BEA WebLogic Portal®

Cache Reference

Version 10.2
Revised: February 2008

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 & Monitoring > Service Administration**.

2. Select **Cache Manager** in the Resource 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 added appears in the list of caches.

Weblogic Portal Framework Caches

Table 1 through Table 18 detail information on WebLogic Portal framework caches.

Table 1 CategoryTreeCache

Cache	CategoryTreeCache
Use	Holds portlet category trees
Key	Webapp name
Value	CategoryTree objects
Notes	

Table 2 communitiesEntityPropertyCache

Cache	communitiesEntityPropertyCache
Use	Holds community membership capability information for users accessing communities
Key	A combination of community definition ID and the user name
Value	A map of community membership capabilities
Notes	This cache optimizes access to community membership properties for members of a community. Base the cache size on the expected number of users and the expected number of communities that each user would normally access within the same time period. Misses to this cache generally result in one database call per request where the miss occurred.

Table 3 communitiesMemberActiveCache

Cache	communitiesMemberActiveCache
Use	Caches information about active status for community members
Key	Username
Value	A String representing the users' community member record active status
Notes	This information is used with status for individual community memberships to determine overall active status. Size this cache proportionately to the number of community users that you expect to be logged in at the same time. Misses to this cache generally result in one database call per request where the miss occurred.

Table 4 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.

Table 5 portalLocalizationLocaleCache

Cache	portalLocalizationLocaleCache
Use	Used to store collection of LocalizationLocale objects (LocalizationLocale specifies language, character encoding, country, and variant)
Key	The key is private static final String called portalLocalizationLocaleCachekey.

Table 5 portalLocalizationLocaleCache (Continued)

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 6 portletControlTreeCache

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

Table 7 PortletCategoryCache

Cache	PortletCategoryCache
Use	Holds portlet category objects
Key	PortletCategoryDefinitionId

Table 7 PortletCategoryCache (Continued)

Value	PortletCategoryDefinition objects
Notes	

Table 8 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).

Table 9 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

Table 10 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 11 portalLayoutDefinitionCache

Cache	portalLayoutDefinitionCache
Use	Holds LayoutDefinition objects
Key	LayoutDefinitionId
Value	LayoutDefinition objects
Notes	

Table 12 portalMarkupdefinitionCache

Cache	portalMarkupdefinitionCache
Use	Used to store MarkupDefinition objects
Key	A MarkupDefintionID

Table 12 portalMarkupdefinitionCache (Continued)

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).

Table 13 portalThemeDefinitionCache

Cache	portalThemeDefinitionCache
Use	Holds ThemeDefinition objects
Key	ThemeDefinitionId
Value	ThemeDefinition objects
Notes	

Table 14 netuix.community.definition.cache

Cache	netuix.community.definition.cache
Use	Holds community definitions
Key	A combination of webapp name, portal path, and desktop path for a community
Value	CommunityDefinition objects
Notes	

Table 15 netuix.community.id.to.path.cache

Cache	netuix.community.id.to.path.cache
Use	Maps community definition IDs to the communities' webapp names, desktop path, and portal path
Key	A CommunityDefinitionId
Value	The communities' webapp names, desktop path, and portal path
Notes	

Table 16 netuix.notification.global

Cache	netuix.notification.global
Use	Holds notifications targeted to a user, but not targeted to an individual web application
Key	Username
Value	ArrayList of Notification objects
Notes	

Table 17 wlp.urlCompression.compressed

Cache	wlp.urlCompression.compressed
Use	Maps compressed URL IDs to the expanded URL
Key	The numeric compressed URL ID
Value	The expanded URL
Notes	

Table 18 wlp.urlCompression.expanded

Cache	wlp.urlCompression.expanded
Use	Maps expanded URLs into compressed URL IDs
Key	Expanded (full) URLs
Value	Compressed URL ID
Notes	

WSRP Caches

Table 19 through Table 24 detail information on WSRP caches.

Table 19 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 20 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 or 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 21 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.

Table 22 proxyPortletRendeDependenciesCache

Cache	proxyPortletRendeDependencies Cache
Use	This caches render dependencies obtained from remote producers

Table 22 proxyPortletRendeDependenciesCache (Continued)

Key	RenderDependencyCacheKey (the proxy portlet's Unique ID) containing: • Versioned app name • Webapp name • Producer handle • WSDL URL • Namespace prefix
Value	Array containing: • List<IRenderDependencyTag> • List<IScriptFragment>
Notes	Used when the cacheRenderDependencies property (in the .portlet file) is true and the portlet is a WSRP proxy portlet. Max entries: 500; TTL: Unlimited.

