site to be saved in Site Administration.
   - The value for Property 2 defaults to `True`, as defined in View Mapping A.
   - The value for Property 3 defaults to `True`, as defined in `siteRepository.xml`.

   When you create a site using Site Category B in Site Administration, the following happens:
   - Property 1 is not visible in Site Administration. Its value in the Site Repository is null. (Note that, while this example does not demonstrate it, it is possible to provide a default value for a property *and* mark it as hidden, in which case the default value will be stored in the Site Repository and the property will not appear in Site Administration).
   - The value for Property 2 defaults to `False`, as defined in View Mapping B.
   - The value for Property 3 defaults to `True`, as defined in `siteRepository.xml`.

In general, you should define the existence of site properties in the `siteRepository.xml` and define the behavior of properties (read only, defaults, required, and so on) in view mappings. The exception to this rule is a property that should behave identically across all sites stored in the Site Repository.

For detailed information on creating custom categories and view mappings that control what you see in Site Administration, see Defining Site Categories. In particular, pay attention to the caveats described in the Important Warning section. For more details on view mappings themselves, see the *ATG Content Administration Programming Guide*.

Configuring ShareableType Components

You use `ShareableType` components to define which Nucleus components and other objects should be shared by all sites and which should have a separate instance for each site or sharing group. This section discusses the out of the box `ShareableType` configuration and how you might choose to modify it.

Default Nucleus Component Sharing Behavior

Like custom site configuration properties, the `ShareableType` components that are available in Site Administration depend on which ATG modules you are running. ATG Commerce is configured with a registered `ShareableType` component, `/atg/commerce/ShoppingCartShareableType`, which designates the shopping cart and comparison lists as shareable resources and uses `atg.ShoppingCart` as its ID:

```
$class=atg.multisite.NucleusComponentShareableType

# The shareable type ID used by application code
id=atg.ShoppingCart

# The Nucleus components associated with this shareable type
paths=/atg/commerce/ShoppingCart,\
      /atg/commerce/catalog/comparison/ProductList

# Information used to find strings appropriate for localized UIs
displayNameResource=ShoppingCartShareableTypeName
resourceBundleName=atg.commerce.order.UserMessages
```

You register a `ShareableType` component by adding its full pathname to the `shareableTypes` property of the `/atg/multisite/SiteGroupManager` component. ATG Commerce configures the `SiteGroupManager` like this:

```
shareableTypes+=/atg/commerce/ShoppingCartShareableType
```

After a Nucleus `ShareableType` component is created and registered, the components it represents become partitioned at run time such that a single instance of each component exists for each site (note that this is only the first step in the data sharing process; configuring sharing groups is the second and they are described below). The ATG platform services track which components are partitioned and make sure the correct component is affected when changes are made on any given site. For components that aren't affiliated with a `ShareableType`, one instance of the component exists that is shared by all sites. Because of the default configuration of `ShoppingCartShareableType`, each new site you create gets its own shopping cart and product comparisons list.

You further refine how shareable types behave by setting up sharing groups in Site Administration. With a sharing group, you can specify multiple sites that share a single resource. For example, consider CRS, which has three sites: ATG Store US, ATG Store Germany, and ATG Home. CRS configures a sharing group, `Site group US`, that contains ATG Store US, ATG Home and a reference to the `ShoppingCartShareableType` component's ID, meaning ATG Store US and ATG Home share a single shopping cart and a single comparison list. ATG Store Germany is not part of this group, so it has its own shopping cart and comparison list.

If you want all sites you create to share the same shopping cart and comparison list, you must modify either `ShoppingCartShareableType` or `SiteGroupManager`. To remove data partitioning for *either* the shopping cart or the comparison list, remove its component name from the `ShoppingCartShareableType.paths` property. To remove data partitioning for *both* the shopping cart and the comparison list, un-register the `ShoppingCartShareableType` component by removing it from the `SiteGroupManager.shareableTypes` property.

If you have additional session-scoped Nucleus components whose data you want shared by a subset of your application's sites, you must create and register additional `ShareableType` components that point to them. Note that if your Nucleus components are part of a larger system, you may also need to do additional custom development to make sure that the entire system is site-aware. For example, making the shopping cart site-aware in ATG Commerce also required updates to the entities that support the shopping cart in order for the entire system to work. See the *ATG Programming Guide* for more details.

Using Non-Nucleus ShareableType Components

Non-Nucleus `ShareableType` components allow you to create sharing groups that contain sites that share an abstract concept or some other type of non-Nucleus object. Commerce Reference Store implements the first case, a sharing group of sites that are related by an abstract concept. The remainder of this section describes how this mechanism works.

At any given time in an ATG application, there is a current site. Non-Nucleus shareable types let you determine what other sites share a particular concept with the current site. For example, CRS implements a `ShareableType` component called `Related Regional Stores` that represents the concept of regional peers of the same store. CRS has a sharing group that includes the ATG Store US and ATG Store Germany sites, along with the `Related Regional Stores` shareable type, indicating that ATG Store US and ATG Store Germany are regional peers.

CRS JSP code uses the `/atg/dynamo/droplet/multisite/SharingSitesDroplet` to determine which other sites are in a `Related Regional Stores` sharing group with the current site. When ATG Store US is the current site, this droplet returns ATG Store Germany and vice versa. This approach allows CRS to implement one generic piece of JSP code that renders a widget that allows shoppers to switch among regional peer sites, as shown in this excerpt from `store.war/navigation/gadgets/regions.jsp`:

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

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

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

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

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

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

              <div id="atg_store_regions">
              <h2>
```

```
                <fmt:message key="navigation_internationalStores.RegionsTitle" />
                <fmt:message key="common.labelSeparator"/>
              </h2>

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

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

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

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

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

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

This code renders the following area of the CRS UI:

The CRS approach to regional peers 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 `Related Regional Stores` shareable type component is defined in the `/atg/store/RelatedRegionalStoresShareableType.properties` file found in CRS's `Store.Estore.International` module. Its definition looks like this:

```
$class=atg.multisite.ShareableType

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

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

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

Note: For details on creating non-Nucleus shareable type components, see the *Multisite Request Processing* chapter of the *ATG Programming Guide*.

5 Configuring Multiple Sites

This chapter describes tasks a site administrator must do to create and configure sites and sharing groups. It includes the following sections:

Creating Sites

Once your site configuration properties, site categories, and shareable type components are defined, business users can begin creating sites using Site Administration in the ATG Business Control Center (BCC). When you create a site in Site Administration, you are adding a `siteConfiguration` item to the `Site` repository. Each `siteConfiguration` item has a unique ID that identifies it. Sites are deployed assets and must go through the standard CA process to deploy them to the production instance (create a project for the new sites, create the sites, review the project, and deploy it to production).

Many site configuration properties point to other resources, which may be either repository items like catalogs and price lists, or file-based assets like images and resource bundles. These other resources must exist before the site can be fully functional and tested. Also, business users will need to know how to reference them (for example, they will need the paths to image files and resource bundles).

If you are converting an existing non-multisite application to multisite, you must start by creating a site configuration item that represents the existing application. See Initial Steps to Converting an Existing Non-Multisite Application to Multisite for detailed instructions.

Setting Individual Site Properties

The properties you can set in a site configuration are determined by the modules that are running and the site category you choose to base your site on. In other words, the ATG platform includes a set of properties, ATG Commerce includes additional properties, other ATG products include still more, and you may also have custom properties defined for your sites. Correspondingly, the site category you choose may show some properties, hide others, and define property behavior such as defaults and the editors

you use to set them. You edit site properties in Site Administration. For detailed instructions on using Site Administration, see Site Configuration Properties.

Creating Sharing Groups

Once you have created sites, you can put them into sharing groups. You define sharing groups in Site Administration by creating site groups that share components. For detailed instructions, see Site Groups.

Initial Steps to Converting an Existing Non-Multisite Application to Multisite

If you are converting an existing non-multisite application to multisite, you must start by creating a site in Site Administration that represents the existing application, and then associating that site's ID with the existing assets. The section below describes this process. It assumes that the existing application and its assets will become one registered site in your multisite application.

Note: Converting an existing application to multisite requires additional configuration and coding to make sure your application's components and JSP pages behave properly in a multisite environment. The information provided in this section represents the first steps in converting an application to multisite and shows you how to create a foundation for the rest of the multisite development process.

Registering a Site and Associating It with a Catalog

In this procedure, you will create a site in Site Administration that represents the existing site, and then associate that site with the existing catalog.

Note: These steps assume that you have upgraded your application to the latest release, including migrating your schemas, running a full deployment in CA, and running Catalog Update Service. For detailed information on these steps, see the ATG migration documentation for your specific version of ATG software.

To register a site and associate it with a catalog:

1. In the BCC, expand Site Administration, then click Manage Site Assets.
2. Enter a name and optional description for the project and click Continue.
3. Create a new site and assign the existing application's catalog as the default catalog for the new site. During this step, the site ID for the new site is assigned to the existing catalog. The same ID is also assigned to any of the catalog's categories and sub-catalogs.
4. At this point, existing catalog folders will appear in ATG Merchandising as unsited items. If you want to keep them, these folders also need a site ID. Follow the steps below to assign a site ID to your catalog folders:

- In the BCC, expand Merchandising, then click Manage Commerce Assets.
- Enter a name and optional description for the project and click Continue.
- Navigate to Site Catalogs, then to Unsited Items.
- Click the split screen icon at the bottom left of the window:

- In the bottom portion of the split screen, navigate to Site Catalogs.
- Drag a catalog folder from the top pane to the site you want it affiliated with in the bottom pane. The catalog folder is now affiliated with the site and has that site's ID assigned to it.

5. Deploy the project. During deployment, the site ID you assigned to the catalog is propagated from the catalog down to its constituent products and SKUs by the `CatalogMaintenanceService`, which runs automatically on deployment.

   Note: For more information on the `CatalogMaintenanceService`, see the *ATG Commerce Programming Guide*.

Updating Existing Data

This section describes how to associate a site ID with any existing data (orders, gift lists, and so on). This step is necessary so that ATG's multisite features can identify data that belongs to the existing application. You create these associations by populating a number of site ID columns in the ATG database with the ID of the site that corresponds to your existing application. The schemas for ATG products include site ID columns for any tables that contain data that require site association (generally, although not always, these columns are named `site_id`). For example, to indicate that gift lists belong to an existing application, you populate the `dcs_giftlist` table's `site_id` column.

The site ID columns fall into these categories:

- Columns that are populated for you as you use various ATG tools like ATG Merchandising and Site Administration.
- Columns that must be manually populated for your existing data. ATG recommends using database tools to update these columns.
- Columns that are populated by ATG migration scripts.
- Columns that do not require a value and, depending on your application's requirements, may not need to be populated at all.

The following section describes the various site ID columns, their intended uses, and how they get populated. For versioned tables that are manually populated, you need only populate the publishing schema. The other schemas will be populated upon deployment.

The ATG Platform

Table(s)	Description of site ID column(s)
das_seo_sites	The site_id column of this table stores the sites property of the SEOTags repository items. This property allows you to associate one or more sites with an SEOTags item, effectively allowing you to define different page titles, meta keywords, and descriptions for each site. If you need to differentiate page titles, keywords, or descriptions for your sites, manually populate this column. Otherwise, you can ignore it.

ATG Campaign Optimizer

Table(s)	Description of site ID column(s)
abt_test	A test can be assigned to all sites or a single site. The site_id column in this table stores the site ID value when a test is assigned to a single site. If you have any undeployed tests from your existing application that need to be associated with a single site, you should edit them in ATG Campaign Optimizer. Saving the edits will update this table with site ID information.

ATG Commerce

Table(s)	Description of site ID column(s)
dbcpp_sched_order	The site_id column in this table stores site membership data for scheduled order objects. ATG recommends manually populating all scheduled order objects with the site ID that corresponds to your old site. Also, scheduled orders that are based on existing orders should use the same site IDs as the existing orders.
dcs_catalog_sites	The site_id column in this table stores site membership data for catalog items in the Catalog repository. This column is populated when you assign one or more catalogs to a site in Site Administration or ATG Merchandising. After assigning a catalog to a site, the site ID columns for the catalog's child categories, products, and SKUs must also be populated with this same ID. This process is handled by the Catalog Maintenance Service. See Registering a Site and Associating It with a Catalog for details.
dcs_category_sites dcs_product_sites dcs_sku_sites	The site_id columns in these tables store site membership data for categories, products, and SKUs, respectively. These columns must be populated with the same site IDs that are associated with the ancestor catalogs. This process is handled by the Catalog Maintenance Service. See Registering a Site and Associating It with a Catalog for details.

Table(s)	Description of site ID column(s)
dcs_catfol_sites	The site_id in this table stores site membership data for catalog folders. This column should be populated so that existing catalog folders appear and perform as expected in ATG Merchandising. See Registering a Site and Associating It with a Catalog for details on how to do so.
dcs_giftitem	The site_id column in this table stores the site a gift item is associated with. To avoid potential display issues, this property should be manually populated with the site ID that corresponds to the old site. **Note:** The exception to this rule is if you only have one catalog for all your sites, in which case you can skip this step.
dcs_giftlist	The site_id column in this table stores the site from which a gift list was created. ATG recommends manually populating this column for all gift lists but not for wish lists. No site ID should be stored for wish lists because there is only one wish list per profile and it should not be associated with any specific sites. To retrieve a list of gift lists that should be marked with a site ID, query the dcs_user_giftlist table. For a list of wish lists, query the dcs_user_wishlist table. The following SQL statement sets the site ID for gift lists only to MySiteId: `Update dcs_giftlist set site_id = 'MySiteId' where id in (select giftlist_id from dcs_user_giftlist)` **Note:** The value of the site_id column is used in conjunction with the GiftlistManager.siteScope property to determine which gift lists may be accessed from any given site. See *Gift and Wish Lists in a Multisite Environment* in the *ATG Commerce Programming Guide* for more information.
dcs_prm_sites	The site_id column in this table stores site constraints for promotions. If this property is empty, the associated promotion is valid on all sites. ATG recommends using ATG Merchandising to configure promotions that need to be constrained to specific sites. Manual population of this column is not necessary.
dcspp_item	The site_id column in this table represents the site that a shopping cart item is associated with. ATG recommends manually populating the site_id column for any orders you intend to keep after you convert your application to multisite.

Table(s)	Description of site ID column(s)
dcspp_order	The dcspp_order table represents customer orders. This table includes two site ID-related columns, creation_site_id and site_id. The DCS module populates the creation_site_id column when the first item with a non-null site ID is added to the order. By contrast, the site_id column is modified over the lifespan of the order. During the course of normal site usage, the site_id column is updated with the current site ID: - Each time a new item is added to the order, including when the quantity of an existing item in the cart is updated. - When the order is submitted. ATG recommends manually populating both the creation_site_id and site_id columns with the site ID that corresponds to your old site.

ATG Customer Intelligence

Table(s)	Description of site ID column(s)
See list in description	ATG Customer Intelligence adds a number of site ID-related columns to existing tables. All of these columns are required and must be populated with some value. For existing applications, the ATG migration script sets the site ID columns to 0 which represents an unspecified site. If you want to associate an application's existing data with a site ID, you must manually populate the site_id column values. The affected tables and their columns are listed below. arf_site_visit(site_id) arf_registration(site_id) arf_line_item(submitted_site_id, origin_site_id, item_site_id) arf_promotion_usage(site_id) arf_return_item(submitted_site_id, item_site_id) arf_query(site_id) arf_view_content(site_id) arf_search_site_visit(site_id)

ATG Outreach

Table(s)	Description of site ID column(s)
cc_landing_page	Each row of this table represents a link that will be presented to an e-mail recipient and, as such, the row must refer to a defined page in a defined domain. If the ATG Outreach user enters an absolute URL, then the domain for the link is known. However, because individual JSP pages can be used by many sites in a multisite environment, if the ATG Outreach user opts to select a JSP file asset for the landing page, it is necessary to also record the identity of a site in order to establish the domain for the link. The site_id column provides this data. ATG recommends editing this data in ATG Outreach. **Note:** If the ATG Outreach customer provides an absolute URL, the site_id value is null.
epub_process	A campaign can be assigned to all sites or a single site. The site_id column in this table stores the site ID value when a campaign is assigned to a single site. ATG recommends editing campaigns in ATG Outreach, during the course of which this table will be updated. Campaigns that are assigned to all sites have a value of -1.

ATG Search

Table(s)	Description of site ID column(s)
srch_site_cntnt	The site_id column in this table stores site membership data for search content sets. This column is populated by ATG Search Administration when you configure your search content sets. No manual population is necessary.

Note: You can specify site IDs for unstructured content in user-defined meta properties that you create in ATG Search Administration. See the *ATG Search Administration Guide* for details on how to add site IDs for unstructured content.

Custom Application Data

In addition to the tables listed in the sections above, you should also consider whether your application has custom data that should be made site-specific. For example, CRS adds the following tables that contain site-specific data:

Table(s)	Description of site ID column(s)
crs_back_in_stock	The site_id column of this table represents the site a customer was on when an out-of-stock item was added to the cart. This data allows you to create a back-in-stock e-mail notification that uses an appropriate site context.

Table(s)	Description of site ID column(s)
crs_promo_content	This table holds data that supports promotional content items, such as category and product URLs that the customer can click when viewing the promotional content item. The site_id column of this table stores the site the promotional content item is associated with, making it possible to create a link to the associated site's home page In the absence of a category or product link.
crs_store_location	This table holds details, such as store name, address, and phone number, for physical stores. The site_id column of this table indicates which site the physical store is associated with.

6 Multisite Application Development and Configuration

This chapter describes application-level development and configuration tasks that implement multisite features. It includes the following sections:

Assigning Catalogs to Sites

In a multisite commerce application, you must determine how many catalogs you will have and which sites will use them. For example, in CRS, there are two catalogs, `masterCatalog` and `homeStoreCatalog`. ATG Store US and ATG Store Germany both use `masterCatalog`, while ATG Home uses `homeStoreCatalog`.

Assigning Site IDs to Catalog Items

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

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

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

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

Once you have created a site and associated one or more catalogs with it, you use ATG Merchandising to populate and organize the catalogs. Note that, for catalogs that are associated with more than one site, changes made for one site will be reflected in the catalogs for the others. One of the catalogs you create can be specified as the default catalog for the site, described below.

Note: See the ATG Commerce Programming Guide for more information on the `CatalogCompletiontService` and `CatalogMaintenanceService`.

Assigning a Default Catalog to a Site

In Site Administration, you can specify the default catalog for each site. For every request, ATG Commerce sets the customer's `catalog` profile property to the current site's default catalog. This mechanism ensures that the customer will always see a catalog that is correct for the current site (see the ATG Commerce section for more details on how this mechanism works).

You set the default catalog for a site using this option on the Site tab:

When specifying a default catalog, you can choose to create a new catalog or use an existing one. If you choose to create a new catalog, you must subsequently populate it with categories, products and SKUs in ATG Merchandising. Multiple sites can have the same default catalog (for example, in CRS, ATG Store Germany uses the same default catalog as ATG Store US).

Assigning Price Lists to Sites

Typically, most multisite environments use price lists, not SKU-based pricing, to price their items. With SKU-based pricing, you may only specify one price per item. Price lists are more flexible and make it possible to have different prices on different sites for the same SKU.

As you approach your pricing strategy in a multisite application, you should consider items that appear on multiple sites and whether you want the prices for those items to be the same across those sites or different. For example:

- In CRS, ATG Home and ATG Store US both include products in the Home Accents category. The CRS pricing strategy for these products is to use the same price for the products on both sites.
- ATG Store US and ATG Store Germany share a catalog and present the same set of products to the customer. However, ATG Store US prices are in dollars while ATG Store Germany prices are in Euros; therefore these two sites use different price lists.

For cases where you want prices to be consistent across sites, you should specify the same list and sale price lists for those sites. Typically, sites that share a shopping cart should also share price lists. This ensures that the price lists can cover any products that may coexist in the shopping cart at the same time. ATG Home and ATG Store US both use the `List Prices` and `Sale Prices` price lists.

For cases where you want prices to be different across sites, you should specify a different set of price lists for each site. For example, ATG Store US uses `List Prices` and `Sale Prices`, while ATG Store Germany uses `German List Prices` and `German Sale Prices`.

In Site Administration, you specify the default price list and default sale price list for each site. For every request, ATG Commerce uses this information to set the customer's `priceList` and `salePriceList` profile properties based on the current site context. This mechanism ensures that the customer will see the correct price lists for the current site (see the ATG Commerce section for more details on how this mechanism works).

Implementing a URL Strategy

As described previously, a multisite application must be able to quickly determine what site a given request is associated with. To this end, the ATG platform supports two main URL management strategies:

- In a path-based strategy, the domain name is the same for all sites, and the site is keyed off the context root in the URL. For example:

