

How to cache WebSphere Commerce pages with the WebSphere Application Server dynamic cache service

Authors:

Cecilia Chan is a staff software developer at the IBM Toronto Lab, Ontario, Canada. She is a lead developer of the Cache component of WebSphere Commerce.

James Tang is a staff software developer at the IBM Toronto Lab, Ontario, Canada. He is a lead developer of the Cache component of WebSphere Commerce.

Barbara Wong is a advisory software developer at the IBM Toronto Lab, Ontario, Canada. She is the team lead of the Cache component of WebSphere Commerce.

Date : May, 2004
Version 1.0

1.1 Overview	5
2.1 Introduction	6
3.1 Configuring the dynamic cache service	8
3.1.1 Enable the dynamic cache service	8
3.1.2 Define cache-entry elements	8
3.1.3 Define cache ID generation rules	11
3.1.1 Define dependency ID generation rules	12
3.1.1 Define invalidation ID generation rules	14
4.1 Displaying cache information	16
4.1.1 Dynamic cache monitor	16
4.1.2 DynaCacheEsi application	17
4.1.3 Performance monitoring infrastructure (PMI)	21
4.1.3.1 Tivoli Performance Viewer	22
5.1 Caching strategy	27
5.1.1 Which pages should be cached	27
5.1.2 Cache whole pages or page fragments	27
5.1.3 Cache whole page excluding certain fragments	30
5.1.4 Where caching should take place	32
5.1.5 How to invalidate the cached data	33
6.1 Implementing Caching Strategy	34
6.1.1 JSP include mechanisms	34
6.1.2 Session dependent and session independent	34
6.1.2.1 Cache filter	35
7.1 Edge Side Include (ESI) caching	39
7.1.1 Caching static data	39
7.1.2 Caching full pages using ESI	39
7.1.3 Caching fragments using ESI	40
8.1 External cache groups	44
8.1.1 Edge components Caching Proxy	44
8.1.1.1 External cache configuration at content host	44
8.1.1.2 Dynaedge-cfg.xml file modifications	45
9.1 Tutorial	47
9.1.1 B2BDirect business model	47
9.1.1.1 Publishing a store archive	47
9.1.2 Defining cache policy	49
9.1.2.1 Caching whole pages	50
9.1.2.2 Caching page fragments	50
9.1.2.3 Defining cache IDs	50
9.1.2.4 Defining dependency IDs	51
9.1.2.5 Store relationship considerations	52

9.1.3 Caching the ToolTech store catalog pages	54
9.1.3.1 Caching as whole page or page fragments	55
9.1.3.2 Defining cache policies for the JSPs	58
9.1.3.3 Marketing campaign	61
9.1.3.4 Rewriting JSPs for caching	61
9.1.3.5 Defining cache entry and cache ID	63
9.1.3.6 Defining dependency ID	65
9.1.4 Displaying cache information	66
10.1 Cache invalidation	81
10.1.1 Rule-based	81
10.1.2 Servlet-based invalidation	81
10.1.3 Command-based invalidation	81
10.1.4 Time-based invalidation	82
10.1.5 Group-based invalidation	83
10.1.6 Programmatic invalidation	84
10.1.6.1 Invalidation through the cache monitor	85
10.1.6.2 DynaCacheInvalidation command	85
10.1.6.3 CACHEIVL table	85
10.1.6.4 Database Triggers	85
10.1.7 Building invalidation policies in WebSphere Commerce	86
10.1.7.1 Invalidation rules in cachespec.xml	86
10.1.7.2 Servlet-based versus command-based invalidation	86
10.1.7.3 Building dependency IDs for invalidation	87
10.1.8 Cache invalidation verification	92
11.1 Conclusion	96
11.1.1 What should be cached	96
11.1.2 Where caching should take place	96
11.1.3 How to invalidate the cached data	96
11.1.3.1 Sort out what events causes the fragment change	97
11.1.3.2 Decide which invalidation vehicle to use	98
11.1.3.3 Put together the invalidation policies	99
11.1.3.4 Measure the effectiveness of the invalidation scheme	102
12.1 Simple file servlet	103
12.1.1 Selective caching	104
13.1 Problem determination	105
13.1.1 WebSphere Application Server trace	105
13.1.2 WebSphere Commerce cache trace	105
13.1.3 Trace strings	105
13.1.4 Web server plug-in trace	106
13.1.5 Trace output for Application Server	106

14.1	Sample JSPs	107
14.1.1	Original StoreCatalogDisplay.jsp	107
14.1.2	Original HeaderDisplay.jsp	111
14.1.3	Original SidebarDisplay.jsp	114
14.1.4	StoreCatalogDisplay.jsp	116
14.1.5	CachedParametersSetup.jsp	119
14.1.6	CachedStoreCatgalogDisplay.jsp	121
14.1.7	StoreCatalogProductEspot.jsp	123
14.1.8	CachedHeaderDisplay.jsp	125
14.1.9	CachedSidebarDisplay.jsp	127
15.1	Sample of WebSphere Commerce Cache Trace	128
	Appendix A	134
	Appendix B	136
	Notices	137

1.1 Overview

In general, caching improves response time and reduces system load. Caching techniques have long been used to improve the performance of World Wide Web Internet applications. Most techniques cache static content (content that rarely changes) such as graphic and text files. However, many Web sites serve dynamic content, containing personalized information or data that changes more frequently. Caching dynamic content requires more sophisticated caching techniques, such as those provided by the WebSphere Application Server 5.0 dynamic cache, a built-in service for caching and serving dynamic content.

Web sites built using WebSphere Commerce 5.5 serve dynamic content. The purpose of this white paper is to demonstrate how to cache WebSphere Commerce dynamically generated content, using the dynamic cache service. A hands on approach describing how to implement dynamic caching with WebSphere Commerce is presented, using examples taken from one of the sample stores (B2BDirect - ToolTech) provided with the product.

For background and related information, refer to the following documents:

1. *WebSphere Application Server Information Center*, available at the following URL:

<http://www.ibm.com/software/webservers/appserv/infocenter.html>

Refer to the topic entitled "*Improving performance through the dynamic cache service*", which can be found in the WebSphere Application Server Version 5 Information Center under "*Applications*", and then under "*Application Services*".

2. *WebSphere Commerce Administration Guide*, a user guide available at the following URL:

<http://www.ibm.com/software/genservers/commerce/wcbe/library/lit-tech-general.html>

Refer to the section entitled "*Dynamic caching*".

3. *WebSphere Commerce Store Development Guide*, a user guide also available at the above URL. Refer to the topic entitled "*Caching your store pages*".

