BEA WebLogic Portal®

Cache Reference

Version 9.2
Revised: July 2006

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

WebLogic Portal Cache Reference

WebLogic Portal Cache Reference

This reference guide lists the available caches for WebLogic Portal that can be managed within the Portal Administration Console.

Caches referenced in this guide can be modified within the Administration Console. Although some caches are not pre-configured within the Administration Console. You can add these caches to the Administration Console.

This book includes the following sections:

Adding a Cache

If you want to use a cache that is not in the list of configured caches, you must add the cache to the Portal Administration Console.

To add a cache:

1. Choose **Configuration Settings > Service Administration**.
2. Select the Cache Manager node in the tree.
3. In the Browse tab, click **Add Cache**.
4. Enter the name of the cache.
5. Optionally, enter or modify the default cache configuration settings.
6. Click **Update**.

The cache you have added appears in the list of caches.

- Portal Framework Caches
- WSRP Caches
- Content and Ad Caches
- User Management Caches
- Campaign and Discount Caches
- Commerce Caches

Portal Framework Caches

Table 1 portalContentUriCache

Cache	portalContentUriCache
Use	Used to store portal content URIs for a combination of webapp, portal, locale and optional user name.
Key	Key is equal to portal path + name of web application.
Value	Portal content URI
Notes	Set this cache according the number of portals that have associated content URIs. The default values are recommended. Default values: MaxEntries=500; TimeToLive=-1

Note: The portalContentUriCache is not included in the Administration Console. If you want to manage this cache, you need to add it manually, see "Adding a Cache" on page -1.

Table 2 portalLocalizationLocaleCache

Cache	portalLocalizationLocaleCache
Use	Used to store collection of `LocalizationLocale` objects. Localization locale specifies language, character encoding, country and variant.
Key	The key is private static final String called `portalLocalizationLocaleCachekey`.
Value	A set of LocalizationLocale objects.
Notes	Default TTL value should be okay. Max Entries could be set to a number based on the number of rows in the L10N_LOCALE table, i.e. number of supported locales. Default values: MaxEntries=500; TimeToLive=-1

Table 3 portletControlTreeCache

Cache	portletControlTreeCache
Use	Used to store portlet control trees for floating portlets.

Table 3 portletControlTreeCache (Continued)

Key	The combination portletInstanceId and locale.
Value	A portlet control tree.
Notes	Default TTL value should be okay, MaxEntries could be set to a number based on number of floatable portlet instances in a portal (including user customized portlets) and number of supported locales. It is recommended that the TTL be left at -1 because the cached default desktop needs to be kept in the cache indefinitely and the cached item for a logged in user is removed when they log out so there is no need to expire a user's cached items. To avoid having the LRU mechanism kick the cached default desktop out of the cache, the MaxEntries should be set to at least (max # of concurrent logged in users + 1) X (# of locales supported). If the cache is too small then LRU will kick out the cached default desktop and the memory saving advantage of this approach will be lost. Default values: MaxEntries=500; TimeToLive=-1

Note: This portletControlTreeCache is not included in the Administration Console. If you want to manage this cache, you need to add it manually, see "Adding a Cache.".

Table 4 portletPreferencesCache

Cache	portletPreferencesCache
Use	Used to store portlet preferences.
Key	An instance of PortletPreferenceId.
Value	A map of preferences.
Notes	Default TTL and Max Entries values could be set to a value depending on amount of available memory and total number of preferences (at the application level). Defaults: MaxEntries = 500, TimeToLive=60000 (one minute)

Note: This portletPreferencesCache is not included in the Administration Console. If you want to manage this cache, you need to add it manually, see "Adding a Cache.".

Table 5 portalLocalizationResourceCache

Cache	portalLocalizationResourceCache
Use	Used to store localization resources.
Key	The localizationIntersection.
Value	A LocalizationResource.
Notes	Default TTL and MaxEntries values could be set to a value based on total number of localization resources in the system, which is a combination of non-customized and customized localization resources, and the amount of available memory. Default values: MaxEntries=500; TimeToLive=-1

Note: The portalLocalizationResourceCache is not included in the Administration Console. If you want to manage this cache, you need to add it manually, see "Adding a Cache.".