  ```
  http://www.sports.com/baseball/
  http://www.sports.com/football/
  http://www.sports.com/basketball/
  ```

  An individual Web application (WAR file) can have only one context root, so to make it possible for all of these sites to be part of the same Web application, ATG uses a technique called *virtual context roots*. Using virtual context roots, you map the sites' context roots (for example., `/baseball/`, `/football/`) to the actual context root of the Web application. The virtual context root still appears in the browser when a customer accesses the site.
- In a domain-based strategy, each site has a different domain name, and a request is associated with a site based on the domain name. There are two variants of this approach, depending on whether the subdomain changes or the parent domain does. If the parent domain changes, the domain names might be:

  ```
  http://www.baseball.com/
  http://www.football.com/
  http://www.basketball.com/
  ```

If the subdomain changes, the domain names might be:

```
http://baseball.sports.com/
http://football.sports.com/
http://basketball.sports.com/
```

This strategy is more complicated to implement because it requires domain name registration as well as DNS server support. Also, application servers do not, by default, share cookies across domains. To handle this situation, the ATG platform provides a mechanism called session recovery, which is described in detail in the *ATG Programming Guide*.

When developing your multisite application, you must choose a strategy and implement it. CRS implements a path-based strategy that defines `/crs` as the context root for the entire application along with the following virtual context roots for the individual sites:

- `/crs/storeus`
- `/crs/storede`
- `/crs/homeus`

For more details on the strategies themselves, see the URL Management section. For detailed information on implementing the strategies see *Multisite URL Management* in the *ATG Programming Guide*.

Configuring Personalization Assets to be Site-Aware

ATG includes several personalization features that allow you to display dynamic, personalized content to Web site visitors:

- Targeters specify what content to show, to whom, and under what conditions.
- User segments define groups of users who have similar interests or requirements that make them candidates for targeted content.
- Content groups define groups of items that share characteristics that may be of interest to a group of users.

These three types of assets work together. When presenting personalized content, targeters may use user segments to define the audience and content groups to define the content.

In a multisite-enabled environment, the BCC user interface includes extensions to the rule configurations for personalization assets that allow you to make them site-aware. The following table shows the rule extensions for each asset type:

Personalization Asset	Multisite Rule Extensions
User segment	Specify site-specific rules that override the user segment's default rule and allow the user segment to behave differently on different sites. For example, the default rule may specify that a Young user segment consists of customers aged 20 to 30 years old, while a site override rule can specify that on Site A, the Young user segment consists of customers that are 20 to 40 years old. All sites will use the default rule, except for Site A, which will use the override rule.
Targeter	Specify the site on which to search for content. Specify site-specific rules that override the targeter's default rule and allow the targeter to behave differently on different sites.
Content group	Specify the site on which to search for content. Specify site-specific rules that override the content group's default rule and allow the content group to behave differently on different sites.

You add site-aware personalization assets using the Personalization options in the BCC. For existing applications that are being converted to multisite, you can use this same utility to modify existing assets to be site-aware. For detailed information, see the *ATG Business Control Center User's Guide*.

Notes:

- The multisite configuration extensions appear only in the rules. The remaining configuration options for personalization assets are the same in a multisite environment or for an environment with no registered sites.
- For examples of multisite personalization assets from CRS, refer to the *ATG Commerce Reference Store Overview*.
- Personalization assets containing site-specific rules cannot be edited in the ATG Control Center (ACC).

Configuring Scenarios to be Site-Aware

Scenarios can take advantage of the following multisite extensions:

- Multisite awareness is available in many *events* that are recognized by scenarios, including, but not limited to: `LogIn`, `LogOut`, `StartSession`, `PageVisit`, `SlotItemRequest`, and `SubmitOrder`.
- The `siteName` parameter has been added to the `FillSlot`, `EmailNotify`, and `SendEmail` actions to enable site-specific e-mails.
- A scenario may use a targeter, segment, or content group that makes use of multisite rule extensions. The current site dictates which asset rules are run. If the asset has a

site-specific override for the current site, then that rule is run. If not, then the default rule runs.

Note: Scenarios are edited in the ACC.

Setting the Site Scope for Order Lookups

Out of the box, ATG Commerce sets the `/atg/commerce/order/OrderLookup` component's `siteScope` setting to `null`, so that the `OrderLookup` servlet bean returns all orders in a customer's history. You can modify this setting to return only orders for the current site, or only orders for the sites in a sharing group. See *ATG Commerce Guide to Setting Up a Store* for more details.

Configuring Promotions and Coupons to be Site-Aware

In a multisite environment, you can specify that a promotion is valid only on specific sites. For example, CRS includes two site-specific promotions:

- ATG Home – Free Shipping: This promotion is valid on ATG Home only and it gives customers free shipping when they check out from ATG Home.
- ATG Home Save 20% on One Item: This promotion is valid for the ATG Home and ATG Store US sites. It gives the customer a 20% discount on the highest priced item in an order that comes from the ATG Home site (that is, the item has a `SKU.sites` property that contains the ATG Home site ID).

Coupons, by default, can be redeemed on any site, however, they can also be made site-aware. A site-aware coupon can be redeemed only on those sites where its associated promotions are valid.

You make promotions and coupons site-aware using ATG Merchandising. For details on this process, see the *ATG Merchandising Guide for Business Users*.

Setting the Site Scope for Gift and Wish List Activity

ATG Commerce allows you to control which sites are allowed to access any given gift list. This functionality allows more than one site to access the same list, effectively letting the sites share the list. Gift list access is controlled by the `/atg/commerce/gifts/GiftlistManager` component's `siteScope` setting. Out of the box, ATG Commerce sets the `GiftlistManager.siteScope` setting to `all`, so all gift lists are shared by all sites. CRS's `Store.Estore` module modifies this setting and makes the assumption that stores that share shopping carts should also share gift lists, so it sets the `siteScope` property in the `/atg/commerce/gifts/GiftlistManager.properties` file as follows:

```
siteScope=atg.ShoppingCart
```

This setting dictates that the current site can access its own gift lists as well as any gift lists that are affiliated with sites that share a shopping cart with the current site. For example, in CRS, ATG Store US and ATG Home share a shopping cart, therefore, when ATG Store US is the current site, it can access gift lists affiliated with ATG Store US *and* ATG Home. ATG Store US cannot access gift lists affiliated with ATG Store Germany because it does not share a shopping cart with ATG Store Germany.

Note that wish lists are always universal. This is because a customer can only have one wish list, so all sites must be able to access it. ATG Commerce provides filtering functionality that allows you to filter the items in a wish list to show only those that are appropriate for the current site context.

Notes:

- While CRS uses the shopping cart `ShareableType` to control gift list activity, you can use the ID of any `ShareableType` component you have created in its stead.
- See the *ATG Commerce Programming Guide* for more information on using gift and wish lists in a multisite environment.

Configuring ATG Search in a Multisite Environment

This section provides an overview of working with ATG Search in a multisite environment from both an indexing and querying perspective.

Note: This section covers ATG Search only. For information on working with ATG's basic search functionality, refer to the *ATG Search Administration Guide.*

Specifying Content Sets for Sites

When you configure a site in Site Administration, you can set the ATG Search content sets it is associated with. ATG Search manages search queries so that, when you query the index for a given site, you only get back results from content sets that are associated with that site.

For details on creating content sets, see the *ATG Search Administration Guide*. For information on assigning content sets in ATG Knowledge, refer to the *ATG Knowledge Configuration Guide.* For information on assigning content sets in ATG Self Service, refer to the *ATG Self Service Configuration Guide*.

ATG Search Querying in a Multisite Environment

When setting up ATG Search, you configure components that determine the search environment to be queried. In a multisite environment, ATG Search uses this configuration in conjunction with site context information to automatically generate query constraints that limit search results to those that are appropriate for both the search environment and the site context.

Out of the box, ATG Search uses a search environment that is appropriate for most production systems. For details on this configuration and how to modify it, see the *ATG Search Query Guide*. For an example of how to pass site context information to ATG Search in a JSP page, see Implementing a Multisite Search Control.

Implementing Facets in a Multisite Environment

Environments that have ATG Merchandising and ATG Search installed can use a feature called facets. A facet is a way of constraining searches to return groupings of products based on a property value. When you implement facets in site pages, you see groupings that appear as a range of selections named for property values or letters in the alphabet that represent them. By clicking an A-C selection, you can view products, for example, that have a manufacturer name that begins with those letters.

Depending on how you define your facets, they may be available on individual sites only or across multiple sites. For details on creating facets for a multisite application, see the *ATG Merchandising Guide for Business Users*. For information on designing pages that use facets to refine the lists of products displayed to users, see the *ATG Search Query Guide*.

Implementing Search Merchandising in a Multisite Environment

Environments that have ATG Merchandising and ATG Search installed can use a feature called Search Merchandising, which enables commerce sites to customize search results based on business initiatives and customer purchasing patterns. In ATG Merchandising, you create search configurations, which contain rules and settings that affect how search results display to a specified group of customers. Language, user segment, and/or site can be used as dimensions for determining which search configuration to use for a given user.

In a multisite environment, merchandisers can:

- Use a specified site as a search configuration dimension.
- Define search configurations that can be used across multiple sites.
- Assign percentage weights to sites to determine the relative importance of a site in determining the order of search results.

For information about administering search configurations so they are ready for use by business users, see the *ATG Merchandising Administration Guide*. For information about creating search configurations, see the *ATG Merchandising Guide for Business Users*.

Once you have created your search configurations, Search Testing lets you test and see previews of the impact that search configuration changes will have on the application before updates are actually deployed to staging or production.

In a multisite environment, merchandisers can:

- Select a site as a dimension when running a search test.
- Specify a site on which to simulate their search, or conduct a cross-site search test by searching across all registered sites in a specific search environment.

For information on configuring search tests, see the *ATG Merchandising Administration Guide*. For information about running search tests and interpreting the results, see the *ATG Merchandising Guide for Business Users*.

7 JSP Page Updates

This chapter describes development tasks that ensure the JSP pages are rendering content that is appropriate for the site context. It includes the following sections:

Styling Your JSP Pages

The JSP pages you develop for your application and the way you style them is driven by your application's requirements. CRS uses the same JSP pages for all of its sites along with a set of site-specific cascading style sheets (CSS) that style the templates differently for the different sites and locales. CRS uses a custom site configuration property, `cssFile`, along with JSP code to set the cascading style sheet used for each request. The remainder of this section describes how this mechanism works.

Setting the Base CSS Filename for an ATG Commerce Reference Store Site

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

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

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

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

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

Linking to the Correct CSS in ATG Commerce Reference Store

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

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

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

  <%-- Other page code exists here but has been removed for readability. --%>

  <%-- Load the site specific CSS --%>
```

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

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

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

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

<%-- Other page code exists here but has been removed for readability. --%>

</dsp:page>
```

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

Other CSS Files

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

Adding Controls That Change Site Context

To add controls that change the site context, the first thing you must do is retrieve a list of the `siteConfiguration` items to which you want to provide switching controls. This could be a list of all the sites in your application, or a list of the sites in a particular sharing group.

If you are rendering switching controls for all sites in your application, you can use the `/atg/multisite/SiteManager` component. With the `SiteManager`, you can retrieve `siteConfiguration` items for all existing sites, all enabled sites or all active sites. The following examples show how.

To retrieve `siteConfiguration` items for active sites:

```
<dsp:getvalueof var="activeSites"
                bean="/atg/multisite/SiteManager.activeSites"/>
```

To retrieve `siteConfiguration` items for enabled sites:

```
<dsp:getvalueof var="enabledSites"
                bean="/atg/multisite/SiteManager.enabledSites"/>
```

To retrieve `siteConfiguration` items for all sites:

```
<dsp:getvalueof var="allSites"
                bean="/atg/multisite/SiteManager.allSites"/>
```

If you are rendering switching controls for the sites in a sharing group, use the `/atg/dynamo/droplet/multisite/SharingSitesDroplet` instead. Depending on the input parameters you provide, this droplet returns all sites that are in a sharing group with the current site (or with a site other than the current one, if you provide a site ID). For example, you can return all sites that share a shopping cart with the current site. This droplet can be configured to exclude the current site from the list of returned `siteConfiguration` items, allowing you to render controls that do not provide a link to the site the customer is currently viewing. You can also specify that only active sites are returned.

After you have retrieved a list of `siteConfiguration` items, you can render links to them. CRS uses the `/atg/dynamo/droplet/multisite/SiteLinkDroplet` to build these links. This example shows how `store.war/navigation/gadgets/sites.jsp` renders the Store portion of the CRS site picker. It uses the shopping cart sharing group to determine which sites links should be rendered for.


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

  <%-- Get the site ID for the current site. The current site should not
  be rendered as a link in the Store site picker. --%>
```

```
<dsp:getvalueof var="currentSiteId" bean="Site.id"/>

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

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

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

        <ul>

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

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

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

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

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

```
                </c:otherwise>

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

Similar code renders the Country portion of the picker and can be found in `store.war/navigation/gadgets/regions.jsp`. The `regions.jsp` page uses the Related Regional Stores sharing group to identify the collection of sites that should be rendered next to the Country label. Note that because the country name requires translation, `regions.jsp` retrieves it from a resource file, not from the `site` object.

For more details on the `SiteManager` component, see Configuring Site Types. For details on the `SharingSitesDroplet` and `SiteLinkDroplet`, see the *ATG Page Developer's Guide*.

Adding Cross-site Links

Multisite functionality allows you to show links to products from other sites while viewing the current site. There are a number of locations where these links could be appropriate. For example, in CRS, cross-site product links appear in the following places:

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

In addition to product links, you may also want to create links to static pages from another site (for example, `index.jsp` pages). ATG Commerce includes several servlet beans that facilitate the creation of cross-site links in both cases:

- `atg.droplet.multisite.SiteIdForItemDroplet` obtains a site ID for a repository item. There are two component instances that are based on this class, `/atg/dynamo/droplet/multisite/SiteIdForItem` and `/atg/commerce/multisite/SiteIdForCatalogItem`. These components are

essentially the same except that `SiteIdForCatalogItem` sets an additional parameter, `shareableTypeId`, for convenience.

- `atg.droplet.multisite.SiteLinkDroplet` generates URLs that reference other sites.

The following sections describe how CRS uses these servlet beans to generate its cross-site links.

Creating a Cross-site Link to a Static Page

The code sample in this section is a modified version of CRS's `store.war/global/gadgets/crossSiteLinkGenerator.jsp` page. It takes a known site ID and creates a URL that follows this pattern:

`production-URL/path-to-static-page`

For example:

```
/crs/storeus/index.jsp
```

In this example, `/crs/storeus` is the *`production-URL`* and `/index.jsp` is the *`path-to-static-page`*. Here is the code sample:

```
<%-- Pass a site ID and path to SiteLinkDroplet to get a static site-specific
URL. --%>
<dsp:droplet name="/atg/dynamo/droplet/multisite/SiteLinkDroplet">
   <dsp:param name="siteId" value="homeSite"/>
   <dsp:param name="path" value="/index.jsp"/>
   <dsp:oparam name="output">

      <%-- Render the generated URL. --%>
      <dsp:getvaluof var="siteLinkUrl" param="url" />
      <dsp:a href="${siteLinkUrl}">Go to Home Store</dsp:a>

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

The code sample resolves the various portions of the URL as follows:

- `SiteLinkDroplet` uses the value of its `siteId` parameter to determine which site it is building a URL for, and then it retrieves the *`production-URL`* portion of the URL from the `productionURL` property for that site. In our example, 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 for the link.
- `SiteLinkDroplet` then appends the value of its `path` parameter, `/index.jsp`, to the `productionURL` to create the final URL, `/crs/storeus/index.jsp`.

Creating a Cross-site Link to a Dynamic Page

The next code sample is also a modified version of the `crossSiteLinkGenerator.jsp` page. It uses `SiteIdForCatalogItem` in conjunction with `SiteLinkDroplet` to generate a URL that follows this pattern:

`production-URL/product-template/product-ID`

For example:

```
/crs/storeus/browse/productDetailsSingleSku.jsp?productId=xprod200
```

The product template that gets used depends on the product you are creating a link for, hence, the dynamic nature of this example. In the URL above, `/crs/storeus` is the *`production-URL`*, `/browse/productDetailsSingleSku.jsp` is the *`product-template`* and `?productId=xprod200` is the *`product-ID`*. Here is the code sample:

```
<%-- 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 code sample resolves the various portions of the URL as follows:

- The `SiteIdForCatalogItem` droplet determines the best site ID to use for the product.
- The `SiteLinkDroplet` uses the `siteId` returned by `SiteIdForCatalogItem` to determine which site it is building a URL for, and then it retrieves the *`production-URL`* portion of the URL from the `productionURL` property for that site. In our example, 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 for the link.
- The *`product-template`* portion comes from the product's `product.template.url` property, which is passed to `SiteLinkDroplet` in its `path` parameter.
- `SiteLinkDroplet` generates a URL comprised of the production URL and product template. The JSP then appends the `?productId=`*`productID`* portion to the URL using this statement:

  ```
  <dsp:param name="productId" param="product.repositoryId"/>
  ```

Additional Notes:

- In some instances, you will want your application to render product links to other sites (for example, cross sells). In other instances, changing the site context is inappropriate so, while you may be displaying products from other sites, you should not render them as links to other sites (for example, order confirmation).
- For more details on the `SiteLinkDroplet`, `SiteIdForItem`, and `SiteIdForCatalogItem` servlet beans, see the *ATG Page Developer's Guide*.
- `crossSiteLinkGenerator.jsp` is a CRS gadget that simplifies adding cross-site functionality to many locations and includes logic for adding cross-site links in a variety of circumstances.
- The display name in the BCC for the site configuration's `productionURL` property is Site Base URL.