2.1 Introduction

WebSphere Application Server 5.0 offers a built-in dynamic cache service for serving dynamic content and caching data. The dynamic cache service improves application performance by caching outputs produced by specified servlets, JavaServer Pages™ (JSP), Web services, and commands. The dynamic cache service intercepts calls to those executable entities and responds by serving the cached outputs as specified.

The dynamic cache service includes :

- *Servlet/JSP result cache*, to cache whole pages or fragments generated by a Servlet or JSP.
- *Command cache*, to cache command objects. To be cacheable, command objects must define an interface that extends the `CacheableCommand` Java™ interface.
- *Edge Side Include (ESI) caching*, to cache, assemble and deliver dynamic web pages at the edge of an enterprise network.
- *Invalidation Support*, to ensure the content of the cache is correct. Invalidation can be rule-based, time-based, group-based, and programmatic.
- *Replication support*, to enable cache sharing and replication among multiple servers.
- *Disk offload capability*, to enable caching large amounts of data, and to preserve cache content while the application server is stopped and restarted.

The caching behavior of the WebSphere Application Server dynamic cache service is specified by cache policies defined by `<cache-entry>` elements in cache specification configuration XML (“`cachespec.xml`”) files.

As the dynamic cache service places objects in the cache, it labels them with unique identifying strings (*cache IDs*) constructed according to `<cache-id>` rules specified in the `<cache-entry>` elements. Once an object with a particular *cache-id* is in the cache, a subsequent request for an object with the same *cache-id* is served from the cache (a cache “hit”). The `<cache-id>` rules define how to construct cache-ids from information associated with an application server *request* (to execute a Servlet, JSP, or command), including how information may be obtained programmatically from `CacheableCommand` objects.

Cached objects are removed from the cache according to information provided in their `<cache-entry>` elements, such as the `<timeout>`, `<priority>` and `<invalidation>` elements.

The `<timeout>` and `<priority>` elements configure expiry and eviction policies. When the available cache memory is full, a least recently used (*LRU*) caching algorithm removes cached objects with lower priority, or offloads them to disk if the disk offload capability is enabled, before those with higher priority.

The `<dependency-id>` and `<invalidation>` elements define rules that generate *dependency IDs* and *invalidation IDs*, which together specify that certain objects should be removed from the cache when certain requests (such as those that update cached

information) are processed. When an object is cached, its generated dependency IDs are associated with it in the cache. When a request causes invalidation IDs to be generated, all objects associated with those invalidation IDs are removed from the cache.

The dynamic cache service responds to changes in the `cachespec.xml` file. When new versions of the file are detected, the old policies are replaced. Objects cached through the old policy file are not automatically invalidated from the cache; they are either reused with the new policy or eliminated from the cache through its replacement algorithm.

For complete detailed documentation of the `cachespec.xml` file please refer to the topic "*cachespec.xml file*" in the *WebSphere Application Server Information Center*, available at the following URL:

<http://www.ibm.com/software/webservers/appserv/infocenter.html>.

3.1 Configuring the dynamic cache service

The dynamic cache service is an in-memory cache system that has disk offload capability. Cacheable objects are defined the `cachespec.xml` file found inside the Web application archive (WAR), Web module `WEB-INF` directory, or enterprise bean `WEB-INF` directory. A global `cachespec.xml` file can also be placed in the application server properties directory, but the recommended method is to place the cache configuration file with the deployment module.

There is a sample cache configuration file (`cachespec.sample.xml`) located in the `WASInstallDir/properties` directory. The `cachespec.dtd` file is available in the application server properties directory.

The following sections detail the steps required to configure dynamic caching in the `cachespec.xml` file.

3.1.1 Enable the dynamic cache service

Although the dynamic cache service is enabled by default, servlet caching needs to be explicitly enabled. For details of how to enable the dynamic cache service and, for example, servlet caching, please refer to *"Enabling globally the dynamic cache service"* and *"Configuring servlet caching"* in the *WebSphere Application Server Information Center*, available at the following URL:

<http://www.ibm.com/software/webservers/appserv/infocenter.html>

3.1.2 Define cache-entry elements

The root element of the `cachespec.xml` file, `<cache>`, contains `<cache-entry>` elements. The WebSphere dynamic cache service parses the `cachespec.xml` file during startup, and extracts a set of configuration parameters from each `<cache-entry>` element.

This section explains the function of the following `<cache-entry>` elements in the `cachespec.xml` file used by WebSphere Commerce:

- `<class>`
- `<name>`
- `<property>`

The following sections explain other `<cache-entry>` elements such as `<timeout>`, and `<priority>` that configure information such as cache expiry and cache eviction policy, and elements that specify how to generate identifiers such as the `<cache-id>`, `<dependency-id>`, and `<invalidation>` elements used to configure invalidation of cached objects.

Class

```
<class>servlet | command</class>
```


The `<class>` element is required. It governs how the application server interprets the remaining cache policy definition. The value `servlet` indicates that the cache policy defines caching rules for a servlet or a JavaServer Page™ (JSP) deployed in the WebSphere Application Server servlet engine. The value `command` indicates that the cache policy defines caching rules for a class that implements the WebSphere `CacheableCommand` Java interface.

The following are examples of the `<class>` element:

```
<class>command</class>
<class>servlet</class>
```

Note: While WebSphere Commerce Version 5.5 commands implement the `CacheableCommand` Java interface, almost all of them also extend from the `AbstractECTargetableCommand` Java class. Because that class does not implement the `Serializable` Java interface, these commands cannot be cached. They can, however, be used to generate cache invalidation IDs, as specified by the `<invalidation>` element. Please refer to the [*Cache invalidation methodologies*](#) section for details.

Name

```
<name>classname</name>
```

The `<name>` element is required. `classname` specifies the fully qualified class name of the command or servlet. There are two ways to use the `<name>` element to specify a cacheable object:

- For commands, `classname` must specify the fully qualified classname, including the package name, if any, and class name, as well as the trailing `.class` qualifier, of the configured object.
- For servlets or JSPs, this element must include the fully qualified class name of the servlet, or the full URI of the JSP.

You can specify multiple `<name>` elements within a `<cache-entry>` if you have different mappings that refer to the same Servlet.

The following are examples of the `<name>` element:

```
<name>com.ibm.commerce.dynacache.commands.MemberGroupsCacheCmd
Impl.class</name>
<name>com.ibm.commerce.server.RequestServlet.class</name>
<name>/ToolTech/ShoppingArea/CatalogSection/CategorySubsection
/StoreCatalogDisplay.jsp</name>
```

Property

```
<property name="key">value</property>
```

where `key` is the name of the property being defined, and `value` is the corresponding value.