Table 6 portalControlTreeCache

Cache	portalControlTreeCache
Use	Used to store portal control trees. Only used for streaming portals.
Key	The combination of webapp, portal, desktop, locale and optional user name.
Value	A portal control tree.
Notes	Default TTL value should be okay. This cache will contain one entry for the default portal, plus one entry for each user who has customized his or her portal. Max Entries could be set to a number based on number of users and available memory. If there are any changes to portal this cache will be flushed. Default values: MaxEntries=500; TimeToLive=-1

Table 7 portalMarkupDefinitionCache

Cache	portalMarkupDefinitionCache
Use	Used to store `MarkupDefinition` objects.

Table 7 portalMarkupDefinitionCache (Continued)

Key	A MarkupDefintionID.
Value	A MarkupDefinition.
Notes	Set this according to the number of rows in the PF_MARKUP_Definition. Markup is the blueprint for a portal library resource (desktop, book, page, portlet, placeholder, menu, Look And Feel, layout, shell or theme). Default values: MaxEntries=500; TimeToLive=60000 (one minute).

Note: This cache is not included in the Administration Console. If you want to manage this cache, you need to add it manually, see "Adding a Cache.".

WSRP Caches

Table 8 remoteProducerInfoCache

Cache	remoteProducerInfoCache
Use	Caches the metadata for producers added to a consumer application.
Key	Name of the consumer web application.
Value	A java.util.HashMap containing producer metadata. This map is keyed with the producerHandle of each producer.
Notes	This cache is used to look for producer metadata when a user or administrator is trying to interact with a remote portlet or a producer. Default values: MaxEntries=500; TimeToLive=-1

Note: The remoteProducerInfoCache is not included in the Administration Console. If you want to manage this cache, you need to add it manually, see "Adding a Cache.".

Table 9 registrationHandleCache

Cache	registrationHandleCache
Use	Used to store registration handles of all registered consumers, for all producers.
Key	The `registrationHandle` of the consumer.
Value	A java.lang.boolean object with a value of true/false.
Notes	This cache is used to cache whether or not a particular `registrationHandle` is valid. Default values: MaxEntries=500;TimeToLive=-1.

Note: The registrationHandleCache is not included in the Administration Console. If you want to manage this cache, you need to add it manually, see "Adding a Cache.".

Table 10 proxyPortletCache

Cache	proxyPortletCache
Use	This caches the ProxyPortlets by proxyportletId.
Key	String representing the portlet instance id.
Value	Information from the consumer registry and about the proxy portlet instance (`com.bea.wsrp.services.persistence.internal.ProxyPortletInfoInternal.ProxyPortletInfoInternalobject`).
Notes	Default values: MaxEntries: 100; TimeToLive = -1

Content and Ad Caches

Table 11 binaryCache.*repository_name*

Cache	binaryCache.*repository_name*
Use	Used to store binary property values for a repository node.

Table 11 binaryCache.*repository_name* (Continued)

Key	String (node ID + Property ID)
Value	A byte array associated with the binary property.
Notes	Set this according to the number and size of binary property values. Default values: MaxEntries: 10; TimeToLive:60000 (one minute)

Table 12 adServiceCache

Cache	adServiceCache
Use	Used to store the results of searches for content rendered in a placeholder (ads). Used by the AdHelper to increase the speed of ad queries.
Key	The ad query (java.lang.String)
Value	A Content []
Notes	Set this according to the number of ad queries and the amount of content expected to be retrieved. Consider basing the maximum size on the total number of ad queries. If the ads returned from a particular query do not change, consider increasing the TTL. Default values: MaxEntries=32; TimeToLive=300000 (five minutes)

Note: This adServiceCache is not included in the Administration Console. If you want to manage this cache, you need to add it manually, see "Adding a Cache.".

Table 13 nodePathCache.*repository_name*

Cache	nodePathCache.*repository_name*
Use	Used to store a list of nodes for a repository based on a path.
Key	A String (NodeID).

Table 13 nodePathCache.*repository_name (Continued)*

Value	A Node.
Notes	Set according to the number of nodes in a repository. Default values: MaxEntries=50; TimeToLive=60000 (one minute)

Table 14 searchCache

Cache	searchCache
Use	Used to store an array of IDs for nodes that satisfy a content search.
Key	A Search, which contain parameters for a query.
Value	An ID array of nodes that satisfy a query.
Notes	There is only one search cache used for all repositories. Default values: MaxEntries=20; TimeToLive-=60000 (one minute) Set the MaxEntries according to the amount of content expected to be retrieved. Set Time To Live according to how fresh the content should be.