Rendering a Site Indicator

When displaying product information, you may want to render an indicator that provides a site name and/or logo that indicates which site a product belongs to. You could also choose to make this site indicator a cross-site link to the product. CRS uses two types of site indicators, one that is logo-based and another that is name-based:

Logo-based site indicator

Name-based site indicator

CRS uses the site configuration's `siteIcon` property to render logo-based site indicators. This property is part of the standard site configuration properties included with the ATG platform and its value is a string that is relative to the `storedocroot.war` Web application's context root and points to an image resource. For example, the `siteIcon` property for ATG Home is `/crsdocroot/images/storefront/homeStore_logo_small.png`. This image includes both the logo and the name of the store, as shown in the first example above. Note that logo-based site indicators that are rendered in e-mails must use a fully qualified URL, including server name and port, to properly reference the image resource file. For name-based site indicators, CRS uses the site configuration's `name` property, which is also a standard site configuration property included with the ATG platform.

An additional property that can display a site indicator is `favicon`. The `favicon` is a 16x16 pixel icon.

CRS's `store.war/global/gadgets/siteIndicator.jsp` page renders a variety of site indicators, depending on the parameters you provide it. The following table describes the site indicators that `siteIndicator.jsp` can render and the parameters you need to provide for each type:

Site Indicator Type	Parameters
Icon/no link	`mode = icon`
Icon/link	`mode = icon` `asLink = true`

Icon/link/absolute URL	`mode = icon`
	`asLink = true`
	`absoluteResourcePath = true`

Name only/no link	`mode = name`
Name only/link	`mode = name` `asLink = true`

Note: While `siteIndicator.jsp` is capable of rendering site indicators with cross-site links, CRS renders its site indicators without them.

Filtering Items for the Current Site

You can use custom filter servlet beans to filter repository items so that you only display items that are appropriate for a customer's current site context. For example, ATG Commerce implements a custom filter, `/atg/registry/CollectionFilters/GiftlistSiteFilter`, that filters gift lists so that only lists that are associated with a specified site or sharing group are displayed. `GiftlistSiteFilter` is of

class `atg.commerce.gifts.GiftlistSiteFilter`, which is an extension of the generic collection filtering class `atg.service.collections.filter.CachedCollectionFilter`. You can use `GiftlistSiteFilter` as a model to create similar filters for other repository item types. See the *ATG Commerce Programming Guide* for detailed information on this filter.

Note: The catalog repository has an internal mechanism that allows the repository to return only those items that are appropriate for the current site context, obviating the need for custom filtering of catalog repository items. This includes catalog, category, product, and SKU item types. For more information on this mechanism, see Site Specific Repository Data (Site Membership).

Implementing a Multisite Search Control

A multisite search control allows customers to choose which sites to search on. For example, they could choose to search only the current site or to search all sites that share a shopping cart with the current site. The way you implement a multisite search control depends on whether you are using ATG Search or the native search functionality that comes with the ATG platform.

Note: This section assumes ATG Search has been properly configured for a multisite environment. See Configuring ATG Search in a Multisite Environment for information on doing so.

ATG Search Multisite Control

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

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

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

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

  <dsp:oparam name="output">
    <%-- Sort sites --%>
    <dsp:getvalueof var="sites" param="sites"/>
```

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

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

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

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

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

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

Basic Search Multisite Control

ATG's basic search form handlers are based on the class `atg.repository.servlet.SearchFormHandler`. This class accepts arguments that allow you to filter search results to include only those repository items that are affiliated with specified sites. When using a search form handler based on this class, or one of its subclasses such as `atg.commerce.catalog.custom.CatalogSearchFormhandler`, you can set one of the following search form handler properties to filter search results:

- `siteIds` contains the list of site IDs. Only items that belong to the specified sites are returned. If `siteIds` is null, `siteScope` is used instead.

- `siteScope` specifies the site scope to use when searching for items. Possible values include:
  - `current` returns matching repository items from the current site only. This is the default.
  - `all` returns all matching repository items and does not filter based on site context.
  - `any` returns matching repository items that belong to any site.
  - `none` returns matching repository items that do not have any site affiliations.
  - *`shareable-type-ID`* returns matching repository items that belong to any sites that are in a sharing group with the current site, as defined by the `ShareableType` component ID. For example, you can return items that belong to sites that share a shopping cart with the current site.

In addition to `siteIds` and `siteScope`, you can also set the parameter `activeSitesOnly`. When this parameter is set to `true`, search results are further limited to active sites only. When it is set to `false`, results from both active and disabled sites are returned.

`SearchFormHandler` includes a method, `generateResultSet()`, that builds and executes the search query. If necessary, this method will take the `siteIds` and `siteScope` parameters into account, as shown in this code sample:

```
if (q != null) {
   // Generate site context filters using the generateSiteFilter() method.
   // This method uses the siteIds and siteScope parameters to create the filters.
   ContextFilteringRules siteFilter = generateSiteFilter();

   // Add site context filters to the query
      if (siteFilter != null) {
         q = view.getQueryBuilder().addContextMembershipFilters(q, siteFilter);
   }
   // Execute the query
   RepositoryItem[] items = view.executeQuery(q, 0, endIndex, null);
```

Subclasses that override `generateResultSet()` and need to be site-aware should implement similar logic to create and execute queries that incorporate site context filters. CRS implements such a subclass in `atg.projects.store.catalog.StoreSearchFormHandler`. The `/atg/commerce/catalog/ProductSearch.properties` component uses this class to search the Catalog Repository for product items. The remainder of this section shows several examples of how to use the `ProductSearch` component in a JSP page. These examples are simplified versions of the `store.war/search/gadgets/simpleSearch.jsp` page found in CRS.

Example 1: This code sample shows a product search form that uses the `siteIds` property to filter search results. In this example, `siteIds` is set to include the current site and sites that share a shopping cart with the current site.

```
<dsp:importbean bean="/atg/dynamo/droplet/multisite/SharingSitesDroplet" />
<dsp:importbean bean="/atg/dynamo/droplet/ForEach" />
<dsp:importbean bean="/atg/multisite/Site"/>
<dsp:importbean bean="/atg/commerce/catalog/ProductSearch"/>
<dsp:getvalueof id="contextroot" idtype="java.lang.String"
  bean="/OriginatingRequest.contextPath"/>

<dsp:form action="${contextroot}/search/searchResults.jsp" method="post"
  id="simpleSearch" formid="simplesearchform">

    <%-- Search input control --%>
    <dsp:input bean="ProductSearch.searchInput" type="text" value="" />

    <%-- Get the list of sites that share a shopping cart with the
    current site. --%>
    <dsp:droplet name="SharingSitesDroplet">
      <dsp:param name="shareableTypeId" value="atg.ShoppingCart"/>
      <dsp:param name="excludeInputSite" value="true"/>

      <%-- Loop through the sites that share a shopping cart and render labels
      and checkboxes for them. --%>
      <dsp:oparam name="output">
        <dsp:droplet name="ForEach">
          <dsp:param name="array" param="sites"/>
          <dsp:setvalue param="site" paramvalue="element"/>

            <%-- Display a checkbox and name for the current site first. --%>
            <dsp:oparam name="outputStart">
              <dsp:input bean="ProductSearch.siteIds" type="checkbox"
                         beanvalue="Site.id" checked="true" id="currentStore"/>
              <label for="currentStore">
                <dsp:valueof bean="Site.name"/>
              </label>
            </dsp:oparam>

            <%-- Display the other sites that share shopping cart with the current
            site. --%>
            <dsp:oparam name="output">
                <dsp:input bean="ProductSearch.siteIds" type="checkbox"
                           paramvalue="site.id" id="otherStore" checked="false"/>
                <label for="otherStore">
                  <dsp:valueof param="site.name"/>
                </label>
            </dsp:oparam>
        </dsp:droplet>
      </dsp:oparam>

      <%-- If there are no shared sites, include the current site only and
      don't display checkboxes. --%>
```

```
      <dsp:oparam name="empty">
        <dsp:input bean="ProductSearch.siteIds" type="hidden"
                   beanvalue="Site.id"/>
      </dsp:oparam>
    </dsp:droplet>

    <%-- Display the search form's submit button. --%>
    <dsp:input bean="ProductSearch.search" type="submit"  value="Search"/>

</dsp:form>
```

Example 2: In this example, the product search form returns results from all active sites.

```
<dsp:importbean bean="/atg/dynamo/droplet/ForEach" />
<dsp:importbean bean="/atg/multisite/Site"/>
<dsp:importbean bean="/atg/commerce/catalog/ProductSearch"/>
<dsp:getvalueof id="contextroot" idtype="java.lang.String"
bean="/OriginatingRequest.contextPath"/>

<dsp:form action="${contextroot}/search/searchResults.jsp" method="post"
id="simpleSearch" formid="simplesearchform">

    <%-- Search input control --%>
    <dsp:input bean="ProductSearch.searchInput" type="text" value="" />

    <%-- Get the list of all active sites --%>
    <dsp:getvalueof var="sites" bean="/atg/multisite/SiteManager.activeSites"/>

        <%-- Loop through the all active sites --%>
        <dsp:droplet name="ForEach">
          <dsp:param name="array" value="${sites}"/>
          <dsp:setvalue param="site" paramvalue="element"/>

            <%-- Display a checkbox and name for the current site first. --%>
            <dsp:oparam name="outputStart">
              <dsp:input bean="ProductSearch.siteIds" type="checkbox"
                         beanvalue="Site.id" checked="true" id="currentStore"/>
              <label for="currentStore">
                <dsp:valueof bean="Site.name"/>
              </label>
            </dsp:oparam>

            <%-- Display the other sites that share a shopping cart with the
            current site. --%>
            <dsp:oparam name="output">
              <dsp:getvalueof var="siteId" param="site.id"/>
              <dsp:getvalueof var="currentSiteId" bean="Site.id"/>
              <%-- Check that the site is not the current site. If so, display
```

```
            its name and checkbox so it can be included in the search. --%>
            <c:if test="${siteId != currentSiteId}">
              <dsp:input bean="ProductSearch.siteIds" type="checkbox"
                         paramvalue="site.id" id="otherStore" checked="false"/>
              <label for="otherStore">
                <dsp:valueof param="site.name"/>
              </label>
            </c:if>
          </dsp:oparam>
        </dsp:droplet>

    <%-- Display the search form's submit button --%>
    <dsp:input bean="ProductSearch.search" type="submit"  value="Search"/>

</dsp:form>
```

Example 3: This code sample shows a product search form that uses the `siteScope` property to filter search results. In this example, `siteScope` is set to a shareable type ID so that only products from sites that share a shopping cart with the current site are returned.

```
<dsp:importbean bean="/atg/commerce/catalog/ProductSearch"/>
  <dsp:getvalueof id="contextroot" idtype="java.lang.String"
                  bean="/OriginatingRequest.contextPath"/>

  <dsp:form action="${contextroot}/search/searchResults.jsp" method="post"
            id="simpleSearch" formid="simplesearchform">

    <%-- Search input control --%>
    <dsp:input bean="ProductSearch.searchInput" type="text" value="" />

    <%-- Specify the shopping cart shareable type ID in the siteScope
    property. --%>
    <dsp:input bean="ProductSearch.siteScope" type="hidden"
               value="atg.ShoppingCart" />

    <%-- Display the search form's submit button. --%>
    <dsp:input bean="ProductSearch.search" type="submit"  value="Search"/>

  </dsp:form>
```

Creating E-mail Templates in a Multisite Environment

E-mail templates, in general, are rendered the same as any JSP page and can use the current site context to customize their content so that it is site-specific. When creating e-mail templates, keep in mind the following two caveats:

- All URLs in an e-mail template must be fully qualified. Using the `SiteLinkDroplet` servlet bean to create any URLs ensures that they will be fully and properly formed, so you should use it for all links in e-mail templates.
- E-mail templates cannot link to cascading style sheets and, therefore, should instead use inline styles.

CRS includes a number of e-mail templates that you can use as examples in its `store.war/emailtemplates` directory.

Changing Site Context within a Single Request

If you have part of a JSP page that needs to be rendered in a site context other than the current site, you can use the `/atg/dynamo/droplet/multisite/SiteContextDroplet`. The output that is wrapped in the `SiteContextDroplet` component's `output` parameter is rendered in the context of the site specified in its `siteId` parameter. The following is an example of `SiteContextDroplet` usage:

```
<dsp:droplet name="/atg/dynamo/droplet/multisite/SiteContextDroplet">
   <dsp:param name="siteId" value="storeSiteDe"/>

      <dsp:oparam name="output">

         <%-- Content that should be rendered in the specified site context goes
         here. --%>

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

Part III: Site Administration Tasks

The following section describes working with Site Administration. To create multiple sites, a Site Developer works in tandem with a Site Administrator to develop the site components. The following sections are divided into Site Administration Programming tasks and Site Administrator tasks; however, the information provided in either section is pertinent to both.

The following sections describe Site Administration Programming tasks:

Installing Site Administration

Defining Site Categories

Configuring URLs

Configuring Site Types

Sharing Data

The following section describes Site Administration administrative tasks:

Using Site Administration

8 Installing Site Administration

Site Administration is included in the installation of the ATG platform when you run the ATG Configuration Installation Manager (CIM) script. When running CIM you can select Site Administration from the Product Selection menu.

This section contains the following:

Installing Site Administration using CIM

The following is an example of the CIM installation that installs the ATG platform, ATG Content Administration and ATG B2B Commerce:

```
=======CIM MAIN MENU==============
enter [h]elp, [q]uit to exit
Choose the task you want to perform:
  *[P]  Product Selection
   [A]  Select Application Server
   [C]  Custom CIM Plugin Launcher
   [B]  Batch File Save
> P
-------PRODUCT SELECTION-----------
enter [h]elp, [m]ain menu, [q]uit to exit

Select product you wish to install by entering the corresponding item number.
    (Searching for products... done.)

Choose one of the following options: (* = Currently selected )

[1]  ATG Platform and Content Administration :
     Includes, optionally, ACI and Preview
[2]  ATG Site Administration :
     Includes ATG Platform and Content Administration.
[3]  ATG Search :
     Includes ATG Platform, Content Administration and ATG Search Admin
```

```
     and optional ACI and Preview
[4]  ATG B2C Commerce :
     Includes ATG Platform, Content Administration and, optionally, ACI,
     Preview, ATG Search, and Merchandising
[5]  ATG B2B Commerce :
     Includes ATG Platform, Content Administration and, optionally,
      ACI, Preview, and Merchandising
[6]  ATG Commerce Reference Store :
     Includes ATG Platform, Content Administration, Site Administration,
     B2C Commerce, Merchandising and, optionally, ACI and Preview
[D]  Done

Select one or more > 2,5
```

Requirements

The following are required to run Site Administration.

Site Administration Access Rights

Site Administrators must have the `siteAdminUser` or the BCC Administrator role, which allows access to Site Administration. The `Siteadmin-role-data.xml` contains the role and the access right for Site Administration. For information on configuring BCC access, refer to the *ATG Business Control Center Administration and Development Guide*.

Additionally, users must be configured with the BCC Member User role and the `epubUser` role. Although not all users who use ATG Merchandising will be Site Administrators, all Site Administrators must have access to ATG Merchandising assets. For information on configuring ATG Merchandising rights, refer to the *ATG Merchandising Administration Guide*.

9 Defining Site Categories

When creating multiple sites, you create properties and environments that are used by each site. Site category, using view mapping, define these properties.

This section contains the following:

Site Categories Overview

When you create a site in Site Administration, it is based upon a *site category* that represents specific site elements, providing metadata for your site. The properties and values that are used by site categories are defined in view mappings that you create for specific or a group of sites.

Site categories, which are not versioned repository assets and are not deployed to your production environment, must be registered with Site Administration before they become available for assignment.

Registering a site category is comprised of setting up the site category's view mapping information, then using the site category administration interface in the BCC to create site configurations using the `siteTemplate` item descriptor. For information on registering sites, refer to Identifying Valid Site Types.

Site Administrators use the Manage Site Category menu to access, create, edit or delete site categories. Site categories are created by site developers who set up ATG View Mapping customizations as required.

Site category components are accessed using the Manage Site Categories link within the Site Administration BCC.

Site Category Elements

The following elements are used to make site categories:

- Site Category ID – The system generates a unique and non-modifiable identifier whenever a site category is created.

- Site Category Name – The site category requires that you provide a unique name.
- Site Category Description – When creating a site category, provide a comprehensive description of the category that allows other administrators to understand what components are contained within the configuration.
- Site Item Mapping Name Information - This identifies the ATG View Mapping components associated with the site category. View Mapping determines which configurable properties are included in any site category.

  Note: The Site Item Mapping Name is used in the creation of the site category, not the Site Item Mapping ID.

When a Site Administrator creates a new site category, they provide the site category information using the site category panels. Site Administration has a default site category that can be used as is, or used as an example to create new categories. The following is an example of the default Site Administration site category configuration:

```
<gsa-template>
  <import-items>
<!-- Site Templates -->

<add-item item-descriptor="siteTemplate" id="sa_default_category">
  <set-property name="name"><![CDATA[Default Category]]></set-property>
  <set-property name="description"><![CDATA[This is the default ATG site
      configuration category that will include all available
      properties.]]></set-property>
  <set-property name="itemMappingId"><![CDATA[SiteCategory:default]]>
      </set-property>
</add-item>

</import-items>
</gsa-template>
```

For additional information, refer to the Implementing Site Categories section.