The following table contains properties used by WebSphere Commerce when defining cache rules for servlets. For a complete list of properties, please refer to the topic "*cachespec.xml file*" in the *WebSphere Application Server Information Center*, available at the following URL:

<http://www.ibm.com/software/webservers/appserv/infocenter.html>

Property	Valid Values	Default Value	Description
EdgeCacheable	true false	false	If this property is <code>true</code> , the given servlet or JSP file is externally available from an edge server. Whether or not the servlet or JSP file is cacheable, depends on the rest of the cache specification.
consume-subfragments	true false	false	When a servlet is cached, only the content of that servlet is stored, with placeholders for any other fragments it includes or to which it forwards. <code>Consume-subfragments</code> (CSF) tells the cache not to stop saving content when it includes a child servlet. The parent entry (the one marked CSF) will include all the content from all fragments in its cache entry, resulting in one large cache entry that has no includes or forwards, but rather the content from the whole tree of entries. This can save a significant amount of application server processing, but is typically only useful when the external HTTP request contains all the information needed to determine the entire tree of included fragments.
save-attributes	true false	true	When this property is set to <code>false</code> , the request attributes are not saved with the cache entry.
store-cookies	true false	true	When this property is set to <code>false</code> , the request cookies are not saved with the cache entry.
alternate_url			Specifies the alternate URL used to invoke the servlet or JSP file. This property is ignored unless the

			EdgeCacheable property is also set for the cache entry.
do-not-consume (Only available with WAS 5.0.2 + PQ81651 or later releases)	true false	false	When this property is set to true, the fragment will not be consumed by the parent entry with the consume-subfragments property.

The following are examples of the `<property>` element:

```
<property name="consume-subfragments">true</property>
<property name="save-attributes">false</property>
<property name="store-cookies">false</property>
```

Note: By default, the dynamic cache service caches all cookies and request attributes (when caching results from a servlet or JSP execution) along with the cached results. However, WebSphere Commerce cookies and request attributes contain user specific information which should not be cached.

Therefore, the following `<property>` elements must be specified for a servlet `<cache-entry>` element:

```
<property name="save-attributes">false</property>
<property name="store-cookies">false</property>
```

And the following `<property>` element must be specified for a JSP `<cache-entry>` element:

```
<property name="save-attributes">false</property>
```

3.1.3 Define cache ID generation rules

As the dynamic cache service places objects in the cache, it labels them with unique identifying strings (*cache IDs*) constructed according to `<cache-id>` rules specified in the `<cache-entry>` elements. Once an object with a particular cache ID is in the cache, a subsequent request for an object with the same cache ID is served from the cache (a cache “hit”). The `<cache-id>` rules define how to construct cache-IDs from information associated with an application server request, including how information may be obtained programmatically from `CacheableCommand` objects.

Each `<cache-entry>` can specify multiple `<cache-id>` elements, to define multiple cache ID generation rules. They are executed in the order they appear in the `<cache-entry>` element until either:

- a rule generates a valid cache ID, or
- all the rules have been executed.

If none of the cache ID generation rules produce a valid cache ID, then the object is not cached.

Each `<cache-id>` element defines a rule for caching an object and is composed of the sub-elements `<component>`, `<timeout>`, `<priority>` and `<property>`.

The `<component>` sub-element can appear many times within the `<cache-id>` element. Each time it specifies how to generate a component of a cache ID. There are several different types of component elements, such as parameter, session, attribute, locale, method, and field.

The `<timeout>`, `<priority>`, and `<property>` sub-elements can be used to control cache entry expiry, cache eviction policy, and other generic properties for a cached object with an identifier generated by its enclosing `<cache-id>` element.

For more information about the `<cache-id>` element, and its sub-elements, please refer to the topic "*cachespec.xml file*" in the *WebSphere Application Server Information Center*, available at the following URL:

<http://www.ibm.com/software/webservers/appserv/infocenter.html>

The following `<cache-entry>` example uses a `<cache-id>` element to cache results created by a JSP and generate a cache ID with two components, "storeId" and "catalogId", obtained from parameters in the request object:

```
<cache-entry>
  <class>servlet</class>
  <name>/ToolTech/. . ./StoreCatalogDisplay.jsp</name>
  <property name="save-attributes">false</property>
  <property name="store-cookies">false</property>
  <timeout>3600</timeout>
  <priority>3</priority>
  <cache-id>
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
  </cache-id>
  ...
</cache-entry>
```

Refer to the *Defining cache policy* section for more examples.

3.1.1 Define dependency ID generation rules

Dependency IDs are additional cache group identifiers that associate multiple cache entries to the same group identifier and are used in conjunction with invalidation IDs (described in the next section) to remove invalidated objects from the cache.

Cached objects can also be invalidated explicitly by using the WebSphere dynamic cache API . This API is not discussed further in this paper. For more information, refer to the Javadoc API information for the *invalidateById* and *invalidateByTemplate* methods in the *com.ibm.websphere.cache.Cache* interface from the *WebSphere Application Server Information Center* available at the following URL:

<http://publib.boulder.ibm.com/infocenter/wasinfo/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/javadoc/ae/index.html>

Dependency IDs are generated according to rules specified by `<dependency-id>` elements within `<cache-entry>` elements. When an object is placed in the cache, its generated dependency IDs are associated with it in the cache.

Each `<cache-entry>` element may contain several `<dependency-id>` elements. Each `<dependency-id>` element causes a dependency ID to be generated when a request that matches its `<cache-entry>` element is received.

Each dependency ID is generated by concatenating its dependency ID base string with the values, each prefixed with a colon (": "), returned by its `<component>` sub-elements. If a required component returns a null value, then the entire dependency ID is not generated and is not used. Multiple dependency ID rules can exist for each `<cache-entry>` and dependency ID rules are executed separately.

The following `<cache-entry>` example uses two `<dependency-id>` elements to generate two dependency IDs, "storeId" and "catalogId", obtained from parameters in the request object:

```
<cache-entry>
  <class>Servlet</class>
  <name>/ToolTech/. . ./StoreCatalogDisplay.jsp</name>
  <property name="save-attributes">false</property>
  <property name="store-cookies">false</property>
  <timeout>3600</timeout>
  <priority>3</priority>

  <cache-id>
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
  </cache-id>

  <dependency-id>storeId
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>
```

```

    <dependency-id>catalogId
      <component id="catalogId" type="parameter">
        <required>true</required>
      </component>
    </dependency-id>
  </cache-entry>

```

In this example, when a request to execute the specified `StoreCatalogDisplay.jsp` is received with parameters `"storeId=10001&catalogId=10010"`, the generated dependency IDs are `"storeId:10001"` and `"catalogId:10010"`.