Table 23 complexProducerPortletHandleToIdCache

Cache	complexProducerPortletHandleToIdCache
Use	The complex producer (WSRP) uses this cache to look up the portlet's primary instance ID
Key	The remote portlet's handle
Value	The remote portlet's primary instance ID
Notes	Should be sized to fit the number of remote portlets in concurrent use. Default values: Size: 1000; TTL: 1 hour.

Table 24 complexProducerPortletIdToDefinitionLabel

Cache	complexProducerPortletIdToDefinitionLabel
Use	The complex producer (WSRP) uses this cache to look up the portlet's definition label
Key	The remote portlet's ID

Table 24 complexProducerPortletIdToDefinitionLabel

Value	The remote portlet's definition label
Notes	Should be sized to fit the number of remote portlets in concurrent use. Default values: Size: 1000; TTL: 1 hour.

Content and Ad Caches

Table 25 through Table 38 detail information for content and ad caches.

Table 25 binaryCache.*repository_name*

Cache	binaryCache.*repository_name*
Use	Used to store binary property values for a repository node
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 26 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).

Table 27 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 (Node path)
Value	A node
Notes	Set according to the number of nodes in a repository. Default values: MaxEntries=50; TimeToLive=60000 (one minute).

Table 28 nodeCache.*repository_name*

Cache	nodeCache.*repository_name*
Use	Used to store a list of nodes for a repository based on an ID
Key	An ID (NodeID)
Value	A node
Notes	Set according to the number of nodes in a repository. Default values: MaxEntries=50; TimeToLive=60000 (one minute).

Table 29 repositoryConfigCache

Cache	repositoryConfigCache
Use	VCR cache; caches repository configuration information
Key	(String) repository name
Value	RepositoryConfig object associated with that repository name
Notes	

Table 30 repo.explicitPropertyCache

Cache	repo.explicitPropertyCache
Use	BEA repository cache; caches explicit property information for all BEA repositories
Key	(String) repository name
Value	Collection of repository property definition information for explicit properties in that BEA repository.
Notes	

Table 31 repo.nodeIdCache.*repository_name*

Cache	repo.nodeIdCache.*repository_name*
Use	BEA repository cache; caches node information for a specific BEA repository instance
Key	Node ID
Value	Repository node data
Notes	

Table 32 repo.nodePathCache.*repository_name*

Cache	repo.nodePathCache.*repository_name*
Use	BEA repository cache; caches node information for a specific BEA repository instance
Key	Node path
Value	Repository node data
Notes	

Table 33 repo.typeBinaryCache.*repository_name*

Cache	repo.typeBinaryCache.*repository_name*
Use	BEA repository cache; caches node binary property information for a specific BEA repository instance
Key	Node UID + binary property UID
Value	Byte[]
Notes	

Table 34 repo.typeIdCache.*repository_name*

Cache	repo.typeIdCache.*repository_name*
Use	BEA repository cache; caches node type information for a specific BEA repository instance
Key	Type (objectclass) ID
Value	Repository type data
Notes	

Table 35 repo.typeNameCache.*repository_name*

Cache	repo.typeNameCache.*repository_name*
Use	BEA repository cache; caches node type information for a specific BEA repository instance
Key	Type (objectclass) name
Value	Repository type data
Notes	

Table 36 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 you want the content.

Table 37 typeCache.*repository_name*

Cache	typeCache.*repository_name*
Use	VCR cache, caches Type (ObjectClass) information
Key	ObjectClass ID
Value	ObjectClass object
Notes	

Table 38 typeNameCache.*repository_name*

Cache	typeNameCache.*repository_name*
Use	VCR cache, caches Type (ObjectClass) Name --> TypeID mapping
Key	ObjectClass Name
Value	ObjectClass ID
Notes	

User Management Caches

Table 39 through Table 44 detail information on user management caches.

Table 39 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 40 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 41 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 42 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.

Table 43 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.

Table 44 credentialEntryCache

Cache	credentialEntryCache
Use	Caches credential vault entries with encrypted credential
Key	com.bea.p13n.security.management.credentials.internal.CredentialEntryLocator
Value	com.bea.p13n.security.management.credentials. CredentialEntry.
Notes	Default values: Max Entries=100; Time To Live=1 hour.

Campaign and Discount Caches

Table 45 through Table 48 detail information on campaign and discount caches.

Table 45 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 46 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).

Table 47 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.
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 48 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.

Table 48 ProductItemCache (Continued)

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.