Site Properties Configuration

The properties for each site are based on the site category configuration. There are a number of default site properties that are available for all sites. Additional default site properties are added when you install other applications or modules. For example, if you have ATG Commerce installed, additional properties are installed for Commerce when running CIM. Refer to your ATG documentation for additional information on site properties installed and used by your application.

Site Property Categories

Site properties can be grouped into logical categories that are presented in the Site Definition screen.

The following site properties are relevant only if the property is included in the site category:

- Required - A property value must be present, for example, the value can not be null
- Automatic - Property value must be present, but the value is automatically populated by the system

Site Administration provides two default categories: Basic and Site Life Cycle.

Property Editors

Each configurable property is edited using a property editor. The value type set for a configurable property determines the property editor that is used. Editor properties are defined in the `viewmapping.xml` file. Editors are provided for the following value types:

- Strings
- Integers
- Float
- Date
- Time Stamp
- Enumerated
- Enumerated String
- Boolean
- Repository Item
- Collection

Note: Ensure that any copies you make of the view mapping file include the property editors. For additional information on editors, refer to *ATG Merchandising Guide for Business Users*.

Site Configuration Properties

The following are the default site configuration properties as defined in the `siteRepositoryTemplateResources.properties` file.

Note: Properties are dependant on the modules that are installed on your system. The following default site configuration properties are installed during the ATG platform installation:

Property	Req	Auto	Description
Site Name	Yes	No	This is the name assigned to the site, not the URL or homepage name.
Site Description	No	No	Required for ATG ACI.
Site ID	Yes	Yes	System generated unique IDs that can be set to a user-defined ID.
Site Category	Yes	No	Identifies the site category on which the site is based.

Property	Req	Auto	Description
Enabled Status	Yes	No	A site is `Enabled` or `Disabled`. The system references this property to determine whether to display the site. This property is set to `Disabled` by default.
Site Type	Yes	No	The type of site as defined by the site types registry in `/atg/multisite/SiteManager.siteTypes`
Site Icon	No	No	Location of the media asset in the file system to be used as the site icon.
Favicon	No	No	Location of the media asset in the file system to be used as the favorite inline site icon. Favicons are 16x16, 32x32 or 64x64 pixel square icons associated with a particular website.
Site Down URL	No	No	A URL where the incoming request is redirected when the Enabled Status has been set to Disabled.
Opening Date	No	No	The system references this property to determine whether to display the site to the end user but also accepts a null value. By default, the site is considered `Open` on this date at 12:00 a.m.
Pre-Opening URL	No	No	The system references this property to display an alternative public URL prior to `Opening Date`.
Closing Date	No	No	The system references this property to determine whether to display the site. The system will accept a `null` value, in which case, the system will refer to the `Enabled Status` property. By default, site is considered `Closed` as of 12:00a.m.
Post-Closing URL	No	No	The system may reference this property to determine the alternative public URL after `Closing Date`.
Site Base URL	No	No	The URL that will be used to resolve a browser request with a Web application. This may or may not be the public URL for the site.
Alternative URL(s)	No	No	More than one URL may map to the same Site Base URL. The Site Base URL itself may never be exposed to the public. The value is set to `null` by default.
Context Root	No	No	If the Site Base or Alternative URL is a virtual context root (or a context root that does not map to a physical Web application) this property stores the value for the actual context root that maps to a physical Web application.

Property	Req	Auto	Description
Site Priority	No	No	This property determines which site or site related item (such as catalog or pricelist) to retrieve in the event of a conflict. The value is set to `null` by default. **Note:** Only positive integer values can be assigned to this property.
Created By	Yes	Yes	This property is automatically populated with the name of the internal user who created the site.
Created Date	Yes	Yes	This system-generated property stores the date when the site was created.
Last Modified By	Yes	Yes	This system-generated property identifies the internal user who last modified the site.
Last Modified Date	Yes	Yes	This system-generated property stores the date when the site was last amended whenever an amendment is saved to an existing site.

New properties can be added to the site categories using the ATG View Mapping system.

The Site Repository `SiteRepositoryTemplateResource.properties` resource bundle contains the default properties used by Site Administration:

```
# item descriptors
itemDescriptorSiteConfiguration=Site
itemDescriptorSiteTemplate=Site Category
itemDescriptorSiteGroup=Site Group

itemDescriptorSiteConfiguration.plural=Sites
itemDescriptorSiteTemplate.plural=Site Categories
itemDescriptorSiteGroup.plural=Site Groups

# categories
categoryLife=Site Life-cycle
categoryBasics=Basics

# basic properties
name=Site Name
description=Site Description
siteTemplate=Site Category
siteIcon=Site Icon
favicon=Favicon
siteTypes=Site Types
sitePriority=Site Priority
templateName=Category Name
```

```
contextRoot=Context Root

siteGroupDisplayName=Site Group Name
sites=Sites in Group
shareableTypes=Shared Data

# life-cycle properties
enabled=Enabled Status
productionURL=Site Base URL
activityDates=Active Dates
modificationTime=Last Modified Date
creationTime=Created Date
author=Created By
lastModifiedBy=Last Modified By
openDate=Opening Date
closingDate=Closing Date
preOpeningURL=Pre-Opening URL
postClosingURL=Post Closing URL
closedURL=Site Disabled URL
siteDownURL=Site Disabled URL
additionalProductionURLs=Alternative URL(s)

# other
active=Active
required=Required
editable=Editable

# template item resources
template.name=Category Name
template.description=Category Description
template.itemMappingID=Item Mapping ID

#item descriptor icons
iconSwf.siteConfiguration.small=resource:ATGResources/
  assetIcon.site_iconSmall
iconSwf.siteConfiguration.large=resource:ATGResources/
  assetIcon.site_iconLarge
iconSwf.siteGroup.small=resource:ATGResources/
  assetIcon.siteGroup_iconSmall
iconSwf.siteGroup.large=resource:ATGResources/
  assetIcon.siteGroup_iconLarge
iconSwf.siteTemplate.small=resource:ATGResources/
  assetIcon.siteTemplate_iconSmall
iconSwf.siteTemplate.large=resource:ATGResources/
  assetIcon.siteTemplate_iconLarge
```

Site Repository Definition

All registered sites have a site ID and are recorded in the `siteRepository` definition. The Site Repository, which runs on all server instances, stores site configurations, site categories and site groups that are created with Site Administration. The Site Repository stores the following item descriptors:

- `siteConfiguration` – An item descriptor that contains the site configuration properties. Versioned `siteConfigurations` are deployed from the publishing server to the production server
- `siteGroup` – An item descriptor that contains sharing group properties. Versioned `siteGroups` are deployed from the publishing server to the production server
- `siteTemplate` – An item descriptor for a site category. Site categories are non-versioned instances that are stored in the `siteRepository` definition on the publishing server. Site categories do not get deployed.

The repository is also used by other applications to access site and property information. For example, `siteRepository` is used by `siteContext` to look up sites and identify site properties. The `siteManager` class provides additional methods for the `siteRepository`.

For additional information on `siteManager` and `siteContext`, refer to the *ATG Programming Guide*.

The following is an excerpted example of the item descriptors within the `siteRepository.xml` file:

```
<item-descriptor name="siteConfiguration" default="true" last-modified-
      property="modificationTime" display-property="name" display-name-
      resource="itemDescriptorSiteConfiguration" item-cache-size="1000"
      query-  cache-size="1000">
    <attribute name="plural-display-name-resource"
        value="itemDescriptorSiteConfiguration.plural"/>
    <attribute name="resourceBundle"
        value="atg.multisite.SiteRepositoryTemplateResources"/>
    <attribute name="categoryBasicsPriority" value="110"/>
    <attribute name="categoryLifePriority" value="120"/>
    <attribute name="updateLastModified" value="true"/>
    <attribute name="iconSwf" value="iconSwf.siteConfiguration.small"/>
    <attribute name="largeIconSwf"
        value="iconSwf.siteConfiguration.large"/>
    <table name="site_configuration" type="primary" id-column-names="id">
      <property name="id" column-names="id" data-type="string" />
      <property name="name" column-names="name" data-type="string"
          category-resource="categoryBasics" display-name-resource="name"
          required="true">
        <attribute name="propertySortPriority" value="0"/>
      </property>



<item-descriptor name="siteGroup" display-property="displayName"
    display-name-resource="itemDescriptorSiteGroup" item-cache-size="1000"
```

```
    query-cache-size="1000">
  <attribute name="plural-display-name-resource"
      value="itemDescriptorSiteGroup.plural"/>
  <attribute name="resourceBundle"
      value="atg.multisite.SiteRepositoryTemplateResources"/>
  <attribute name="iconSwf" value="iconSwf.siteGroup.small"/>
  <attribute name="largeIconSwf" value="iconSwf.siteGroup.large"/>
    <table name="site_group" type="primary" id-column-names="id">
      <attribute name="resourceBundle"
          value="atg.multisite.SiteRepositoryTemplateResources"/>
        <property name="id" column-names="id" data-type="string"/>
        <property name="displayName" column-names="display_name"
            data-type="string" category-resource="categoryBasics"
            required="true" display-name-resource="siteGroupDisplayName"/>
    </table>
    <table name="site_group_sites" type="multi"
        id-column-name="site_group_id">
      <property name="sites" column-names="site_id" data-type="set"
          component-item-type="siteConfiguration"
          category-resource="categoryBasics"
          display-name-resource="sites">
        <attribute name="removeNullValues" value="true"/>
      </property>
    </table>
    <table name="site_group_shareable_types" type="multi"
        id-column-name="site_group_id">
      <property name="shareableTypes" column-names="shareable_types"
          data-type="set" component-data-type="enumerated string"
          category-resource="categoryBasics"
          display-name-resource="shareableTypes">
        <attribute name="stringEnumProvider"
            value="/atg/multisite/ShareableTypeStringEnumProvider"/>
        <attribute name="propertySortPriority" value="100"/>
      </property>
    </table>
</item-descriptor>



<item-descriptor name="siteTemplate" versionable="false"
    display-property="name"
    display-name-resource="itemDescriptorSiteTemplate"
    item-cache-size="1000" query-cache-size="1000">
  <attribute name="plural-display-name-resource"
      value="itemDescriptorSiteTemplate.plural"/>
  <attribute name="resourceBundle"
      value="atg.multisite.SiteRepositoryTemplateResources"/>
  <attribute name="deployable" value="false"/>
  <attribute name="iconSwf" value="iconSwf.siteTemplate.small"/>
  <attribute name="largeIconSwf" value="iconSwf.siteTemplate.large"/>
```

```
    <table name="site_template" type="primary" id-column-names="id">
      <property name="id" column-names="id" data-type="string"/>
      <property name="name" column-names="name" data-type="string"
          display-name-resource="template.name" required="true">
        <attribute name="unique" value="true"/>
        <attribute name="propertySortPriority" value="0"/>
      </property>

      <property name="description" column-names="description"
          data-type="string" display-name-resource="template.description"
          required="true">
        <attribute name="propertySortPriority" value="20"/>
      </property>

      <property name="itemMappingId" column-names="item_mapping_id"
          data-type="string"
          display-name-resource="template.itemMappingID"
          required="true">
        <attribute name="propertySortPriority" value="40"/>
      </property>
    </table>
</item-descriptor>
```

For a list of tables used in the Site Repository refer to Tables Used in the Site Repository.

Configuring Site Categories

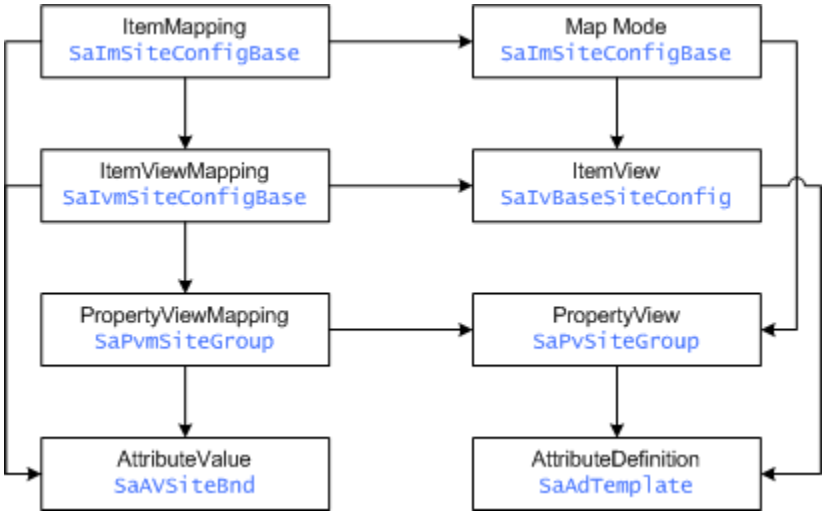
Site categories use the ATG View Mapping system to define UI details for repository items. View mappings consist of an item mapping that corresponds to the repository item to display in the UI and multiple item view mappings and item views for each tab on the UI.

View mappings use property view mappings and property views that can be defined on a per property basis. The item view mapping holds the relevant property view mappings. Default editors are defined based on the data type of the property.

View mappings use attributes such as `required` or `read only`, to define relevant UI details. For information on creating `required` or `read only` properties, refer to the Creating Required or Read Only Properties section. Both item and property views have attribute definitions defined. The item and property view mappings, as well as the item mapping, have attribute values defined. The attribute values and the attribute definitions are combined into one attribute map. View mapping finds item mappings that use the four following attributes in the Item Mapping repository: `name`, `itemName`, `itemPath` and `mode` properties.

Multiple view mappings can refer to the same repository item, allowing you to configure view mappings that identify different attributes or values for each property.

The following diagram displays the relationship between the elements in the View Mapping system. This diagram also displays the ATG naming convention for these elements:



The view mappings repository holds an `ItemMapping` repository item for each site category. The site category and its item mapping association are configured using the item mapping `name` value.

The item mapping name used for the category must be unique and should be formatted as `SiteCategory:`*`category name`*. The view mapping item name and item path are set to the Site Configuration repository item values.

Default Site Category Configuration

Default site category configuration information is stored in the `SiteAdmin/Versioned /install/data/viewmapping.xml` file. The following item mapping is used when the `SiteRepository:siteTemplate` item view mapping is set to `SiteCategory:default`:

```
<add-item item-descriptor="itemMapping" id="SaImSiteDefaultCategory">
  <set-property name="name"><![CDATA[SiteCategory:default]]>
      </set-property>
  <set-property name="description"><![CDATA[AssetManager default site
      itemMapping]]></set-property>
  <set-property name="mode"><![CDATA[AmMmDef]]></set-property>
  <set-property name="itemPath"><![CDATA[/atg/multisite/SiteRepository]]>
      </set-property>
  <set-property name="itemName"><![CDATA[siteConfiguration]]>
      </set-property>
  <set-property name="formHandler"><![CDATA[SaFhSite]]></set-property>
  <set-property name="viewMappings"><![CDATA[SaIvmSiteDefaultCategory]]>
      </set-property>
  <set-property name="attributes"><![CDATA[resourceBundle=SaAvSiteBnd,
```

```
      showCreationId=SaAvCatCreateId]]></set-property>
</add-item>

<add-item item-descriptor="itemViewMapping" id="SaIvmSiteDefaultCategory">
  <set-property name="name"><![CDATA[AssetManager default site
      itemViewMapping]]></set-property>
  <set-property name="displayName"><![CDATA[SaIvmSiteDefaultCategory.
      displayName]]></set-property>
  <set-property name="view"><![CDATA[SaIvBaseSiteConfig]]></set-property>
  <set-property name="propertyMappings"><![CDATA[openDate=SaPvmTimeStamp,
      closingDate=SaPvmTimeStamp,
      name=SaPvmSiteName,
      contextRoot=SaPvmSiteURL,
      productionURL=SaPvmSiteURL,
      siteDownURL=SaPvmSiteURL,
      preOpeningURL=SaPvmSiteURL,
      postClosingURL=SaPvmSiteURL,
      additionalProductionURLs=SaPvmAlternativeURL,
      sitePriority=SaPvmSitePriority,
      siteTypes=SaPvmSiteTypes]]>
      </set-property>
  <set-property name="attributeValues">
      <![CDATA[resourceBundle=SaAvSiteBnd]]></set-property>
</add-item>
```

The default site category can be modified to limit the number of configurable properties available. However, keep in mind that:

- If the default site category is deleted from the system, a specific site category must be assigned for each site.
- Sites based on a deleted default site category cannot be managed in Site Administration and must have a new site category assigned to them. This development task cannot be performed in the Site Administration UI.

Important Warning

When creating a site category item mapping, do not modify the default item mapping. To ensure that the item mapping properties are available, make a copy of the item mappings and work with the copy.

Site Administration defines custom `propertyView` and `propertyViewMapping` items that provide enhanced functionality to the default property editors. In addition, other ATG modules define their own specific `propertyView` and `propertyViewMapping` items that create enhanced functionality for their site properties.

Ensure that all default `PropertyViewMapping` items are available across all site category `ItemMappings`. If you create your own `PropertyViewMapping`, you must ensure that it includes everything that is contained in the default `PropertyViewMapping`. Using the default `PropertyViewMapping` also allows you to update only one `PropertyViewMaping` file, which updates all site categories using the file.

Also ensure that your site category `itemViewMapping.propertyMappings` property includes the mappings to custom `propertyViewMappings`. You can identify the mappings that are in your default site category by querying the `viewmapping` repository, located at `/atg/web/viewmapping/ViewMappingRepository`, with:

```
<print-item item-descriptor="itemViewMapping"
  id="SaIvmSiteDefaultCategory" />
```

In addition, whenever you add your own custom `propertyView` and `propertyViewMapping` items, you should add them to the `defaultTemplate` and any other site categories that use the related property.

For in-depth information on creating and using view mapping, refer to the *Customizing Asset Display* chapter of the *ATG Content Administration Programming Guide* and the *Controlling Details and Pane Properties* section of the *ATG Business Control Center Administration and Development Guide*.

Registering the Search Content Editor

The default site category `itemMapping` element is configured with the `ID="SaImSiteDefaultSiteCategory"` and `Name="SiteCategoryDefault"` settings. When you add site configuration, you can create an additional `ItemMapping`. However, you must enable the Custom Search Content Editor by registering it within your `viewmappings.xml` file and importing it using the startSQLRepository utility. For information on importing repository data, refer to the *ATG Repository Guide*.