Refer to the [*Defining dependency IDs*](#) section for more examples.

3.1.1 Define invalidation ID generation rules

Invalidation IDs are used in conjunction with dependency IDs (described in the previous section) to remove invalidated objects from the cache.

Invalidation IDs are generated according to rules specified by `<invalidation>` elements within `<cache-entry>` elements. When a request matching a `<cache-entry>` element is received, its generated invalidation IDs are matched against the dependency IDs previously associated with cached objects. Cached objects with an associated dependency ID that is the same as one of the generated invalidation IDs are invalidated.

Each `<cache-entry>` element may contain several `<invalidation>` elements. Each `<invalidation>` element causes an invalidation ID to be generated when a request that matches its `<cache-entry>` element is received.

Each invalidation ID is generated by concatenating its invalidation ID base string with the values, each prefixed with a colon (": "), returned by its `<component>` sub-elements. If a required component returns a null value, then the generated invalidation ID it is part of is discarded.

The following `<cache-entry>` example defines an `<invalidation>` element to generate an invalidation IDs, `"catalogId"`, when the `CatalogUpdateCmdImpl` command is executed:

```

<cache-entry>
  <class>command</class>
  <name>com.ibm.commerce.catalogmanagement.commands.CatalogUpdateCmdImpl</name>
  <invalidation>catalogId
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>

```

In this example, when a request to execute the specified command `com.ibm.commerce.catalogmangement.commands.CatalogUpdateCmdImpl` is received with parameters `"storeId=10001&catalogId=10010"`, the generated invalidation ID is `"catalogId:10010"`. That would cause any cached objects associated with a dependency ID of that value to be invalidated, effectively removing them from the cache.

Refer to the [*Cache invalidation methodologies*](#) section for more examples.

4.1 Displaying cache information

WebSphere Application server provides two tools to monitor the cache :

1. The Cache Monitor, provided as an installable Enterprise Archive file that can be used to monitor the cache statistics and contents.
2. The Performance Monitoring Infrastructure (PMI), which can collect data to be viewed by the Tivoli Performance Viewer.

4.1.1 Dynamic cache monitor

The dynamic cache monitor is an installable Web application that displays simple cache statistics, cache entries, and cache policy information.

To install the cache monitor :

1. Start the application server *server1* .
2. Log on to the Administrative Console *http://hostname:portNumber/admin*.
3. Click **Applications > Install New Application**.
4. Click **Browse** and locate the *CacheMonitor.ear* in the *install_root/installableApps* directory.
5. Click **next** to accept the default settings.
6. For the *Map virtual hosts to web module* screen, select *VH_instanceName_admin* as the virtual host .
7. For *Map modules to application servers* screen, select the application server to be monitored.

Install New Application

Allows installation of Enterprise Applications and Module

[Step 1](#) Provide options to perform the installation

[Step 2](#) Map virtual hosts for web modules

→ Step 3 : Map modules to application servers

Specify the application server where you want to install modules contained in your application. Modules can be installed on the same server or dispersed among several servers.

Clusters and Servers:

<input type="checkbox"/>	Module	URI	Server
<input checked="" type="checkbox"/>	Dynamic Cache Monitor	CacheMonitor.war,WEB-INF/web.xml	WebSphere:cell=buzz,node=buzz,server=WC_demo

Previous

Next

Cancel

Figure 1. CacheMonitor.ear installation

8. Click **Environment > Update Web Server Plugin** to regenerate the Web server plugin configuration.
9. Restart your application server and Web server.
10. Access the Web application using a Web browser and the URL <https://hostname:8002/cachemonitor>.

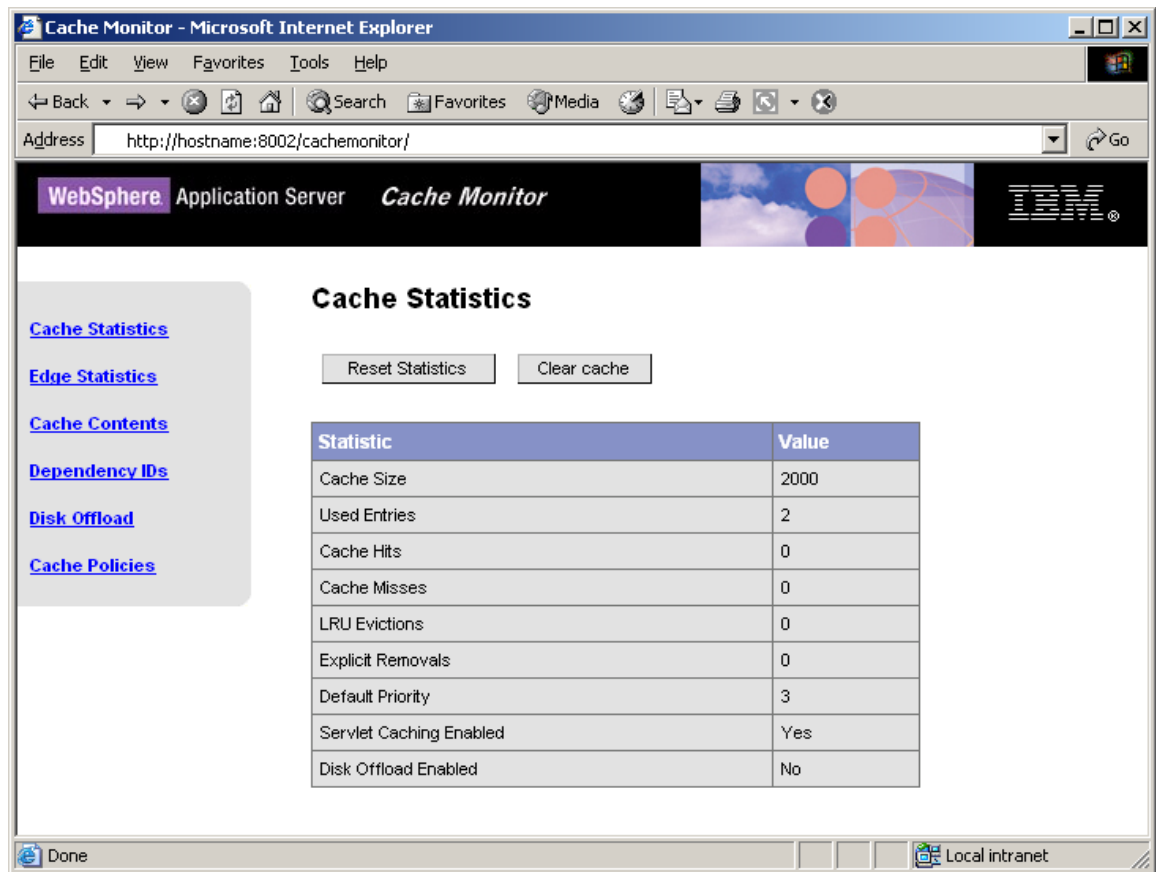


Figure 2. Cache monitor

Note: If you have enabled WebSphere Application Server EJB security, you will need to perform some additional steps as documented in the section *Security configuration for the dynamic cache monitor* in the WebSphere Commerce Security Guide if you want to use the dynamic cache monitor.

