BEA WebLogic Portal™

Performance Tuning Guide

Version 8.1 with Service Pack 3
Document Revised: July 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

General Performance Tuning Guidelines

Tuning Your Portal Domain

Tuning Your Portal Application

WebLogic Portal Cache Settings

CHAPTER 1

General Performance Tuning Guidelines

Application performance is affected by many factors. This chapter discusses a few of the initial aspects that can affect performance and provides links to documentation resources that can assist you.

- WebLogic Server Tuning
- Database Tuning
- Upgrade to Latest Service Packs

WebLogic Server Tuning

Because WebLogic Portal runs on WebLogic Server, it is expected that factors impacting the performance of WebLogic Server will also impact the performance of WebLogic Portal.

For more information about tuning your WebLogic Server, see http://edocs.bea.com/wls/docs81/perform/index.html

Database Tuning

WebLogic Portal provides an optimization script that runs the respective vendor database script that computes database statistics needed for the database optimizer. Typically, this script is called statistics.sql and is stored in the respective BEA install for each supported database in the respective directory.

The statistics.sql script should be run periodically, as well as after the database content significantly changes (e.g., a large number of users are added). Keeping database statistics up to date can dramatically improve query execution time. See

http://edocs.bea.com/wlp/docs81/db/index.html for more information.

Upgrade to Latest Service Packs

Service packs almost always include improvements to some area of performance. Service packs are available individually for download to Contract Support Customers. Go to the http://support.bea.com to log in to eSupport. Navigate to Product Download and Service packs in the left navigation bar. Choose the product of interest and follow links to the version and service pack you are interested in.

CHAPTER 2

Tuning Your Portal Domain

Optimally, when you deploy, you need to create a new domain that is configured for your production environment, including clusters, production configuration settings, etc.

However, if you have deployed a development domain and wish to use it for production, you must change your domain environment settings to optimize performance.

Note: It is not recommended to use a development domain for production, see http://edocs.bea.com/platform/docs81/confgwiz/newdom.html#1059076 for more information.

This topic discusses the following:

- Tuning Your Domain Configuration
- Removing Debugging Tools from Your Domain

Tuning Your Domain Configuration

The domain settings are managed by the setDomainEnv.cmd (or setDomainEnv.sh) script which is found in your domain directory. By default, the script is found in: <bea-home>/user_projects/<domain name>/setDomainEnv.cmd or *.sh.

To edit this file, open it in a text editor.

The following is a table of the start script settings and their appropriate values for a production domain. Remember if you are using a domain that was created for production mode, you do not need to modify the configuration.

Flag Name	Production Mode Setting	Notes
`WLS_PRODUCTION_MODE`	`true`	• Indicates whether you are in a production mode or a development mode. Default is false for domains created in development mode and true for domains created in production mode.
`iterativeDevFlag`	`false`	• Disable this option to prevent checking for changed Workshop files, and if found, rebuilding and redeploying the application. Default is true for domains created in development mode and false for domains created in production mode.
`debugFlag=""`	`false`	• Used in start scripts to set debugging options and indicate if the WebLogic Workshop Debugger should be started. When switched to false, you save the resource overhead used for debugging. • Default is `debugFlag=true` for domains created in development mode; `debugFlag=false` for domains created in production mode.
`testConsoleFlag=""`	`false`	• Verify by checking the log for: `"wlw. testConsole = false".` • Enables the JWS test view. • Default is true for domains created in development mode; false for domains created in production mode.

`logErrorsToConsoleFlag=""`	`false`	• Verify by checking the log for: `wlw.logErrorsToConsole = false` • Saves you additional logging although, the tradeoff is that you may see exceptions more easily when this is set to True (without checking the log). • Default is true for domains created in development mode and false for domains created in production mode.
`verboseLoggingFlag=""`	`false`	• If true, override the default `LOG4J_CONFIG_FILE` (`workshopLogCfg.xml`) with `workshopLogCfgVerbose.xml` . • Priority value in the default file is "warn"; in the verbose version it's "debug". • Verify by checking the log for: "`log4j.configuration = … "workshopLogCfg.xml`" instead of "`workshopLogCfgVerbose.xml`" • You can also start in verbose mode using "`startWebLogic.cmd verbose`". • Saves you debugging overhead. • Default is false for domains created in development mode and false for domains created in production mode.
`pointbaseFlag=""`	`false`	• Indicates whether Pointbase should be started. • Verify by checking for a running Pointbase process. • Saves you the resource overhead of starting Pointbase when it's not needed. • Default is true for domains created with Pointbase as the database.