For example:

```
<update-item item-descriptor="itemViewMapping"
  id="SaIvmSiteNewSiteCategory">
  <set-property name="propertyMappings"
    add="true"><![CDATA[searchContents=SaPvmSiteCntnt]]></set-property>
</update-item>
```

When working with content sets, `includeContentSets` and `excludeContentSets` attributes must be defined in a `propertyView` that uses the Search Content Editor. To create several site categories with different content sets that use the `include` or `exclude` attributes, you must define separate `propertyView` and `propertyViewMappings` for each site category. For information on content sets, refer to the *ATG Search Installation and Configuration Guide.*

Creating Site Categories

You create site categories by performing the following steps:

1. Create a copy of the `viewmapping.xml` file.
2. Create a item mapping based on the site configuration repository item with a unique name. For example:

   ```
   <add-item item-descriptor="itemMapping" id="SaImNewSiteCategory">
     <set-property name="name"><![CDATA[SiteCategory:new category]]>
         </set-property>
   ```

```
  <set-property name="description"><![CDATA[New Site Category
      itemMapping]]></set-property>
  <set-property name="mode"><![CDATA[AmMmDef]]></set-property>
  <set-property name="itemPath"><![CDATA[/atg/multisite/
      SiteRepository]]></set-property>
  <set-property name="itemName"><![CDATA[siteConfiguration]]>
      </set-property>
  <set-property name="formHandler"><![CDATA[SaFhSite]]>
      </set-property>
  <set-property name="viewMappings">
      <![CDATA[SaIvmNewSiteCategory]]></set-property>
  <set-property name="attributes"><![CDATA
      [resourceBundle=SaAvSiteBnd,showCreationId=SaAvCatCreateId]]>
      </set-property>
</add-item>
```

3. Create an item view mapping and item view for the main (general) tab excluding any specific categories or properties you want to hide and including any specific properties you want to see. For example:

```
<add-item item-descriptor="itemViewMapping"
    id="SaIvmNewSiteCategory">
  <set-property name="name"><![CDATA[New Site Category
      itemViewMapping]]></set-property>
  <set-property name="displayName">
      <![CDATA[SaIvmSiteDefaultCategory.displayName]]>
      </set-property>
  <set-property name="view"><![CDATA[SaIvBaseSiteConfig]]>
    </set-property>
  <set-property name="propertyMappings">
      <![CDATA[openDate=SaPvmTimeStamp,
      closingDate=SaPvmTimeStamp,
      name=SaPvmSiteName,
      contextRoot=SaPvmSiteURL,
      productionURL=SaPvmSiteURL,
      siteDownURL=SaPvmSiteURL,
      preOpeningURL=SaPvmSiteURL,
      postClosingURL=SaPvmSiteURL,
      additionalProductionURLs=SaPvmAlternativeURL,
      sitePriority=SaPvmSitePriority,
      siteTypes=SaPvmSiteTypes,
      enabled=SaPvmEnableStatus]]></set-property>
  <set-property name="attributeValues">
      <![CDATA[resourceBundle=SaAvSiteBnd]]></set-property>
</add-item>
```

4. For each new custom tab, add its `itemViewMapping` ID to the `itemMapping` `viewMapping` property. For example for Tab 1:

```
<set-property name="viewMappings">
<![CDATA[SaIvmNewSiteCategory]]></set-property>
```

For Tab 2:

```
<set-property name="viewMappings">
<![CDATA[SaIvmNewSiteCategory, saIvmSecondTab]]></set-property>
```

Note: Once you have created the view mappings for the new tab, you must import the file in to the `viewMappingRepository`.

5. Create a `siteRepository.xml` file and use XML-combine to append these to the site configuration item descriptor.

   For example, edit your new copy of `/atg/multisite/siteRepository.xml` similar to the following:

```
<gsa-template xml-combine="append">
<item-descriptor name="siteConfiguration" xml-combine="append">
  <table name="New_Table" type="auxiliary" id-column-name="id">
    <property name="New_Property" column-names="New_Property"
         data-type="string" category-resource="NewCategory"
         display-name-resource="New_Property">
      <attribute name="resourceBundle"
         value="atg.multisite.myCategoryResources"/>
    </property>
  </table>
</item-descriptor>
</gsa-template>
```

6. Create a repository item specifying the item mapping name to use.

Adding a New Property

Adding a new configurable property to the site category requires updating both the database and Site Repository on the management and production schemas. Additionally, the corresponding Web application must be updated to accommodate this new property.

Important: Do not modify the existing default `viewmapping.xml` and `siteRepository.xml` files.

1. Create a `siteRepository.xml` file.
2. In the relevant category item mapping, include or exclude your new property. If you are defining a custom `propertyView`, you must update the `itemViewMapping.propertyViewMappings` to include the mapping from your new property to your `propertyViewMapping`. Refer to the Important Warning regarding modifications to `propertyViews` and `propertyViewMappings`.
3. For each new property, specify a `display-name-resource` attribute. This attribute should point to a resource bundle that is available in your `classpath`.

   Note: Ensure the new resource bundle file has a unique name that does not overwrite an existing resource bundle.
4. Set the property information as necessary.

Important: ATG best practices suggest that you do not set properties to `required` in the `siterepository.xml`. Instead, use the `required` `viewMapping` attribute to set the property to `required`.

Note: Ensure that the database column for this property is not set to `Null`.

5. Create the database table necessary to represent the above table. Load the data into the database.
6. Define any default property editors that are needed to edit the property detail. Refer to the *ATG Merchandising Administration Guide* for information on creating property editors.

If you are adding a new property to an existing site category, sites that use the category are automatically updated to include the new property. If you are using the `includeProperties` attribute you must manually add the new site configuration property to the `includeProperties` attribute on the appropriate item view mapping. You must also update the database and Site Repository on both the management and production schemas. Refer to the *Add Items* section in the *ATG Repository Guide* for additional information on adding properties.

If you are adding a new table, ensure that the table contains an ID column. If your property requires the creation of a new item type, refer to the *SQL Content Repositories* section of the *ATG Repository Guide.*

Excluding Properties

You can use item mappings to exclude properties or categories of properties from site categories. For example, to exclude ATG Commerce properties from appearing on any tab, specify `excludedCategories=commerce` in the `ItemMapping` `AttributesValues`. You can also use the `excludeProperties` attribute to prevent specific properties from appearing.

To exclude properties from appearing on a specific tab, use the `ItemViewMapping` `AttributeValues` `excludeProperties` and `excludeCategories` attributes.

For example:

```
<set-property name="attributeValues">
  <![CDATA[resourceBundle=AmAvSiteBnd,excludedProperties=
  aMgrAvSiteCategoryNewExcludedProps]]></set-property>
</add-item>

<add-item item-descriptor="attributeValue"
  id="aMgrAvSiteCategoryNewExcludedProps">
  <set-property name="value">
    <![CDATA[siteDownURL,preOpeningURL,siteIcon]]></set-property>
```

Note: If a property is set as required, but excluded from a site category, you will not be able to deploy or create sites. Ensure that all properties that you exclude from a category are not also set as required.

The `exclude` attributes are a comma separated list of properties with no spaces between the properties. For example:

```
ancestorCategories,catalogsRelatedProducts,computedCatalogs
```

Important: When a property is excluded, no value for that property will be included in the site configurations. If, however, a default value for a property has been set at the repository level the value will be included in the site configuration even though the property may have been excluded from the site category. It is an ATG best practice to avoid setting default values for site properties at the repository level.

Including Properties

If you add properties but do not use the `includeProperties` attribute, new properties are automatically shown on the first tab that is displayed. When using the `includedProperties` attribute, new added properties will not automatically show by default. You must include the properties specifically.

To define which properties or category of properties show on a specific tab, use the `ItemViewMapping AttributeValues includeProperties` and `includeCategories`. You can also show all other undefined properties on a specific tab using the `ItemViewMapping AttributeValues showNewProperties` attribute.

For example:

```
<![CDATA[resourceBundle=AmAvSiteBnd,includedProperties=
  aMgrAvSiteCategoryeNewIncludedProps]]></set-property>
</add-item>

<add-item item-descriptor="attributeValue"
  id="aMgrAvSiteCategoryNewIncludedProps">
<set-property name="value">
  <![CDATA[siteDownURL,preOpeningURL,siteIcon]]></set-property>
```

The site configuration repository categorization determines the headings under which a property is displayed in a tab. If no category is set then the properties will be shown in an uncategorized category for that tab.

Removing a Property

When a property is removed from a site category, it must be removed from all existing site configurations that are based on the site category. This includes removing it from the category view mappings.

The database and Site Repository on the management and the production schema must also be updated to ensure that the property is removed properly.

Creating Required or Read Only Properties

You can set properties within the site category as `required` or `read only`. Using view mapping to set a property as required, add a `required` attribute and set the attribute to `true`. To set a property as read only set the `isReadOnly` property to `true`. The following is an example of both property specifications:

```
<!--You can define this once, and reuse it for any property that you want to mark
    as required-->

<add-item item-descriptor="propertyViewAttributeDefinition"
    id="SaPvRequiredTrue">
  <set-property name="defaultValue"><![CDATA[true]]></set-property>
</add-item>

<!-- Using Description as an example – note the required attribute, and
    the isReadOnly property-->
<add-item item-descriptor="propertyView" id="SaPvSiteDescription">
  <set-property name="name"><![CDATA[Site Description
    propertyView]]></set-property>
  <set-property name="description"><![CDATA[Site Description
    propertyView]]></set-property>
  <set-property name="mode"><![CDATA[AmMmDef]]></set-property>
  <set-property name="type"><![CDATA[string]]></set-property>
  <set-property name="viewComponent"><![CDATA[atg.editor.view
    property.DefaultEditor]]></set-property>
  <set-property name="isComponentPropertyView"><![CDATA[false]]>
    </set-property>
  <set-property name="isDefault"><![CDATA[false]]></set-property>
  <set-property name="isReadOnly"><![CDATA[true]]></set-property>
  <set-property name="attributes"><![CDATA
    [inputFieldMaxlength=AmPvadIntInputFieldMaxLength,
    inputFieldMaxWidth=AmPvadIntInputFieldMaxWidth,
    required=SaPvRequiredTrue]]></set-property>
</add-item>
```

Important: To configure a property, ATG best practices suggest that you add the properties in the view mapping, and not the repository. Any modifications to properties that you make in the repository will apply to all sites.

Adding Default Property Values

Site categories allow you to define different default values for a property on a per-category basis. These values are associated with the property in the repository.

To add a default value, create a copy of the `siteConfiguration` property in the site category, and then set its value in the site category. Whenever a site is based on that category, the value will be copied over to the `siteConfiguration` property of the same name.

For example, if you have a property in the `siteConfiguration` named `property1`, create a property in your `siteTemplate` item in the `siteRepository.xml`, also named `property1`. Update the property with your default value and create the `property1` column in your database.

When an existing site category is updated, the Site Administrator will be made aware that changes have been made and will be presented with the new property. The site must be deployed to production before the property can be used.

Configuring Enumerated Strings

When creating a configurable property, you can create a list of values of possible values. The `enumerated string` property type allows you to specify a single-value property or a collection.

For example, when you specify a collection, a user can select any number between 0 and the number of potential values available. With a single-value property, the user can select 0 or 1, unless the property is set to required, and then the user must select 1. For additional information on enumerated strings, refer to the *ATG Repository Guide*.

Note: Performance issues may occur if you create lists with large amount of data. Use enumerated strings to display a limited number of items.

Enumerated strings work in the same way as normal enumerated properties; however, they save a string to the database instead of an integer. Except where values are specified using option tags in the `SiteRepository.xml` file, values are provided by a `StringEnumProvider`, which can be customized for your configuration.

The `stringEnumProvider` returns two arrays: the first array is the code that is saved in the database, and the second array is the value displayed by the UI. When a user saves a shareable type, it is only successful if the code being saved exists in the array of codes returned from the `stringEnumProvider`.

For example:

```
$class=atg.adapter.gsa.StringEnumProviderImpl

codes=code1,code2
resources=value1,value2
```

To specify a list of values, the `data-type` property, or `component-data-type` within your `siteRespository.xml` file must set to `enumerated string`, and you must provide the property with the values by adding the following attribute that calls a nucleus component path:

```
<attribute name="stringEnumProvider" value="location of
  StringEnumProvider "/>
```

The attribute `value` is an implementation of the `atg.adapter.gsa.StringEnumProvider` interface. This default class can have its values appended to array properties.

For example, Site Administration uses enumerated strings in the Site Repository for site groups. A site group is made up of a collection of sites and a collection of `shareableTypes` that are registered with `/atg/multisite/SiteGroupManager`. As such, the site group `shareableTypes` property is of type `enumerated string`, and its `stringEnumProvider` returns the `shareableTypes` from `SiteGroupManager`. This allows Site Administration to validate, at the repository level, that only a valid shareable type can be saved in a site group:

```
<table name="site_types" type="multi" id-column-names="id">
  <property name="siteTypes" column-names="site_type" data-
      type="set" component-data-type="enumerated string" category
      resource="categoryBasics" display-name-resource="siteTypes">
    <attribute name="stringEnumProvider"
      value="/atg/multisite/SiteTypesProvider"/>
    <attribute name="propertySortPriority" value="100"/>
  </property>
</table>
```

Whenever an enumerated string is used, the corresponding Web application must be updated to process the property values.

Internationalized Enumerated Strings

If you need to create an internationalized enumerated string, ensure that your resource property contains the same array of resource keys as the resource keys in your resource bundle.

For example:

```
$class=atg.adapter.gsa.StringEnumProviderImpl

codes=code1,code2
resources=value1,value2
resourceBundle=atg.multisite.newResourceBundle
```

Aside from the resource keys, you create internationalized enumerated strings in the same manner as regular enumerated strings.

Note: This method of internationalization is used only when employing `StringEnumProviders`. If using `option` tags, specify the resources in the `siteRepository.xml` file as you would with a normal enumerated property.

Deleting Enumerated Strings

To delete an enumerated string, remove the string from the appropriate file. Note that if you have any saved assets that reference the deleted string, when the asset is next updated, the system will remove saved values that are no longer in the enumeration.

For additional information on enumerated strings, refer to the *ATG Repository Guide*.

Site Category Types

There are two types of site categories: *inclusive* or *exclusive*.

Inclusive site categories include all of the configurable properties in the system by default. Because all properties are included, you must specifically exclude properties that you do not want appearing in the site category.

The ATG best practice is to specifically exclude all configurable properties that are not required for the type of site that the site category supports. The default site category shipped with Site Administration is inclusive, but can be modified to exclude specific properties.

Exclusive site categories exclude all of the configurable properties in the system by default. With an exclusive site category, you must specifically include all configurable properties that are required for the type of site that the site category is required to support.

Exclusive site categories can append or ignore new configurable properties:

- Append New Properties - When a new configurable property is added to the system, it is automatically added to the site category and available to the site administrator to set as part of the site process.
- Ignore New Properties - When a new configurable property is added to the system, it is not automatically added to the site category. The new configurable property is only available to the site category if specifically included.

Important: It is an ATG best practice to avoid setting default values for site properties at the repository level. When a property is excluded from a site category, no value for that property will be included in your site configurations based on the site category. If, however a default value for a property has been set at the repository level, then the value will be included in the site configuration even though the property may have been excluded from the site category.

Grouping Site Categories

Site categories can be organized into groups using standard view mapping functionality. This enables Site Administration to present configurable site properties in logical tabs and logical categories of properties. Item view mappings allow you to create groups that are tab-based.

For each site category, the associated configurable properties are assigned to groups, such as `general`, `payment`, `shipping`, etc. These groups allow Site Administration to present a site's configurable properties in a logically grouped tab-based manner in a workspace.

There is no limit to the number of groups that can be defined as part of the site categories registration process, however ATG best practice suggests limiting the number and naming of groups to a consistent set.

Combined, site categories groups and site property categories allow Site Administration to present configurable site properties in logical tabs and, within each tab, logical categories of properties.

Repository-level property categorization, using the `category-resource` attributes of the property in the repository definition, allows you to create a group of properties within a tab.

Refer to the Default Site Category Configuration section for additional information on creating categories.

Implementing Site Icons

The `SiteRepository` contains two string properties that access site icons: `siteIcon` and `Favicon`. These properties contain a path to an uploaded image file.

It is recommended that you set a default value for the image properties that point to image directories that are the start path for all images.

For example, create an image directory such as `/images/storefront/site/home/` that can store all images for a site. For each new site, you would only have to fill out the filename part of the path, such as `/images/storefront/site/home/facicon.ico` or `/images/storefront/site/home/siteIcon.png`.

10 Configuring URLs

URL management in Site Administration includes not only the strategies for developing a URL structure, but also the configuration of URL failover, context roots, virtual context roots and validation.

When developing a URL strategy, it is important for both the site developer and the site administrator to coordinate to ensure that all of the components necessary for implementing URLs have been created and configured.

This section contains the following:

URL Strategies

Each ATG Server must have a unique URL for each site; however, your site configuration may contain customer-facing production sites as well as staging, preview and agent sites. As such, it is important to create a pattern that allows you to not only manage URLs but to help you identify internal URLs versus customer-facing production URLs.

When developing a URL strategy, you can create both domain name-based URLs and/or path name, context root URLs.

Domain-Based URLs

Domain Name URLs use a fully qualified domain name in the URL name for a site. This may or may not include paths. For example, `www.company.com` is a fully qualified domain name. This strategy may also use a subdomain, such as `one.company.com` and `two.company.com`.

Because the entire fully qualified domain name is used across different servers that are accessing different domain names (for example a staging domain) the URL value must be transformed for use on other servers.

You can create a URL transformation rule for each server that takes the name of the staging site base URL and transforms it into the production site base URL. These transformed URLs are used for staging requests to site mappings, as well as across absolute URL links. For information on creating URL transformation rules, refer to the *ATG Programming Guide*.

Path- Based URLs

Path-based URLs can be either context root URLs, or a combination of context root and directory paths. Path-based URLS share the same domain name, but are identified with the context root component of the path, which start with a / (forward slash). For example:

```
www.site.com/SiteA
www.site.com/SiteB
```

Because both the context root and path values remain constant per site, there is no need to create URL transformation rules.

Site Administration URLs

When configuring a site using Site Administration, the following multisite-specific URLs exist.

- Site Base URL
- Context Roots
- Alternative URLs
- Path-based URLs

Site Base URL

Site base URLs resolve a browser request from the Web application.

The site base URL:

- Determines the site context for each production request
- Provides the base string value for the absolute URL link generation
- Provides the starting point string that is transformed into staging URLs
- Is associated with a site ID and used to create a mapping pair that is used by the site determination logic.

Site base URLs use server-side validation to determine if the URL is unique. By comparing the URL against the current and other additional and redirect URLs, the URL is identified as unique. The site base URL may or may not be the public URL for the site.

Note: The site base URL must be unique across all sites.

In some cases, the site base URL may be a virtual context root that does not map to a physical Web application. In these circumstances the context root URL will be used to record the actual context root.