4.1.2 DynaCacheEsi application

The DynaCacheEsi.ear is an installable Web application that displays edge cache statistics.

How to install the DynaCacheEsi.ear:

1. Start the application server *server1*.
2. Log on to the Administrative Console <http://hostname:portNumber/admin>.
3. Click **Applications > Install New Application**.

4. Click **Browse** and locate the *DynaCacheEsi.ear* under the *install_root/installableApps* directory.
5. Click **next** to accept the default settings
6. For the *Map virtual hosts to web module* screen, select *VH_instanceName* as the virtual host .
7. For *Map modules to application servers* screen, select the application server to be monitored.

Install New Application

Allows installation of Enterprise Applications and Module

[Step 1](#) Provide options to perform the installation

[Step 2](#) Map virtual hosts for web modules

→ **Step 3 : Map modules to application servers**

Specify the application server where you want to install modules contained in your application. Modules can be installed on the same server or dispersed among several servers.

Clusters and Servers:

<input type="checkbox"/>	Module	URI	Server
<input type="checkbox"/>	DynaCacheEsi	DynaCacheEsi.war \WEB-INF\web.xml	WebSphere:cell=boost,node=boost,server=WC_demo

[Step 4](#) Summary

Figure 3. DynaCacheEsi.ear installation

8. Click **Environment > Update Web Server Plugin** to regenerate the Web Server plugin configuration.
9. Add `<Property Name="esiInvalidationMonitor" Value="true"/>` to *plugin-cfg.xml* file.
- 10.Restart your application server and Web server.
- 11.Access the Web application using a Web browser and the URL <https://hostname:8002/cachemonitor>.
- 12.Click on **Edge Statistics**.

Edge Cache Statistics

[Refresh Statistics](#) [Reset Statistics](#) [Clear cache](#)

Statistic	Total	Host/Process All/All
ESI Processors	1	1
Number of Edge Cached Entries	3	3
Cache Hits	0	0
Cache Misses By URL	14	14
Cache Misses By Cache ID	1	1
Cache Time Outs	0	0
Evictions	0	0

All Hosts ▼

All Processes ▼

[Statistics](#) [Contents](#)

Figure 4. Edge statistics as viewed from the Cache Monitor

Current Edge Cache Contents for Host: boost.torolab.ibm.com Process: 1960

Entries 1 through 4 of 4

< 0 >

[Refresh Contents](#) [Clear cache](#)

Cache ID	Host	Process
/webapp/wcs/stores/servlet/CategoryDisplay_storeId=10051:catalogId=10051:categoryId=10055	boost.torolab.ibm.com	1960
/webapp/wcs/stores/servlet/ProductDisplay_storeId=10051:productId=10360	boost.torolab.ibm.com	1960
/webapp/wcs/stores/servlet/ProductDisplay_storeId=10051:productId=10361	boost.torolab.ibm.com	1960
/webapp/wcs/stores/servlet/StoreCatalogDisplay_storeId=10051:catalogId=10051	boost.torolab.ibm.com	1960

Figure 5. Edge cache contents as viewed from the cache monitor

4.1.3 Performance monitoring infrastructure (PMI)

In order to monitor performance data through the PMI interfaces, you must first enable the performance monitoring services through the WebSphere Application Server Administrative Console.

Here are the steps to configure PMI:

1. Start the application server *server1*.
2. Log on to the Administrative Console *http://hostname:portNumber/admin*.
3. Click **Servers > Application Servers**.
4. Click *server*.
5. On the **Configuration tab > Performance Monitoring Service**.
6. Select the checkbox **Startup**.
7. Click **OK** and save the configuration.
8. Restart the application server.

[Application Servers > WC_demo >](#)

Performance Monitoring Service

Configuration and Runtime Settings for Performance Monitoring Infrastructure (PMI) ⓘ

Configuration

General Properties

Startup	<input checked="" type="checkbox"/>	ⓘ Specifies whether the server will attempt to start the specified service when the server starts.
Initial specification level	<input checked="" type="radio"/> None - All modules below set to "N" (None). <input type="radio"/> Standard - All modules below set to "H" (High) <input type="radio"/> Custom - Modify, add or remove the modules from the below list.	ⓘ A Performance Monitoring Infrastructure (PMI) specification string that stores PMI specification levels for all components in the server. Levels N,L,M,H,X represent None,Low,Medium,High,Maximum respectively.

beanModule=N
cacheModule=N
connectionPoolModule=N
j2cModule=N
jvmRuntimeModule=N

Apply OK Reset Cancel

Additional Properties

[Custom Properties](#) Additional custom properties for this service which may be configurable.

Figure 6. PMI configuration

The PMI module for dynamic cache service is called the *cacheModule* and it contains the data counters for memory cache size, number of invalidations, number of cache hits and cache misses, etc. For details of the dynamic cache data counters, please refer to the topic *performance data organization* in the *WebSphere Application Server Information Center*, available at the following URL:

<http://www.ibm.com/software/webervers/appserv/infocenter.html>

4.1.3.1 Tivoli Performance Viewer

The Tivoli Performance Viewer is a performance monitor shipped with WebSphere Application Server 5.0. The viewer retrieves performance data by periodically polling the administrative server. Data is collected continuously and retrieved as needed within the viewer.

When using the Tivoli Performance Viewer for WebSphere Commerce, you need to specify the SOAP or RMI port number for the *WC_instanceName* application server in order to measure the WebSphere Commerce related counters. One way to determine the *WC_instanceName* application server's SOAP or RMI port number is to look at the *SystemOut.log* file located in the *WAS_installDir/logs/WC_instanceName* directory. Whenever the server starts up, the SOAP or RMI port number will be displayed.