Table 15 nodeCache.*repository_name*

Cache	nodeCache.*repository_name*
Use	Used to store repository nodes. Each repository has its own cache setting.
Key	A String representing the node ID.
Value	A node.
Notes	Set this according to the number of nodes in a repository. Default values: MaxEntries=50; TimeToLive=6000 (one minute)

User Management Caches

Table 16 entityIdCache

Cache	entityIdCache
Use	Caches the IDfor an entity (user or group ID)
Key	A com.bea.p13n.property.PropertyLocator. PropertyLocator is based on a user or group name (ENTITY.ENTITY_NAME) and entity type (ENTITY.ENTITY_TYPE).
Value	The entity ID (java.lang.Long).
Notes	Use the ENTITY table as a guide for the maximum size. The object being stored is a Long, which is fairly small. Therefore, it might be possible to set this cache's maximum size to the number of entries in the ENTITY table. Consider how often the ENTITY table might change when setting the TTL. Default values: MaxEntries=500;TimeToLive=600000

Table 17 jndiNameCache

Cache	jndiNameCache
Use	Stores the JNDI names of entity property managers and UUP managers.
Key	An entity ID.
Value	The home name, which is a string value.
Notes	Set this according the combination of the number of entity property managers and the number of UUP managers. Default values: MaxEntries=500;TimeToLive=600000

Table 18 entityPropertyCache

Cache	entityPropertyCache
Use	Caches property values for users and groups.
Key	A `com.bea.p13n.property.PropertyLocator`. PropertyLocator is based on the user or group name (ENTITY.ENTITY_NAME), entity type (ENTITY.ENTITY_TYPE, user or group) and property set type (PROPERTY_KEY.PROPERTY_SET_TYPE, usually USER).
Value	A `com.bea.p13n.property.EntityPropertyCache` object. This object contains a Map that stores property values keyed off the property set name and property name.
Notes	The larger you can afford to make this cache, the better. Use the ENTITY table as a guide for maximum size. The number of entries in this table should be the maximum number of cache entries that would ever be created. In most cases, there will be more entries here than you would want for a maximum cache size. So consider the average number of users you expect to be using your application at the same time. Consider a TTL based on how often new properties will be added to the property sets. If they are not being modified often, then a higher TTL might be appropriate. Default values: MaxEntries=500;TimeToLive=600000

Table 19 profileTypeCache

Cache	profileTypeCache
Use	Caches user profile types that are used to look up the appropriate user manager profile manager when retrieving a user profile.
Key	A String (the user name).
Value	A String (the profile type).
Notes	This should be set based on the number of concurrent users. Set the TimeToLive never to expire Default values: MaxEntries=100;TimeToLive=3600000

Note: This profileTypeCache is not included in the Administration Console. If you want to manage this cache, you need to add it manually, see "Adding a Cache.".

Table 20 propertyKeyIdCache

Cache	propertyKeyIdCache
Use	Caches the unique ID associated with a property set type, property set and property name combination (primary key in the PROPERTY_KEY database table).
Key	Based on a property set type, property set, and property name combination (inner class called `PropertyKeyLocator`).
Value	The ID (java.lang.Long)
Notes	Maximum size should be set with an eye towards the maximum number of properties in the application (use the PROPERTY_KEY table as an indicator). Consider a TTL based on how often these unique ID combinations are likely to change. Default value: MaxEntries=500;TimeToLive=600000

Campaign and Discount Caches

Table 21 globalDiscountCache

Cache	globalDiscountCache
Use	Stores computed global discount definitions. This is the set of global discounts that is applicable to all users.
Key	The globalDiscountSet name (java.lang.String)
Value	The java.util.Set of `qualificationDiscountDef` objects.
Notes	Set this to the number of global discounts in your application. The frequency of changes to the global discounts should determine TTL. Default values: MaxEntries=10; TimeToLive-=300000 (five minutes)

Table 22 discountCache

Cache	discountCache
Use	Used to store computed discount definitions (applicable to individual customers or to customer segments).
Key	A QualificationDiscountId. This is essentially a wrapping around a java.lang.Integer that represents the ID of a discount.
Value	The java.util.Set of `qualificationDiscountDef` objects
Notes	Set this to the number of discounts in your application. Frequency of changes to the global discounts should determine TTL. Default values: MaxEntries=100; TimeToLive-=300000 (five minutes)