For information on setting up validation for site base URLs, refer to the Site Base URL Validation section.

Site Base URLs and JSPs

When you are creating JSPs, you must include the intended site's site base URL when creating links. Use the `SiteLinkDroplet` when constructing links, as this ensures consistent and proper link generation across sites and across preview, staging and production servers. For information on the `SiteLInkDroplet`, refer to the *ATG Page Developer's Guide*.

Context Roots

A context root allows you to use a single application for multiple sites when using a path-based strategy for site base URLs by creating a path mapping to a Web application. You can map a standard context root to your site that was configured for your application server, or you can create virtual context roots.

Virtual Context Roots

A virtual context root is a context root that does not map to a physical Web application. If the Site Base URL or Alternative URL field uses a virtual context root, the Context Root URL field should be configured to contain the value for the actual context root that maps to a physical Web application.

Virtual context roots allow you to map multiple context roots to a single Web application, allowing you to use a single site to serve multiple sites. For example, if you have a Web application whose context root is `/sports` but you want to create two separate site configurations that use this Web application, you could set the context root site property to `/sports` for each site configuration and set the site configurations' Site Base URLs field to virtual context roots such as `/hockey` and `/basketball`.

By default virtual context roots are disabled. To enable them, you must set the `enabled` property to `true` in the `VirtualContextRootsService.properties` file.

Virtual context root mapping is repository-based. You can extend your site category to use a default context root. If the default context root is available, it is included in any sites created with the category. When configuring a new site, use the Site Base URL field to supply the virtual context root string that maps the site to the actual context root.

To associate the context root property with the virtual context root, the site developer must map the two in the Site Repository as well as configure the appropriate `web.xml` file. For additional information on configuring virtual context roots, refer to the *Multisite URL Management* section in the *ATG Programming Guide.*

Alternative URL

More than one URL can map to the same site base URL. Note that the site base URL may not be the public URL as the site base URL value is set to `null` by default.

The list of alternative URLs provides a direct mapping to site context and allows you to add additional URLs whenever needed. These URLs do not need to follow a specific pattern, but you should choose values that do not conflict with the site base URLs.

Additional URLs are only used for runtime site context determination, are not used for link generation and do not require transformation rules.

Redirection URL

Three redirection URL properties are available by default:

- Site Disabled URL
- Pre-opening URL
- Post-closing URL

These URLs redirect browser requests if the site base URL is unavailable. Redirection URLs are always entered as full domain URLs and are always validated to ensure that you cannot save a URL that already exists in the current site base or alternative URL.

Absolute Link Generation

Some components require an absolute URL link that starts with a fully qualified domain name or a forward slash (`/`). Absolute URLs are constructed using the `SiteLinkDroplet`. Refer to the *ATG Page Developer's Guide* for further information on using this droplet.

Using the site base URL, the `SiteLinkDroplet` constructs a URL. Once a component obtains the site base URL, the component appends the site base URL with a relative path to generate the full absolute URL. If you are using a path-based URL strategy, the path or context root is used.

Production Server Link Generation

On a production server, the `SiteLinkDroplet` returns the site base URL fully qualified domain name. The path can be appended to the site base URL to create an absolute URL.

Staging Server Link Generation

Staging servers follows the same logic as the production server; however, the runtime site base URL cache may have been overwritten with transformed URL values using transformation rules. These transformed site base URL values have been transformed specifically for the staging server and are used by the `SiteLinkDroplet` in link generation.

Preview Server Link Generation

On a preview server, no URL pattern matching is performed and there are no transformation rules used. The `SiteLinkDroplet` on a preview server appends the `pushSiteParam=` query string to allow the preview server pipeline to address the request. The `SiteLinkDroplet` ignores the site base URL value but concatenates the additional path and the `pushSiteParam` query string variable for the domain, server name, or IP number.

URL Validation Types

Three types of URL validation occur within Site Administration:

- URL Uniqueness
- URL Format
- Redirect URL Duplication

URL Uniqueness

URL uniqueness is applicable to site base and alternative URLs only. If an administrator enters a site base or alternative URL that is not unique, the system presents a warning.

Note: The administrator may choose to ignore this warning; however, there will be no further validation of URL uniqueness. In this situation, it is important to note that the administrator may accidentally deploy URL duplicates to the production system.

URL Format

URL prefixes such as `http://` and `https://` are ignored for URL format validation and are not stored as part of the URL in the system.

The URL domain prefix `www` is included when performing both URL validation and storage. Note that `www` is not a mandatory prefix to a full domain. Many forms of full domain are valid, such as:

```
yellow.company.com
www.company.com/yellow
my.company.com/yellow
```

The format validation for URLs is limited to checking if the URL either begins with a `/` (forward slash) or else contains a `.` (dot) before the last character. If a URL fails this limited format validation, the system will return an error message and the administrator will not be allowed to save the URL.

Encoding URL Special Characters

When URLs that have been recorded in Site Administration and stored in the repository are used for generating links, problems may occur if these URLs contain special characters. As such, URL special characters should be replaced by their corresponding hexadecimal equivalent code. For example, the URL `www.example.com/this & that/pages/10%off.jsp` should be entered as `www.example.com/this%20&%20that/pages/10%25off.jsp` where the spaces have been encoded to `%20` and the percentage symbol has been encoded to `%25`.

Note: Site Administration will not automatically encode URLs that contain special characters. This avoids any possibility of re-encoding URLs that have already been entered by the user. It is assumed that all URL fields are populated in an encoded format.

Redirect URL Duplication

If a site base URL is not available, the site can be redirected depending upon the state of the site.

Redirection URL Site States

A site item can exist in a combination of four states:

- Enabled
- Enabled - Pre-launch
- Enabled - Post-closing
- Disabled

These states are managed using Site Administration and persisted in the Site Repository. The states also determine access to the site by either public users or ATG applications. The states correspond to the following site properties:

- Enabled Status – Site can be set to either `Enabled` or `Disabled`
- Opening Date – The date value that marks the opening date of a site
- Closing Date – The date value that marks the closing date of a site

Based on the derived state from the above properties combinations, redirection occurs to a URL that has been defined in the current site configuration. These correspond to the following site properties, which should be complete URLs.

- Site Down URL
- Pre-launch URL
- Post-closing URL

If the site has been disabled or is temporarily down, the browser will be redirected to the site down URL. If the site is enabled or the opening date is in the future, the browser will be redirected to the Pre-Launch URL. If the site is enabled and the closing date is in the past, the browser will be redirected to the post-closing URL.

Configuring Validation

You can configure the URL properties to provide custom validation by using the ATG View Mapping system to override the default property editors.

The site base and alternative URLs implement client-side URL form validation, ensuring that the URL:

- Starts with a `/` (forward slash) character as a context URL, for example `/site12`
- Contains characters after the last `.` (dot) in the URL, such as `www.site12.com`
- Does not start with the `http://` protocol

The URLPropertyValidator.as class implements the atg.assetmanager.editor.view.property.validate.PropertyValidator interface.

Format Validation Configuration

To configure a URL property to use the URLPropertyValidator, update the viewmapping.xml file to use the PropertyViewAttributeDefinition.

For example:

```
<add-item item-descriptor="propertyViewAttributeDefinition"
    id="SaAvURLPropertyValidator">
<set-property name="defaultValue">
<![CDATA[atg.assetmanager.editor.view.property.validate.URLProperty
    Validator]]>
</set-property>
</add-item>
```

The propertyView of the editor implements the validation process, then sets the property view attribute definition in its list of attributes.

For example:

```
<set-property name="attributes"><![CDATA[propertyValidators=
    SaAvURLPropertyValidator]]</set-property>
```

The SaAvURLPropertyValidator can be used in any propertyView that needs URL format validation.

Site Base URL Validation

The site base URL property editor uses the DefaultEditor.as class that implements the URLPropertyValidtor for validation. This property is configured in the viewmapping.xmlfile by creating a new propertyViewMapping and propertyView:

```
<add-item item-descriptor="propertyViewMapping" id="SaPvmSiteURL">
  <set-property name="propertyView"><![CDATA[SaPvSiteURL]]>
    </set-property>
  <set-property name="description"><![CDATA[Site URL
    PropertyViewMapping]]></set-property>
  <set-property name="attributeValues"><![CDATA[]]></set-property>
</add-item>

<add-item item-descriptor="propertyView" id="SaPvSiteURL">
  <set-property name="name"><![CDATA[AssetManager Site URL
    propertyView]]></set-property>
```

```
  <set-property name="description"><![CDATA[AssetManager Site URL
    propertyView]]></set-property>
  <set-property name="mode"><![CDATA[AmMmDef]]></set-property>
  <set-property name="type"><![CDATA[string]]></set-property>
  <set-property name="viewComponent"><![CDATA[atg.assetmanager
    editor.view.property.DefaultEditor]]></set-property>
  <set-property name="isComponentPropertyView"><![CDATA[false]]>
    </set-property>
  <set-property name="isDefault"><![CDATA[false]]></set-property>
  <set-property name="attributes"><![CDATA
    [customAssetPropertyService=SaPvAssetPropertyService,
    propertyValidators=SaAvURLPropertyValidator,inputFieldMaxlength
    =AmPvadIntInputFieldMaxLength,inputFieldMaxWidth=AmPvadIntInput
    FieldMaxWidth]]></set-property>
</add-item>
```

To use the new `propertyMapping`, add the following `propertyMapping` to the item's `itemViewMapping`:

```
        <set-property name="propertyMappings">
            <![CDATA[productionURL=SaPvmSiteURL]]></set-property>
```

The URL validation process uses the `SiteAdminAssetPropertyService`, which is configured in the `viewmapping.xml` file with the following attribute:

```
<add-item item-descriptor="propertyViewAttributeDefinition"
  id="SaPvAssetPropertyService">
<set-property name="defaultValue"><![CDATA[/atg/remote/multisite
  /service/SiteAdminAssetPropertyService]]>
</set-property>
</add-item>
```

The `propertyVIewAttributeDefinition` is added to the `propertyView` attributes tag:

```
        <set-property name="attributes">
        <![CDATA[customAssetPropertyService=SaPvAssetPropertyService]]>
        </set-property>
```

The `validateAssetVIew` method will acquire the `SiteAdminAssetPropertyService` class.

Overriding Default Validation Rules

To override the default server-side validation rules, you must override the `RespositoryViewServiceImpl.validateAssetView` method in the `AssetPropertyService` class. This class is defined in the `viewmapping` file. The `AssetPropertyService` class runs an extended `SiteAdminPropertyEditorAssetViewService` that overrides the default validation method.

Alternative URL Validation

Alternative URL validation is performed using the `PrimitiveCollectionEditor`. This client-side validation differentiates between collection editor properties and single-valued properties. When text is entered into the alternative URL field and saved, the `updatedCollectionPropertyValue` method runs the appropriate validation.

The validation determines if the URL entered in this field is unique by comparing them against the current site base and redirect URLs. Additionally, the URL is compared against all other sites' site base URL, additional URL, and redirect URLs.

Redirect URL Validation

Redirect URL validation uses the `DefaultEditor`. The server-side validation determines if the URLs are unique by comparing it against the current site base and alternative URLs, as well as against all other sites' site base URLs and alternative URLs.

If a site's redirect URL property value (Pre-Opening, Post-Closing, or Site Disabled) contains a string that matches that site base URL (or any alternative URL), the browser will be redirected to the same site context, which will perform another redirect. The cycle will repeat until the user makes an appropriate browser action, such as a close request. This condition may also occur when a redirect URL points to another site's site base or alternative URL that is also redirecting.

To avoid these possible infinite loop scenarios the following validation rules apply:

- Redirect to current site - When a redirect, site base or alternative URL is created or amended, the system validates that there is no duplication between the redirect URLs and the main (site base and alternative) URLs. If a duplicate is detected, the system returns an error message and will not allow the URL to be saved.
- Redirect to other site - When a redirect URL is created or amended the system compares the URL to all existing site base and alternative URLs. If a match is found, the system determines the site state for the matching site. If the site state is not `Enabled`, the system returns an informative warning message alerting the user to the implications of redirecting users to a site that may in turn redirect the user or return an error.

If any of the redirect URLs do not contain a URL, a 404 Server Not Found error will occur.

Managing URLs

URL management provides a way for you to manage large number of URL patterns that may be needed for the production, staging and preview of multiple sites.

Using the domain strategy and/or context root URL strategy, the administrators configure the `SiteContextRuleFilter` to identify the type of pattern and match it with existing URLs.

For detailed information on URL Management, refer to the *ATG Programming Guide.*

Defining Explicit URL Patterns

Whenever a request occurs for URL mapping, the site ID must be associated with the URL. URL patterns are based upon site context. Sites contain site IDs and site attributes that are accessed by `siteContext` objects. In turn, these objects are managed by the `SiteContextManager` component. The `SiteContextPipelineServlet` handles incoming requests and determines their `SiteContext`. For each request, the `siteContext` is set using the `siteContextRuleFilter`, which uses the `URLPatternMatchingRuleFilter` to map URLs to site IDs.

This filter determines the site ID by referring the URL against the configured URL map property. If no match is found, the URL is passed to the `SiteURLManager.getSiteIdForURL` method.

By default, the `URLPatternMatchingRuleFilter` is enabled. The `enableSimpleAlgorithm` property enables the algorithm that matches the URL against the URL map. The `enableSiteURLManagerAlgorithm` property performs the more complex algorithm that matches the values generated from the Site Repository.

The `URLPatternMatchingRuleFilter` also checks if the associated Web application has the `atg.multisite.URLPatternMatchingEnabled` parameter set. By default, this parameter is set to `true`. If set to `false`, the `URLPatternMatchingRuleFilter` will not execute.

Ensure that the `URLPatternMatchingRuleFilter` is enabled in your URL mapping properties file and your Web application `web.xml` file.

The following is an example of a configuration of a URL map:

```
$class=atg.multisite.URLPatternMatchingRuleFilter
urls=\
    /webapps/site1App=site1,\
    /webapps/site4App=site4,\
    company.com=site7,\
    company.com/webapp8=site8
siteURLManager=/atg/multisite/SiteURLManager
enabled=true
enableSimpleAlgorithm=true
enableSiteURLManagerAlgorithm=true
```

The following is an example of a `web.xml` file;

```
<web-app>
  ...
  <context-param>
    <param-name>atg.multisite.URLPatternMatchingEnabled</param-name>
    <param-value>false</param-value>
  </context-param>
  ...
</web-app>
```

For additional information on site context, the `SiteContextPipelineServlet` and the `URLPatternMatchingRuleFilter`, refer to the *ATG Programming Guide*.

11 Configuring Site Types

You can reference a site that is based on a specific site type. For example, when working with an ATG Service site, you may want work with only ATG Self Service sites. Site types allow you to differentiate between ATG application sites. However, a site can also be a combination of multiple site types.

For example, you may have a site that presents ATG Commerce information, and another that presents ATG Self Service information. You can configure your site to use a single site object for both sites, or configure the sites separately, with separate site categories for each type of site.

The `siteType` property ensures that only the correct types of sites are displayed. This property is defined in the `SiteRepository` `SiteConfiguration` item descriptor.

If the property contains no identifier, it is perceived by `siteManager` to be a universal site. If the property does contain an identifier, applications that require specific site types will verify if the property contains their required `siteType`. If it does, the application will make the site available. If it does not contain the required `siteType`, the site is filtered out.

The Site Administration interface uses the `StringEnumProvider` component, defined within the `siteTypes` property, to limit the elements of the `siteTypes` property to the valid site type values defined in the `siteManager`. Enumerated string values are added to the `siteTypes` array by the installed modules.

For additional information on `siteManager`, refer to the *ATG Programming Guide.*

Identifying Valid Site Types

The `siteManager` maintains an array of valid site types. Each application that needs a site type registers with the `siteManager` during startup. For example, a site type of ATG Commerce would be:

```
#/atg/multisite/SiteManager
siteTypes+=commerce
```

The `SiteManager` then exposes three additional methods:

```
/**
 * Returns all sites whose siteTypes property is null or whose
 * siteTypes property includes an element whose values is the
```

```
 * supplied site type name.
 * @param pSiteTypeName The name of the site type.
 * @return The sites whose siteTypes property is null or whose
 * siteTypes property includes an element whose value is the
 * supplied site type name.
 **/
public RepositoryItem[] getSitesByType(String pSiteTypeName);
public RepositoryItem[] getActiveSitesByType(string pSiteTypeName);

/**
 * Returns all of the site types registered with the site Manager.
 * @return The array of valid site type names.
 **/
public String[] getSiteTypes();

/**
 * Sets the valid site type values.
 * @param pSiteTypeNames The array of valid site type names.
 **/
public void setSiteTypes(String[] pSiteTypeNames);
```

The `SiteTypesProvider` is used by the `siteTypes` enumerated string collection to identify the site types that are valid for that property:

```
#/atg/multisite/SiteTypesProvider
$class=atg.adapter.gsa.StringEnumProviderImpl
codes^=/atg/multisite/SiteManager.siteTypes
resources^=/atg/multisite/SiteManager.siteTypes
resourceBundle=atg.multisite.SiteTypeResource
```

The `SiteTypesProvider` component is used by Site Administration to restrict the values that can be stored in the `SiteConfiguration` `siteTypes` property.

Default Site Types

When you create an included or excluded site type property, you can define the site type. If the site type property is included, you can assign a default site type value. For example, you could design a site category for retail sites and the site type value could be set, at a site category level, to default to Commerce.

Note: This property can be overridden by the Site Administrator when creating or amending a site.

The default site types are:

- Null – These are sites that have no site type value and require only standard ATG implementation modules.
- Commerce - Sites that require the ATG Commerce module

- Self-Service - Sites that require the ATG Self-Service module.
- Commerce/Self-Service - Commerce and Self-Service aligned sites that require both the ATG Commerce and Self-Service modules.

If you are using different types of sites, you must specify the valid default site type value in the site category for the `SiteConfiguration` `siteTypes` property.

Note: If you are running an environment where ATG Commerce and ATG Self Service modules are not installed, the ATG best practice is to not include the site type property in any site category, as it cannot be assigned a value. Should other modules be installed later and the site type property added to existing site category, the site type value for existing sites in the system can be changed using Site Administration.

ATG Applications and Site Types

The following ATG products are identified by specific site types. For additional information, refer to the relevant ATG product documentation:

ATG Product	Site Type Used	Description
ATG Commerce	`commerce`	Adds ATG Commerce-specific properties
ATG Merchandising	`commerce`	Uses the `commerce` site type when requesting a list of sites.
ATG Commerce Service Center	`commerce`	Uses the `commerce` site type when requesting a list of sites.
ATG Self Service	`selfservice`	Adds ATG Self Service-specific properties
ATG Service Administration	`selfservice`	Uses the `selfservice` site type when requesting a list of sites.