Example:

```
[6/2/03 7:12:46:141 EDT] 1aadc081 JMXSoapAdapte A ADMC0013I: SOAP connector  
available at port 8881  
[6/2/03 7:14:59:750 EDT] 1aadc081 RMICConnectorC A ADMC0026I: RMI Connector  
available at port 9810
```

Tivoli Performance Viewer can be started from the command line. Go to the *WAS_installDir/bin* directory and run the *tperfvviewer* script.

Windows:

tperfvviewer.bat host_name port_number connector_type

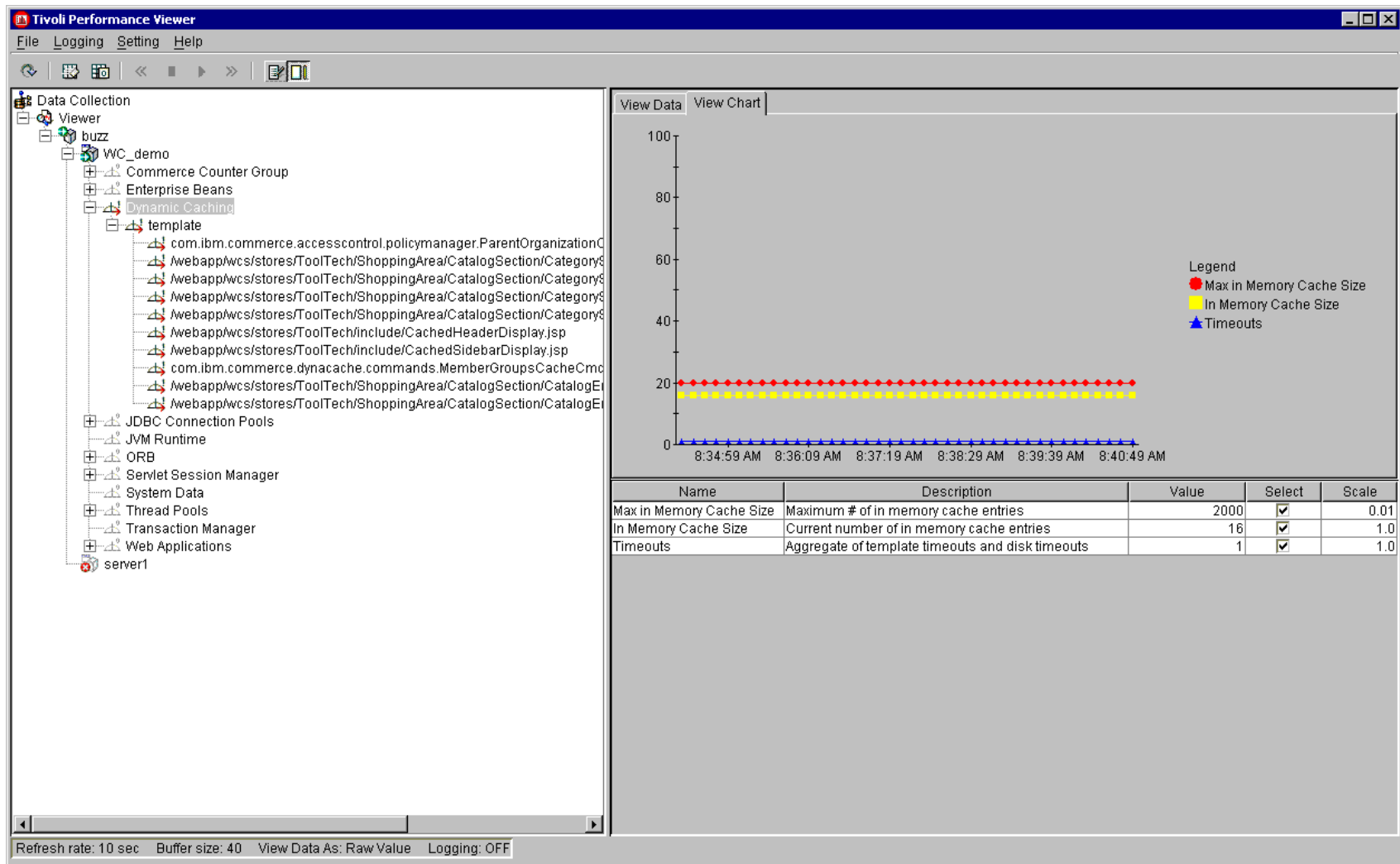
Unix:

tperfvviewer.sh host_name port_number connector_type

For example:

tperfvviewer.bat localhost 8881 SOAP

Figures 7 to 9 show the Tivoli Performance Viewer.