Commerce Caches

Table 23 globalDiscountAssocCache

Cache	globalDiscountAssocCache
Use	Stores computed global discount associations. This is the set of discount associations that is applicable to all users.
Key	CustomerPk, which is a unique identifier for a customer (java.lang.String).
Value	A `DiscountAssociation` object. A discount association is the mapping of a Customer to a Discount. It is used to track and limit how many times the discount is used by a particular customer.
Notes	Default values: MaxEntries=100; TimeToLive-=3600000 (one hour). Set `MaxEntries` to the number of global discount associations in your application. The frequency of changes to the global discount associations should determine TTL. To use this cache, you must start the Weblogic server using the command line option: `-Denable.discount.assoc.caches=true` This enables caching of discount associations and global discount associations. The default is `false`, which means a separate read is performed for every pricing calculation for every line item in every shopping cart. Enabling the cache reduces database load by storing associations in a cache after the first read. You can use the Service Administration tools within the Portal Administration Console to manage this cache. First you must add the cache so that it is visible in the Portal Administration Console, as described in the online help for the Portal Administration Console. You can also manage the cache using the p13n cache API.

Table 24 discountAssocCache

Cache	discountAssocCache
Use	Stores computed discount associations (applicable to individual customers or to customer segments).
Key	CustomerPk, which is a unique identifier for a customer (java.lang.String).

Table 24 discountAssocCache (Continued)

Value	A DiscountAssociation object. A discount association is the mapping of a Customer to a Discount. It is used to track and limit how many times the discount is used by a particular customer.
Notes	Default values: MaxEntries=100; TimeToLive-=3600000 (one hour). Set `MaxEntries` to the number of discount associations in your application. The frequency of changes to the discount associations should determine TTL. To use this cache, you must start the Weblogic server using the command line option: `-Denable.discount.assoc.caches=true` This enables caching of discount associations and global discount associations. The default is `false`, which means a separate read is performed for every pricing calculation for every line item in every shopping cart. Enabling the cache reduces database load by storing associations in a cache after the first read. You can use the Service Administration tools within the Portal Administration Console to manage this cache. First you must add the cache so that it is visible in the Portal Administration Console, as described in the online help for the Portal Administration Console. You can also manage the cache using the p13n cache API.

Table 25 CategoryCache

Cache	**categoryCache**
Use	Stores the root `com.beasys.commerce.ebusiness.catalog`.Category, the total number of categories in the product catalog (java.lang.Integer) and the CategoryInfo for each category. `CategoryManagerImpl` gets the cache name from the `ejb-jar.xml` in `commerce.jar`
Key	The key for the root Category is a static final String variable in the `CategoryManagerImpl` class. The key for the total number of categories is also a static final String variable in the `CategoryManagerImpl` class. The key for a given `CategoryInfo` object is a `com.beasys.commerce.ebusiness.catalog.CategoryKey`.

Table 25 CategoryCache (Continued)

Value	The value for the root Category is `com.beasys.commerce.ebusiness.catalog.Category`. The value for the total number of categories is a java.lang.Integer. The value for the category info objects is a `com.beasys.commerce.ebusiness.catalog.service.category.CategoryInfo`.
Notes	The root Category and the total number of categories occupy two slots in the cache and the remaining slots are occupied by the `CategoryInfo` objects, so consider the total number of categories in the product catalog plus 2 when setting the maximum cache size. Consider how often these categories will change when setting TTL. Default values: MaxEntries:1000;TimeToLive: 8640000

Table 26 ProductItemCache

Cache	ProductItemCache (ProductItemManagerImpl gets the cache name from the `ejb-jar.xml` in `commerce.jar`.)
Use	Stores the total number of product items in the catalog as well as the product items
Key	The key for the total number of product items is a static final String variable in `ProductItemManagerImpl`. The key for the product items is a `com.beasys.commerce.ebusiness.catalog.ProductItemKey`.
Value	The value for the total number of product items is a java.lang.Integer. The value for the product item is a `com.beasys.commerce.ebusiness.catalog.ProductItem`.
Notes	Consider the total number of product items when setting the maximum cache size. Consider how often these product items will change when setting the TTL. Default values: MaxEntries=1000;TimeToLive=21600000