12 Sharing Data

Site Administration allows you to share data as well as configure and maintain the sharing relationships between sites.

When working in a multisite environment, you can configure sites to share data. Data such as Nucleus components or data objects can be identified as shareable types. Site administrators combine shareable types and sites into a site group, where the shareable types are used by all sites in the group.

This section contains the following:

Working with Shareable Type Components

Shareable type components within Site Administration are implementation-specific rather than site-specific, and can contain a single component or multiple components. Shareable type components are also dependant on the applications and modules that you have installed.

For information on creating application-specific shareable data, refer to your ATG application documentation.

Creating Shareable Types

Shareable types are configured using the `SiteGroupManager` `shareableTypes` property. This property points to `ShareableTypes` nucleus components. Any `siteGroup` repository item references `ShareableTypes` components through their ID property.

For example:

```
#NewShareableType.properties
$class=atg.multisite.ShareableType
id=NewShare
genre=atg.ExampleItem
```

```
diaplayName=New Share ShareableType
description=The New Share ShareableType
resourceBundleLocation=atg.newshare.NewShareResources
displayNameResourceName=displayName
descriptionResourceName=description
```

The `atg.mulitisite.ShareableType` class contains the following properties, as displayed above:

- `id` – a unique value
- `genre` – a classification for a shareable type. For example, you could classify all `ShareableTypes` for Nucleus components as a genre
- `displayName` – a simple name
- `description` – a description of the `ShareableType`
- `resourceBundle` – allows you to set the `resourceBundle` string and set values in the file specified by that location that correspond to `displayName` and `description`. Values specified in the resource bundle take precedence over values specified directly in the properties file.

Setting Up ShareableTypes Arrays

The enumerated string options for the `shareableTypes` property are provided by a `StringEnumProvider`, which retrieves the valid shareable types from the `SiteGroupManager`.

You can set up the enumerated string by creating a new `StringEnumProvider` property file.

For example:

```
$class=atg.adapter.gsa.StingEnumProvider
Values+=/atg/multisite/SiteGroupManager.shareableTypes
```

Once the `StringEnumProvider` is configured, the `ShareableType` selector will use the `EnumCollectionEditor`.

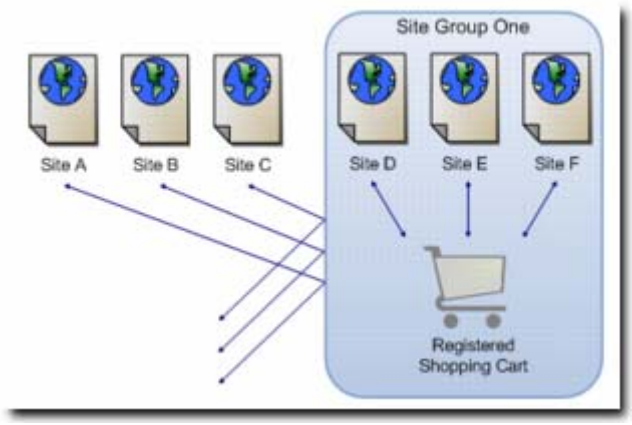
Shareable Types Registration

Shareable type components can be registered or unregistered. Registering components identifies how the component will be shared across all sites:

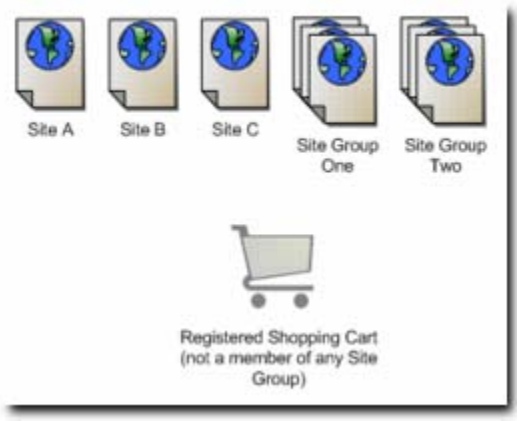
- Components that are not registered will be shared across all sites by default.
- Components that are registered will be shared only when made part of a site group.
- Components that are registered but are not member of a site group are unavailable to any site until added to a site group or are unregistered.

For example, to share a shopping cart across all sites, do not register the component.

To share a shopping cart across the sites within a site group, you must register the component and then add it to a site group. Once the registered component is added to the site group, it is available only to sites within site groups. If a site is not a member of a site group, the registered component will be unavailable.



If you register a component, for example a shopping cart, but do not add it to a site group, the component will not be available to any sites until it is added to a site group.



It is recommended that a shareable type component should be registered with the system only when it is necessary to manage or limit the scope of the shareable type component. For example, if all sites share a shopper's login status, there is no need to define login status as a shareable type component. However, if the login is shared between only a few sites, then it you would register the login status as a shareable type component to apply the limitation.

If a shareable type component is currently registered but needs to be shared across all sites, the shareable type component must be unregistered by the site developer. This excludes the shareable type component from data sharing validation and ensures that it will not be limited within Site Administration.

Registering a Shareable Type

Components are registered with the `/atg/multisite/SiteGroupManager` using the `shareableTypes` property.

For example:

```
shareableTypes+=/atg/multisite/NewShareableType
```

Once you have registered your `ShareableType`, all sites have their own instance of the `shareableType`. However, if the `ShareableType` remains unregistered, the `ShareableType` is shared between all sites.

Once you have registered a `ShareableType`, you can configure site groups that will share the `ShareableType` you have created.

Using Site Groups

Site groups allow site administrators, as well as other applications, to group sites together to share common properties and registered shareable type components, such as a shopping cart.

Sites can belong to multiple site groups. Site groups can contain one or more sites and can contain multiple components, with all shareable components being shared amongst all sites in the site group. Site groups that contain shareable components are also known as Sharing Groups.

Shareables can be a member of many different site groups; however, a site can be a member of only one site group that contains a specific shareable. In other words, a site cannot have a common shareable across site groups. For example, if Site A is part of Site Group One that shares a shopping cart, Site A cannot be part of Site Group Three that shares the same shopping cart.

Site groups are stored as versioned repository assets in the Site Repository and are managed by the `SiteGroupManager` component. The `SiteGroupManager` enables `SiteGroups` and shareable types to obtain information on `ShareableTypes`, `SiteGroup` IDs, and determine if sites share specific `ShareableTypes`. For additional information on the `SiteGroupManager` component, refer to the *ATG Programming Guide*.

The `siteGroup` item descriptor defines the following properties:

```
<item-descriptor name="siteGroup" >
  <property name="id" data-type="string"/>
  <property name="displayName" data-type="string"/>
  <property name="shareableTypes" data-type="array"
    component-data-type="String"/>
  <property name="sites" data-type="array"
    component-item-type="site"/>
</item>
```

The `shareableTypes` property of `siteGroup` is a collection of enumerated strings.

Site Group Validation

Validation occurs whenever a site is added into a sharing group using Site Administration. This validation reviews the sharing group and ensures that the site does not belong to another sharing group that shares the same `ShareableTypes`. This validation will not check multi-component `ShareableTypes`, or `ShareableTypes` that contain more than one component, for example a Shopping Cart and a Product List.

Data sharing validation methods can be configured in the `SiteGroupManager` component. However, these rules will be available only in the Site Administration application. Refer to the *ATG Programming Guide* for additional information on the `SiteGroupManager` component.

Previewing Sites

If you have installed a preview server, the Preview button will be available in Site Administration. When sites are created or modified, they can be displayed in preview mode, which shows the site as it would be presented to a given user on a specific page of a particular website. Additionally, the site context will be automatically passed to the preview launcher based on the site that is currently selected.

You must create the preview URL properties for each JSP that serves as a landing page for that Web application. The `name` value of the properties file identifies the JSP as a landing page. This assists the site administrator by making landing page Preview URLs easily identifiable in the preview launcher.

Note: It is important that the site administrator work with the site developers to identify which preview URLs map to the home page for the site that they wish to preview.

The following components are required for the preview launcher:

- Preview User - The preview is based on a user and will display the site and landing page as it will appear for this user. The users who can be selected for the Preview User parameter are defined by the development team as part of the preview functionality.
- Preview Site - In a multisite implementation any page or other ATG asset could appear on more than one site, so the site context for the preview must be determined.
- Preview Landing Page - This is the JSP page that will be previewed. Each page that can be previewed has an associated preview URL properties file, which contains information that enables the system to preview that page.

For information on configuring a preview server and the necessary properties files, refer to the *ATG Business Control Center Administration and Development Guide*. For information on using the Preview Server in the BCC, refer to the *ATG Business Control Center User's Guide*.

Access Control

To access Site Administration in the BCC, the user must have the `siteAdminUser` or the administrator role, which allows users with this role to access Site Administration without restriction. The `Siteadmin-role-data.xml` contains the role and the access right for Site Administration.

When configuring a new user to access to Site Administration, the user needs the `siteAdminUser` role, the BCC Member User role and the `epubUser` role. Although not all users who use ATG Merchandising will be site administrators, all site administrators must also have access to ATG Merchandising assets.

The `epubSuperAdmin` role allows users to access only the site category Administrator feature without restriction.

Site Administration ships with a pre-configured Site Administrator User that allows access to Site Administration:

Field	Value
First Name	`SiteAdmin`
Last Name	`User`
Roles	`siteAdminUser`
Locale	`en_US`

Email	`siteadmin@example.com`
Password	`siteadmin`
Member	`True`
Login	`siteadmin`

13 Using Site Administration

This section describes the tasks that a site administrator may perform using Site Administration. It contains the following:

Using the Site Administration Interface

Note: Before you can use Site Administration, ensure that you have the correct access control. Access control is set in the BCC under personalization. Contact your ATG administrator to help you with access control.

Accessing the Site Administration Interface

Site Administration is performed from the BCC, where there are two screens that allow you to manage your configurations:

- Manage Site Assets – This screen displays registered sites and site groups. Registered sites are recorded in the Site Repository and have an assigned Site ID. If there are no registered sites, you will not see any sites or site groups.
- Manage Site Categories – This screen displays site categories that are used to provide data for the sites that you create. Site categories are also used to provide configurable properties for your sites.

Although the screens provide access to different site components, the interface is similar. Both screens are divided into separate panes, where different types of information are displayed:

- Browse Pane – This pane displays a list of all of the sites, site groups or site categories that are available. If there are none available, a message will be displayed asking you to create a new asset
- Asset Editing Pane – This pane displays the asset information, such as the ID, Name, Description, etc. This is where you provide information that creates and implements your configuration

The screens also provides dropdown action menus that create or manage assets that you have selected.

For additional information on the user interface, refer to the *ATG Merchandising Guide for Business Users.*

Sorting Assets

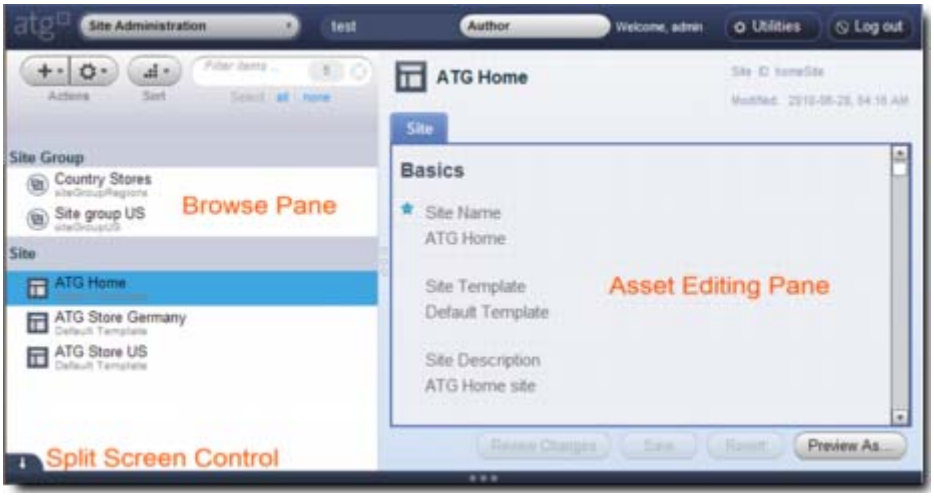
You sort assets in both the site categories and the Site Administration screen by using the Sort menu. When you sort assets on the site category screen, you can sort by site category name or ID. When you sort assets on the Site Administration screen, you can sort by Enabled, meaning whether the site is in the enabled state or not. You can also sort by site name or the site category that has been used to create the site.

Filtering Assets

You can filter assets by entering criteria in the filter field. Your criteria limit what is displayed in the Browse Pane. For example, if you enter `yellow` in the filter field, only assets that contain that word will be displayed.

Comparing Assets

The Manage Site Assets screen contains a split screen control that allows you to split the screen into two separate consoles. This allows you to select two separate assets and view them simultaneously to compare. To split the screen, click on or drag the control upward until the screens are the desired size. To remove split screens, drag the control towards the bottom of the screen until the second screen disappears.



Site States

Sites can be in one of the following states:

- Disabled – The site is not available to external users
- Enabled – The site is available to external users
- Enabled Pre-launch – The site is enabled, but will not be available to external users until its opening date
- Enabled Post-close – The site is enabled, but the closing date of the site has past.

Working with Sites

When you work with a site, you can create, edit, enabled or disable sites.

Creating a Site

Creating a site involves selecting a site category, setting the site properties, saving the site configuration and then deploying the site.

Working with Site Properties

There are a number of default properties that allow you to configure your site. Additionally, there is a default site category that includes these configurable properties. When working with site categories, you can create site categories that contain subsets of these configurable properties.

The site information will change based upon the types of applications that are installed on your servers. For example, if you have an ATG Commerce server installed, you will be asked for ATG Commerce-based configurations. Refer to the appropriate application documentation for information on these application-specific properties.

Default Site Properties

The following are the default site properties for which you provide information:

- Site ID – The Site ID is created automatically by the system, however, you can modify the Site ID if necessary. However, once the site has been saved, you cannot edit the Site ID
- Site Name – The name of the site is displayed to people who use the site. This name is also used in drop downs and site pickers, as well as in Site Administration. The Site Name must be unique
- Site Category – The name of the site category that was used to create this site
- Site Description – The description of the site allows you to provide detailed information, such as "Summer Sales Sites" or "Home Furnishings Site"
- Site Icon – Sites can be associated with icons that display on the site header. The site icon is set by entering the URL for the image, for example `/images/storefront/homesote_logo_small.png`. New icons should be added by the site developers. See the Implementing Site Icons section.
- Favicon – Favicons are 16x16 pixel icons that are used in the Site Picker or in the address bar of the browser. Favicons, like Site icons, are URLs of the location of the icon that have been configured by the site development team. For example, `/images/storefront/site/home/facicon.ico`
- Site Priority – Site priority allows you to identify which sites or site values will be retrieved first
- Site Type – When you are creating your site, you may want to align the site with a specific ATG application. Site types allow you to identify the type of ATG application for the site. For example, the ATG Merchandising application requires ATG Commerce and uses the Site Type setting to make sure that only sites designated as ATG Commerce sites types are displayed.
- Site URLs – There are a number of URLs that must be identified when creating a site. Once you have the site information from the site developer, you can enter the Site Base URL, Alternative URLs, Context Root, or redirection URLs. For information on URLs and URL planning, refer to the Adding URLs section

Note: Before you can create a site using Site Administration, the site developer must configure specific components such as the site properties, site URLs, and shareable component types.

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Create a new site by using the + dropdown menu and selecting Site.

6. Select the site category and press OK. For additional information on site categories, refer to the Site Categories Overview section.
7. Enter the site information and click Create when done.

Editing a Site

Once you have created a site, you can only edit certain fields of the site. For example, you cannot edit the site ID and site categories fields once the site has been saved.

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Select a site to edit.
6. Click the action button and select Edit from the dropdown menu.
7. Edit the site as necessary.
8. When a site is edited, the system saves two new additional fields, Last Modified By and Last Modified Date.

   Note: If the site category used to create the site has been deleted, and the site configuration has been edited, the site category value for the site configuration will change to the default site category.

Importing Sites

It is possible to import sites; however, no validation will occur on imported sites. Because no validation occurs, importing sites could inadvertently cause problems with site categories and other configurable properties and is not recommended.

Adding URLs

When you are configuring the URLs for your site, your ATG servers must have a unique URL for each site. Remember that you can have customer-facing production sites as well as staging, preview and agent sites. It is best to create a pattern that allows you to identify internal URLs versus customer-facing production URLs. Developing a URL strategy should be done in tandem with site developers, who must create rules and site properties for the URLs that you identify. Refer to the Configuring URLs section for additional information on working with URLs.

Your strategy can include creating both domain name-based URLs and/or path name based, context root URLs.

Note that URL names must be unique. When adding URL information into Site Administration, the http: // and https:// prefixes will be ignored. Additionally, any special characters that are included in URLs must be replaced by their special hexadecimal equivalent as Site Administration will not automatically encode special characters.

For additional information on URL strategies, refer to the Managing URLs section or the *Multisite URL Management* section of the *ATG Programming Guide*

Domain Name-Based URLs

Domain-based URLs use a fully qualified domain or subdomain name in the URL name for a site. This may or may not include paths.

For example:

```
www.yellow.com
www.blue.com
yellow.allcolorstore.com
blue.allcolorstore.com
```

Path Name-Based URLs

Path-based URLs can be either context root URLs, or a combination of context root and directory paths. Path-based URLS start with the same domain name, and are identified with a / (forward slash).

In the following example, all sites are based off the `www.allcolors.com` domain:

```
www.allcolors.com/color/primary/yellow
www.allcolors.com/color/yellow
www.allcolors.com/yellow

www.allcolors.com/colors/yellow
www.allcolors.com/colors/blue
www.allcolors.com/colors/green
```

URLs for Site Administration

Your URL strategy should incorporate the following multisite-specific URLs.

Site Base URL Configuration

Note: If the site URL includes a non-default port, it must be explicitly specified when you set that URL in Site Administration.

Site base URLs resolve a browser request from the Web application. The site base URL may or may not be the public URL for the site, but it must be unique across all sites.

In some cases, this URL may be a virtual context root that does not map to a physical Web application. If so, the context root URL records the actual context root.

Context Root URL

If the site base URL or alternative URL is a virtual context root (a context root that does not map to a physical Web application) this context root URL records the value for the actual context root that maps to a physical Web application.

For example, a site with an ID of `yellow-site` could have a Site Base URL of `/yellow`, but there may not be a Web application configured that has a context root of `/yellow`. As such, `/yellow` is a virtual context root. You identify the actual context root you want the `yellow-site` to use. For example you might choose to have the `yellow-site` map to the contents of a Web application with a context root of `/colors`, in which case the Context Root property will be set to `/colors`. So a request to `/yellow/index.jsp` will actually be served up by the `/colors/index.jsp`, but with the flavor of the `yellow-site` site.