I.
Figure 7. Tivoli Performance Viewer - dynamic caching

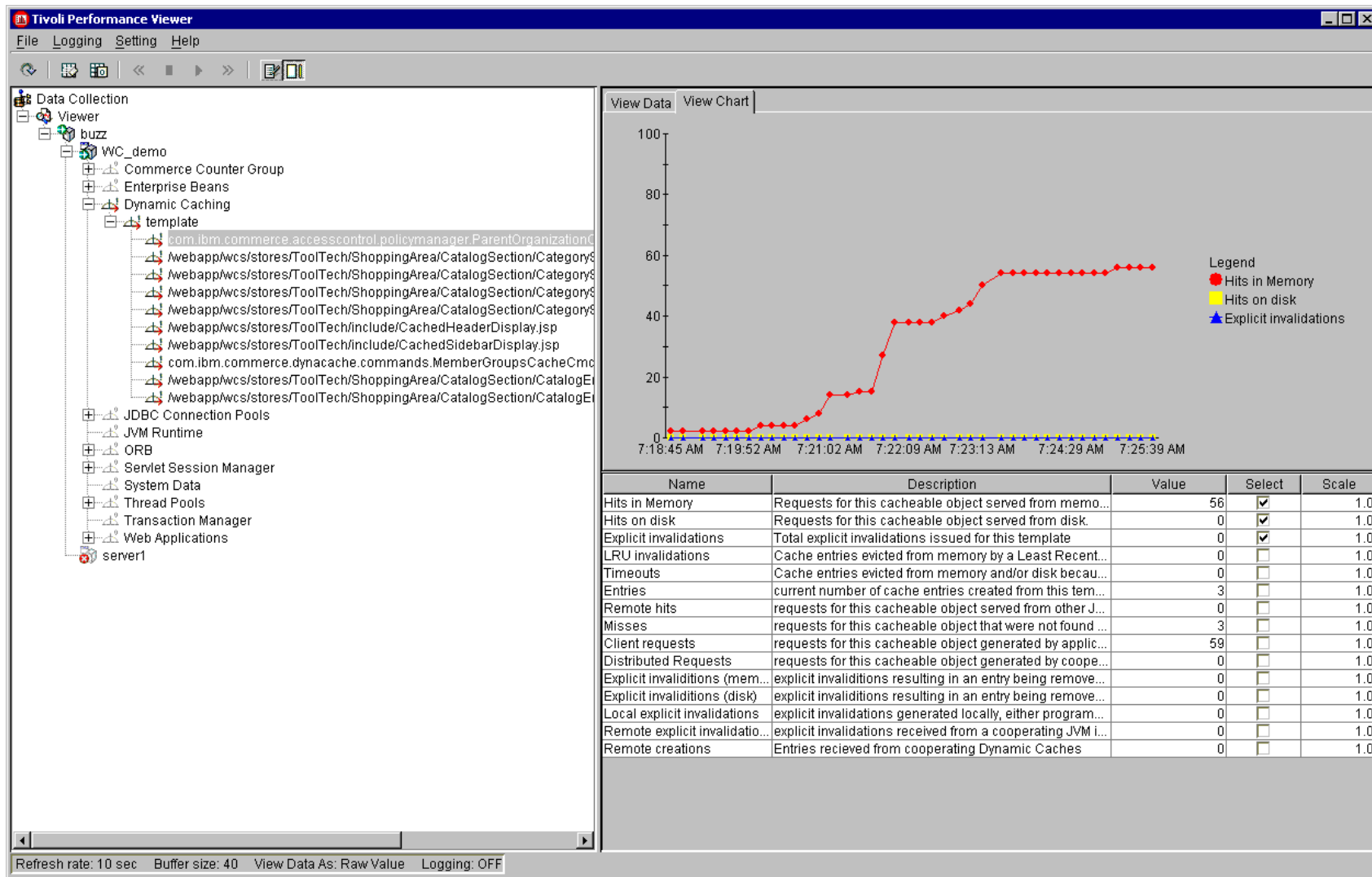
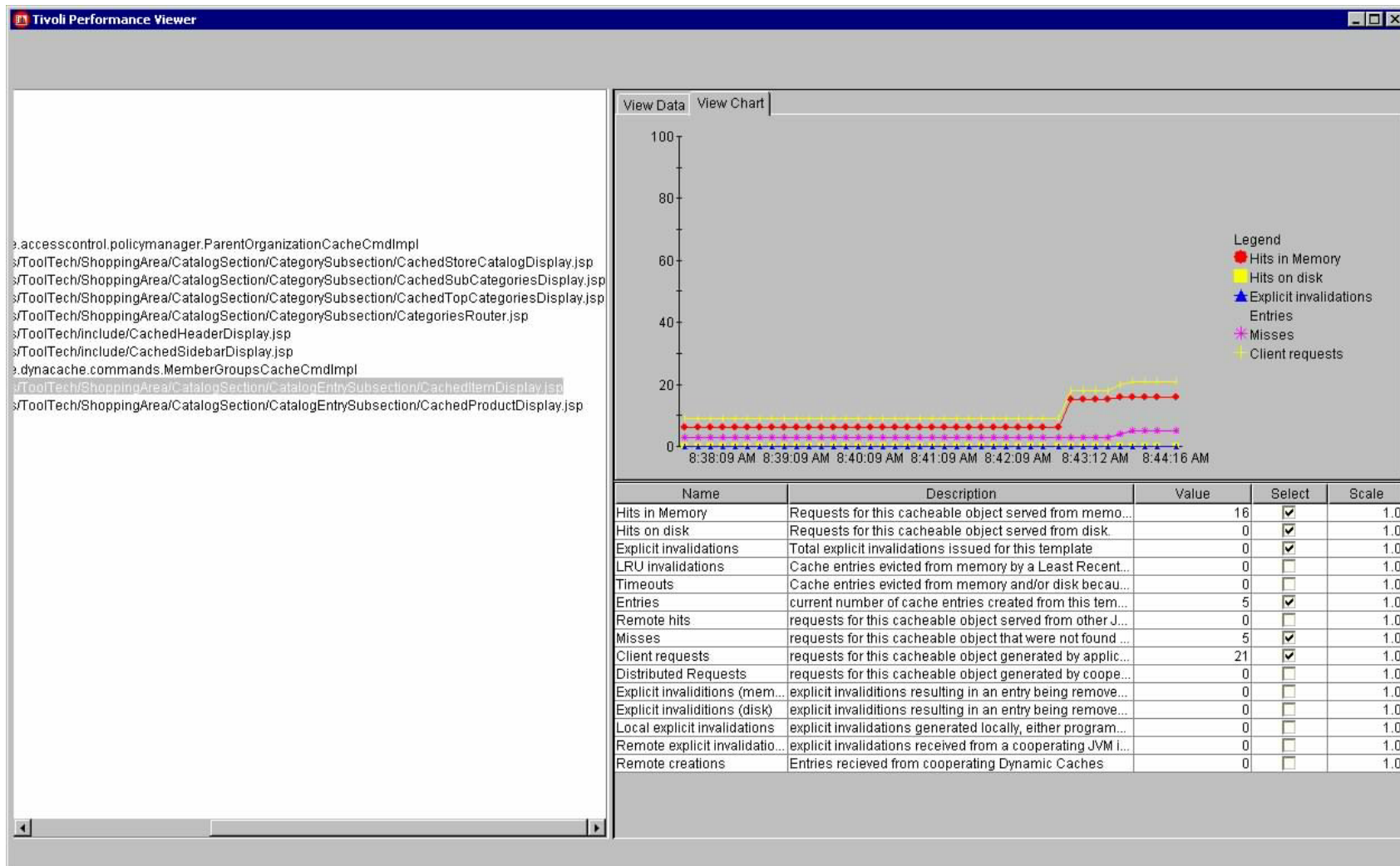


Figure 8. Tivoli Performance Viewer



I..

Figure 9. Tivoli Performance Viewer

5.1 Caching strategy

When determining a caching strategy for WebSphere Commerce, the key issues that need to be resolved are:

- Which pages should be cached
- Cache whole pages or page fragments
- Where caching should take place
- How to invalidate the cached data

5.1.1 Which pages should be cached

When determining which Web pages should be cached, good candidates for caching are pages that:

- are accessed frequently
- are stable for a period of time
- contain contents that can be reused by a variety of users

A good example would be catalog display pages.

5.1.2 Cache whole pages or page fragments

All Web pages consist of smaller and simpler *fragments*. An example of a page fragment could be a header, sidebar, footer or an e-Marketing Spot. Breaking a Web page into fragments or components makes more caching possible for any page, even for personalized pages. Fragments should be designed to maximize their reusability.