Removing Debugging Tools from Your Domain

When deploying a domain, you should remove the debug.properties file from the domain directory. Although this file is helpful during development, debugging should not be done in production environments.

CHAPTER 3

Tuning Your Portal Application

One of the most effective ways to tune your portal application is to monitor and adjust the cache settings used.

Managing Caches

WebLogic Portal provides a single framework for configuring, accessing, monitoring, and maintaining caches. If configured properly, the caches can vastly reduce the time needed to retrieve frequently used data.

Many WebLogic Portal services use preconfigured caches that you can tune to meet your performance needs. Some services use internally configured caches that you cannot configure or access. If you extend or create additional services, you can use the cache framework to define and use your own set of caches.

WebLogic Portal Cache Settings lists caches that might be used by your portal application. Use the list to assist in your tuning. Keep in mind the memory that is available to your system. When modifying the maximum cache sizes also monitor the system memory to determine the effects.

Using the Portal Administration Tool to Configure Cache Settings

You can use the Service Administration tools within the Portal Administration Tool to configure statically-defined caches. For a list of configurable caches, see Appendix , "WebLogic Portal Cache Settings."

When you configure a cache, you modify its parameters to change its behavior or capability. For example, you can set up a cache to hold only the last 10,000 entries and set the time they can remain in the cache. You can also flush the cache so that all new requests for information come directly from the database.

For instructions on how to configure cache settings, see

http://edocs.bea.com/wlp/docs81/adminportal/help/SA_CacheConfig.html

Caching with JSP Tags

When you chose to configure caches on individual JSP tags, you can have more control over individual content queries. Although this can be seen as an advantage, remember that when you control caches with coding, any cache change will require more maintenance, depending on the size (amount of code) of your application.

The following content management-related JSP tags include cache-related attributes:

- <cm:search>
- <cm:getNode>
- <pz:contentSelector>
- <pz:contentQuery>

For more information about these JSP tags and their attributes, see
http://edocs.bea.com/workshop/docs81/doc/en/portal/taglib/JspWlpOverview.html

WebLogic Portal Cache Settings

This appendix lists the caches used to tune performance in WebLogic Portal 8.1. Find a cache you need to adjust, then use the WebLogic Administration Portal to adjust the cache settings.

Portal Framework Caches

Table 0-1 portalContentUriCache

Cache	portalContentUriCache
Use	This caches 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.

Table 0-1 portalContentUriCache

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 0-2 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 0-3 portletControlTreeCache

Cache	portletControlTreeCache
Use	Used to store portlet control trees for floating portlets.
Key	The combination portletInstanceId and locale.

Table 0-3 portletControlTreeCache

Value	A portlet control tree.
Notes	Default TTL value should be okay, Max Entries 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 0-4 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)

Table 0-5 portalLocalizationResourceCache

Cache	portalLocalizationResourceCache
Use	Used to store localization resources.
Key	The localizationIntersection.

Table 0-5 portalLocalizationResourceCache

Value	A LocalizationResource.
Notes	Default TTL and Max Entries 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 0-6 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 0-7 portalMarkupDefinitionCache

Cache	portalMarkupDefinitionCache
Use	Used to store MarkupDefinition objects.
Key	A MarkupDefintionID.

Table 0-7 portalMarkupDefinitionCache

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).

WSRP Caches

Table 0-8 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

Table 0-9 registrationHandleCache

Cache	registrationHandleCache
Use	Used to store registrationHandles of all registered consumers, for all producers.
Key	The registrationHandle of the consumer.

Table 0-9 registrationHandleCache

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.

Content and Ad Caches

Table 0-10 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 0-11 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 0-12 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).
Value	A Node.
Notes	Set according to the number of nodes in a repository. Default values: MaxEntries=50; TimeToLive=60000 (one minute)

Table 0-13 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)

Table 0-14 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 0-15 entityIdCache

Cache	entityIdCache
Use	Caches the id for 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 0-16 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 0-17 entityPropertyCache

Cache	entityPropertyCache
Use	Caches property values for users and groups.

Table 0-17 entityPropertyCache

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 0-18 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: If profileTypeCache is not available, you will need to create it. You can do so by editing the respective application-config.xml file for the associated application. Within the application's application-config.xml file, rename 'unifiedProfileTypeCache' to

'profileTypeCache'. The application-config.xml file is located in the respective application's //META-INF directory. For example:

```
//<BEA_HOME>/USER_PROJECTS/<APPLICATION_NAME>/META-INF/application-config.
xml
```

Table 0-19 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 0-20 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 0-21 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 0-22 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 0-22 categoryCache

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 0-23 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