To associate the context root property with the virtual context root, the site developer must map the two in the Site Repository as well as configure the appropriate `web.xml` file. Refer to the *ATG Programming Guide* for detailed information on creating and configuring virtual context roots.

Alternative URLs

More than one URL may map to the same site base URL. The alternative URL provides additional URLs that map to the site base URL When you add alternative URLs, you enter the value of the URL, for example, `www.allcolorstore.com` or `www.allcolorstore.com/colors`. You must also identify the priority of each alternative URL.

Redirection URLs

There are three redirection URL properties that are available by default:

- Site Disabled URL – This URL will be displayed to the user if the site has been disabled or taken off line
- Pre-opening URL – Displays a URL that is presented before a site has been enabled or not yet been activated
- Post-closing URL – This URL displays if the site has been closed and disabled

These URLs can also be used to redirect users if the Site Base URL is not available. Redirection URLs are always entered as full domain URLs, for example `www.allcolorstore.com/sales/blue/october`.

Identifying URLs

Note: URLs must already be created and configured within your Web application.

To identify the URLs for your site:

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Select a site to edit.
6. Click the action button and select Edit from the drop down menu.
7. Enter the site base URL that was created by your site developer.
8. If you have created them, enter the alternative URLs and the priority in which you want these URLs to be used.
9. If you have a URL that displays when your site is disabled, enter the site disabled URL.
10. If you are creating sites that have specific opening and closing dates, enter a pre-opening URL and post closing URLs for the site.

Working with Shareables

Shareables are a logical grouping of data that represents a feature, status or other element that allows site visitors to navigate between sites while remaining in a single session. Examples of shareables are the Shopping Cart, Login Status or a Gift List.

A shareable can be:

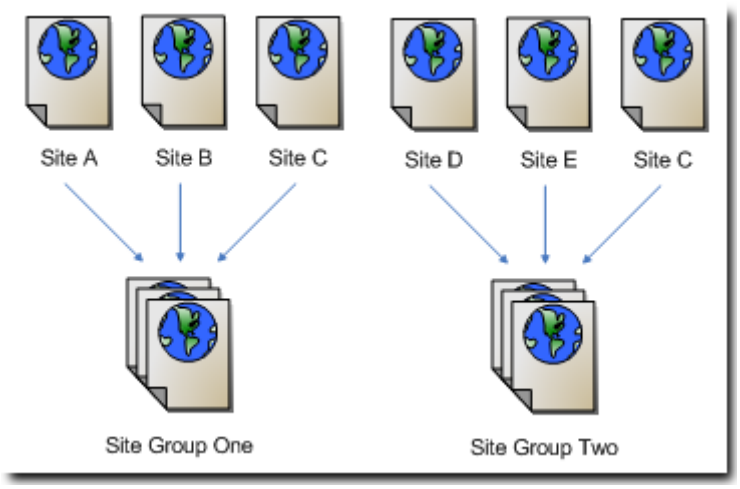
- Single-Component – contains a single defined element, such as a shopping cart or a product list
- Multi-Component – contains more than one element, and allows you to manage dependencies between shareables. For example, if you want all sites that share a shopping cart to also share a product list, you would create a multi-component shareable that includes both components. Any site in a site group that contains this multi-component shareable will share both the shopping cart and the product list

Shareables are configured by the site developer and registered with the Site Repository. Once a shareable has been registered, it can be added to a site group using Site Administration. ATG provides a default shareable, which is a multi-component shareable that includes the shopping cart and the product list components.

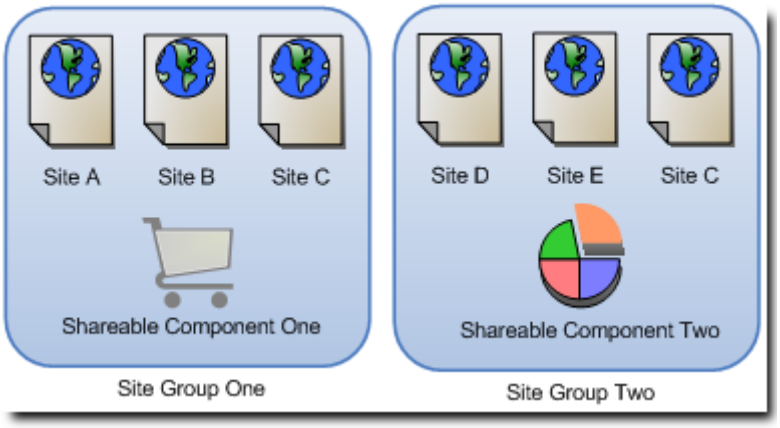
For additional information on registering shareables, refer to the Sharing Data section.

Working with Site Groups

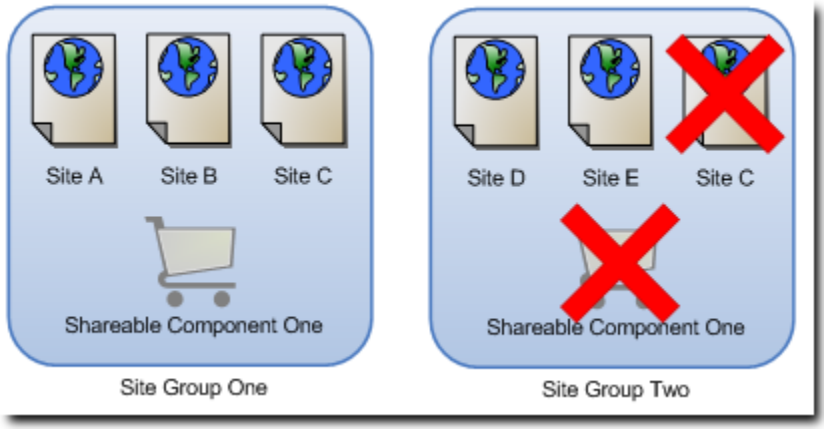
Site groups are a logical grouping of sites that allow you to administer and manage sites that are similar, or to set up sharing between sites. You can add as many sites as necessary to a site group, and sites can be added to multiple site groups.



Site groups can also be used to set up sharing specific shareables between sites. When you add registered shareables, such as a shopping cart, to a site group, it is shared with all of the sites within the group. Site groups that have a shareable are also known as sharing groups.



Site groups will share the registered components with all of the sites within the group. However, you cannot add a site to a site group that contains a shareable that it shares with another site group. For example, If Site Group One and Site Group Two both use Shareable Component One, Site C cannot be a member of Site Group Two:



Creating a Site Group

When you create a site group, any shareables that you add are shared between all sites within the group.

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Click the + icon from the Actions menu and select Site Group
6. The New Site Group panel appears.
7. Enter the site group Name.
8. Select the Sites that should be in the site group. You can click and drag the sites from the left pane into the Sites in Group window. Site groups can initially be created without member sites where you add sites to the group later.
9. Identify the Shared Data that will be available to all sites within the site group.
10. Click Create to create the site group.

Adding a Shareable to the Site Group

A registered shareable can only be added once to a site group. Once a registered shareable has been added to a site group, it is available only to members of the site group.

If a site group does not contain any sites, any shareable can be added to the site group. If the site group does contain sites, the sites in the site group cannot be in any existing site groups that contain this shareable.

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.

4. Enter the name of a project and click Continue.
5. Select the site group to edit.
6. Click the + icon from the Actions menu and select Edit.

   The Site Group panel appears.
7. Select the shared data that will be available to all sites within the site group.
8. Click Save to save the site group.

Adding a Site to a Site Group

When you add a site to a site group, the site cannot already belong to the site group as only one instance of a site can exist in a site group. If the site group contains shareables, the site cannot belong to any other site groups that contain the same shareables.

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Select the site group to edit.
6. Click the + icon from the Actions menu and select Edit.

   The Site Group panel appears.
7. Select the sites that will be part of the site group by clicking and dragging the site into the pane, or by clicking the Select button and selecting the sites.
8. Click Save to save the site group.

Removing a Shareable from the Site Group

When a shareable is removed from a site group, none of the sites within the group will share the shareable.

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Select the site group to edit.
6. Click the + button from the Actions menu and select Edit.

   The Site Group panel appears.
7. Uncheck the shared data that you no longer want to be available to all sites within the site group.
8. Click Save to save the site group.

Removing a Site from the Site Group

When a site is removed from a site group, it no longer will share the site group shareables until added to another group containing the shareables.

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Select the site group to edit
6. Click the + icon from the Actions menu and select Edit.

   The Site Group panel appears.
7. Click the Sites in Group property and then click the Remove Site icon in the Sites in Group panel. The site will be removed from the site group.
8. Click Save to save the site group.

Deleting a Site Group

A site group can only be deleted if there are no site members. Once the site group is deleted, it will be removed from Site Administration.

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Select the site group to delete.

   Note: Before the site group can be deleted, all sites must be removed from the group.
6. Click the + button from the Actions menu and select Delete.

   A confirmation message appears asking if you want to delete the site group.
7. Click OK to delete the site group.

Implementing Site Categories

When you create a new site in Site Administration, it must be based on a site category. Site categories provide a set of metadata about the type of site you are creating. Site categories also provide a list of the configurable elements that can be used when creating new sites. When you set the values for these configurable elements, you create a site configuration. Because one site category can be configured to have many different values, it may have many associated site configurations.

It is important to work with the site developers to identify any configurable properties that should be available for site categories. When the site developers are creating and deploying the Web application,

they define specific functionality as being configurable. They define the configuration and available values for the functionally, and group these configurations into the site category. When you create a site category, you can provide the information or select the values for these configurations.

Site categories are comprised of an ID, a name, a description and an Item Mapping name. The Item Mapping name is created by the site developer and provides user interface mapping information for the site category. For additional information on view mapping and site category configuration, refer to Configuring Site Categories section.

Reviewing a Site Category

You can open an existing site category to review the configuration:

1. Log into the BCC and select Site Administration.
2. Click the Manage Site Categories link.
3. Select a site category to display. The configuration appears in the editing pane.

Creating a Site Category

When creating site categories, you must know the Item Mapping information created by the site developer. Additionally, site category names must be unique or the site category will not be created. To create a site category, perform the following from the BCC:

1. Log into the BCC and select Site Administration.
2. Click the Manage Site Categories link.
3. Use the + dropdown menu under the Action menu to create a new site category.
4. The system will automatically provide a site category ID. Enter a unique name, the description, and the Item Mapping information.

   The `itemMappingId` should be entered using the format: `SiteCategory:`*`category name`*
5. Click Create to create the site category. If the name is not unique, an error message will display indicating that the category cannot be created.

Editing a Site Category

When you edit a site category, you cannot edit the site category ID, which is provided by the system. If you change the name of the site category to a name that is already in use, an error will display indicating that the site category cannot be saved. You must provide a unique name before the edits that you have made can be saved.

To edit a site category, perform the following from the BCC:

1. Log into the BCC and select Site Administration.
2. Click the Manage Site Categories link.
3. Highlight the site category to edit.

4. Use the Action button to select Edit from the menu drop down. Begin editing the site category.
5. You can save the changes you made to the site category by clicking Save. Clicking Revert will erase your changes and keep the site category in its previous version.

Deleting a Site Category

Site categories cannot be deleted if they are associated with a site. Ensure that the site category is not being used by a site before deleting it:

1. Log into the BCC and select Site Administration.
2. Click the Manage Site Categories link.
3. Highlight the site category to delete.
4. Use the Action button to select Delete from the menu drop down. If the site category is still in use, you will receive a warning message that it cannot be deleted.

Applying Site Category Updates

Whenever site categories are updated, you must work with the site developer to ensure that the changes do not result in a loss of information.

Action	Existing Site	New Site
Adding a property	Site administrator updates the sites as necessary.	New sites will automatically include the property.
Deleting a property	Site developer deletes property from site configurations based on site category.	New sites will not include the deleted property.
Adding a value	Site administrator updates the sites as necessary.	New sites will automatically include the value.
Deleting a value	The old value will remain until the site administrator selects the property associated with this value. The old value will be deleted from the site configuration once the new value has been selected.	New sites will not include the value.

Previewing a Site

If you have installed a preview server, the Preview button is available in Site Administration. Previewing a site consists of previewing the site homepage and navigating to other pages from the homepage. When sites are created or modified, you can review them in Preview mode, which shows what would be presented to a given user on a specific page of a particular website.

Your site developers must create a preview URL properties file and identify the `name` value for each page that can serve as a homepage for your application. For information on creating the URL properties file, refer to the Configuring URLs section.

Note: It is important that site administrators work with site developers to identify which preview URLs map to the home page for the preview site.

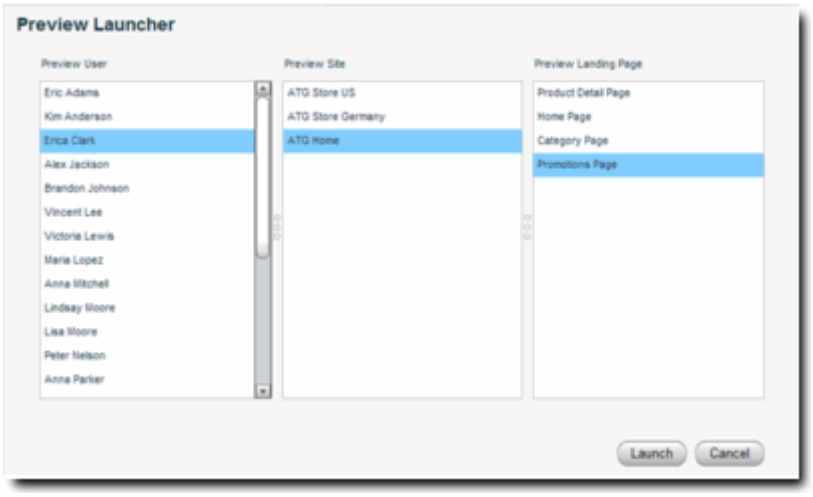
For information on configuring a preview server, refer to the *ATG Business Control Center Administration and Development Guide*. For information on using the preview server in the BCC, refer to the *ATG Business Control Center User's Guide*.

The ATG preview functionality requires that you select three parameters:

- Preview User - The preview is based on a user and will display the site and landing page as it will look like for this user. The users who can be selected for the Preview User parameter are defined by your site developer
- Preview Site – Displays the sites that can be previewed
- Preview Landing Page - This is the JSP to preview. Pages that are capable of being previewed have an associated preview URL Properties file that contains preview information

To preview a site:

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Select a site to preview.
6. Click the Preview As button and select the user, site and landing page from the dropdown menus.
7. Click Launch to preview the site.



Deploying a Site

Deploying a site includes deployment of the site configurations that have been configured in Site Administration. When a site configuration is applied to the corresponding Web application, a new site is created and deployed. You accomplish this by working with your network administrators to provide the URLs for the new site and the site category, allowing them to configure the Web application and deploy the sites.

There are three phases to deploying a site:

1. Save to Publishing - This occurs when the site information is recorded and saved in Site Administration.
2. Deploy to Staging (optional) – When you upload site information to your staging environment in accordance with CA deployment procedures. At this stage, the deployed site can be accessed by internal users only. For information on deploying to a staging server, refer to the *ATG Content Administration Guide for Business Users.*
3. Deploy to Production – When you upload site information to your runtime environment, the site can be accessed by external users. Refer to the documentation for your Web application for information about deploying to a production server.

Note: It is recommended that you deploy sites to your staging and production environments well before the site's launch date, or deploy the site in the `disabled` state to ensure that the site is complete and not accessed externally until all content has been verified and deployed.

Enabling a Site

Once you have deployed a site to your production environment, you can enable the site. By enabling a site, you are allowing external users to access the site. To enable a site:

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Select a site to enable.
6. Click the action button and select Enable from the dropdown menu.

You will be presented with a message that the site has been enabled. If the site has already been enabled, you will be presented with a message that the site is already enabled.

Disabling a Site

Once you have deployed a site to your production environment, you can disable the site so that external users cannot access the site. When you disable a site, it can be accessed by internal applications.

1. Open the BCC.
2. Select Site Administration.
3. Select the Manage Site Assets link.
4. Enter the name of a project and click Continue.
5. Select a site to disable.
6. Click the Action button and select Disable from the dropdown menu.

You will be presented with a message that the site has been disabled. If the site has already been disabled, you will be presented with a message that the site is already disabled.

Appendix A: Tables

Tables Used in the Site Repository

The following tables are used in the `SiteRepository`.

site_template

Primary key: site_template1_p

Property	Column	Nullable	Type	Length
`id`	`id`	no	varchar	40
`name`	`name`	no	varchar	254
`description`	`description`	no	varchar	254
`itemMappingID`	`item_mapping_id`	yes	varchar	40

site_configuration

Primary key: `site_configurat1_p`

Property	Column	Nullable	Type	Length
`id`	`Id`	no	varchar	40
`name`	`name`	no	varchar	254
`description`	`description`	no	varchar	254
`siteTemplate`	`template`	yes	varchar	40
`productionURL`	`production_url`	yes	varchar	254
`enabled`	`enable`	no	one-digit	
`siteDownURL`	`site_down_url`	yes	varchar	254

Property	Column	Nullable	Type	Length
openDate	open_date	yes	datetime	
preOpeningURL	pre_opening_url	yes	varchar	254
closingDate	closing_date	yes	datetime	
postClosingURL	post_closing_url	yes	varchar	254
modificationTime	modification_time	yes	datetime	
creationTime	creation_time	yes	datetime	
author	author	yes	varchar	254
lastModifiedBy	last_modified_by	yes	varchar	254
siteIcon	site_icon	yes	varchar	254
favicon	favicon	yes	varchar	254
sitePriority	site_priority	yes	integer	
contextRoot	context_root	yes	varchar	254

site_additional_urls

Primary Key: siteadditio_url1_p

Property	Column	Nullable	Type	Length
additionalProductionURLs	additional_production_url	yes	varchar	254

site_group

Primary Key: site_group1_p

Property	Column	Nullable	Type	Length
id	id	no	varchar	254
displayName	display_name	yes	varchar	254

site_group_sites

Primary Key: site_group_sites_p

Property	Column	Nullable	Type	Length
site	site_id	no	varchar	40

site_group_shareable_types

Primary Key: site_group_share_p

Property	Column	Nullable	Type	Length
shareableTypes	shareable_types	no	varchar (enum string)	254

site_type

Primary Key: site_type_p

Property	Column	Nullable	Type	Length
siteType	site_type	no	varchar (enum string)	254

Index

A

C

S

T

U

V

W