Dynamic cache service allows WebSphere Commerce to support caching of Web pages, and caching of fragments of pages. Caching a whole Web page simply means that the entire page is cached as big cache entry including all the content from all fragments that have no includes or forwards. This can save a significant amount of application server processing but is typically useful when the external HTTP request contains all the information needed to find the entry.

Web pages can be cached using whole page caching or fragment caching or a combination of both methods. If Web pages are broken into different fragments and the fragments are cached individually, then the fragments can be reused for a wider audience. When a Web page is requested, then different fragments are reassembled to produce the page.

So when determining caching strategy, whether the Web page should be cached as a whole or as a fragment is another important factor. However if the page output has sections that are user-dependent, then the page output is cached in a manner known as *fragment* caching. That is, the JSPs are cached as separate entries and will be reassembled when they are requested. If the page output for a particular WebSphere Commerce command always produces the same result based on the URL parameters and request attributes, then this page output can be cached with the *cache-entry* using the property *consume-subfragments* (CSF) , and the WebSphere

Commerce's store controller servlet (*com.ibm.commerce.server.RequestServlet*) as the servlet name. When the *cache-entry* is defined in this manner, the page output is cached the same way as in WebSphere Commerce Version 5.4 which is known as whole page caching.

The big advantage of using *consumed-sub fragments* with the *controller servlet* is performance but if this mechanism is used, then the page output cannot contain personalized information.

For fragment (JSP) caching, WebSphere Commerce has to execute the command (controller, task and view commands) to identify which JSP is to be executed before dynamic cache can determine whether the JSP can be served from the cache or not. The advantage of this method is flexibility, because different cache entries could be reassembled to form a page based on user information.

Note: In order for a fragment to be cacheable, this fragment has to be executable on its own. That is, it can be executed independently of any other fragments of the Web page.

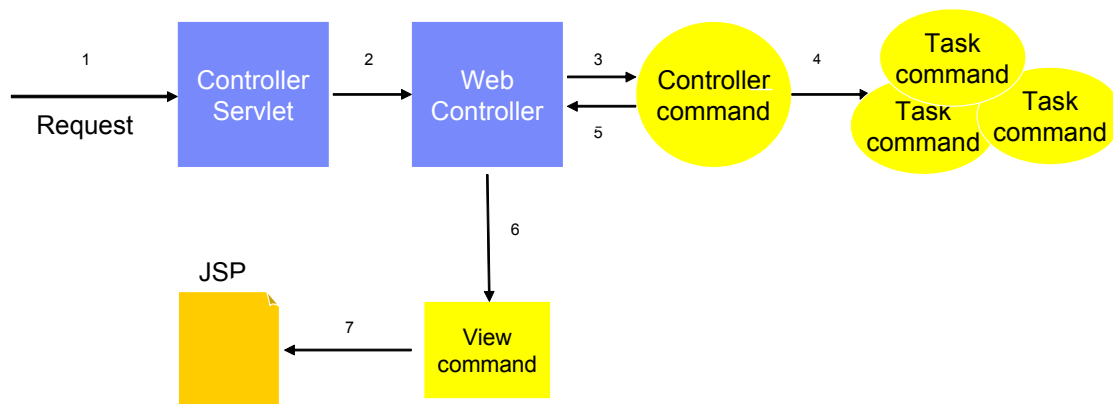


Figure 10. WebSphere Commerce's run time

Let us look at a sample Web page. Figure 11 shows a personalized Product Page which contains user-specific information. There is not much value to caching this whole page.

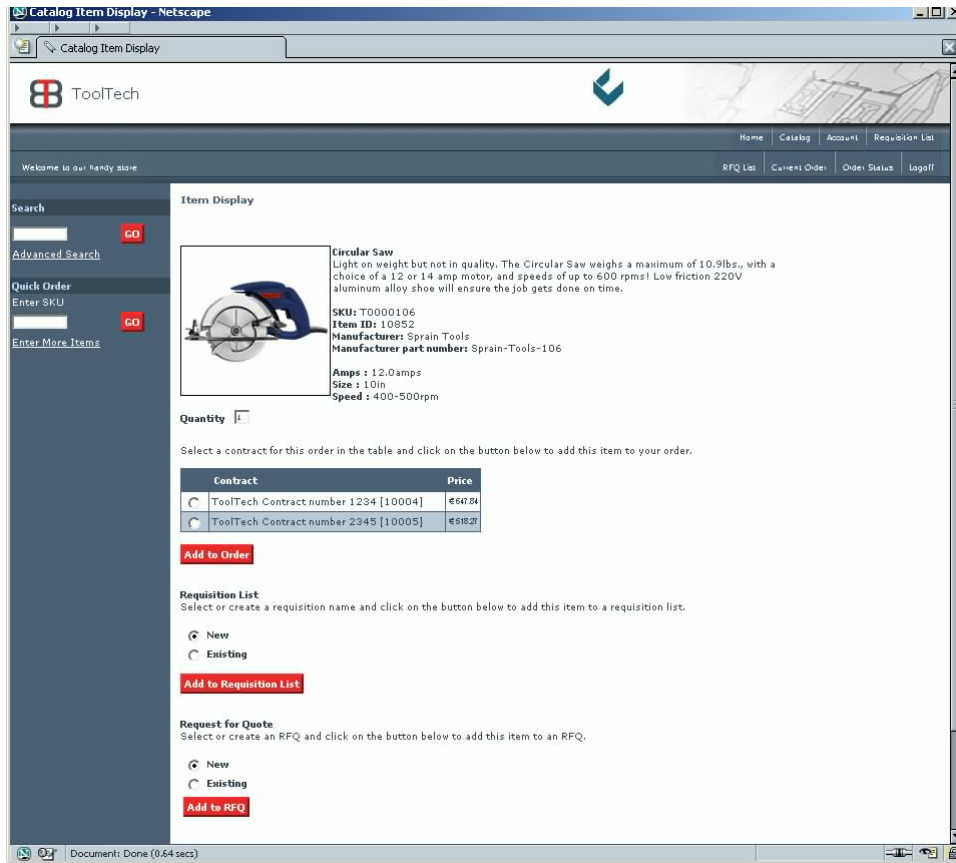


Figure 11. Example of a personalized page

Figure 12 shows the same page broken down into fragments based on reusability and cacheability. In this example, the page can be broken into different fragments:

- SideBar and Requisition list - same across all customers.
- Item Display - same across all customers for this product ID.
- Header and Request for Quote (RFQ) - same across all customers with the same user role(s).
- Add to order - same across all customers with the same contract(s) IDs.

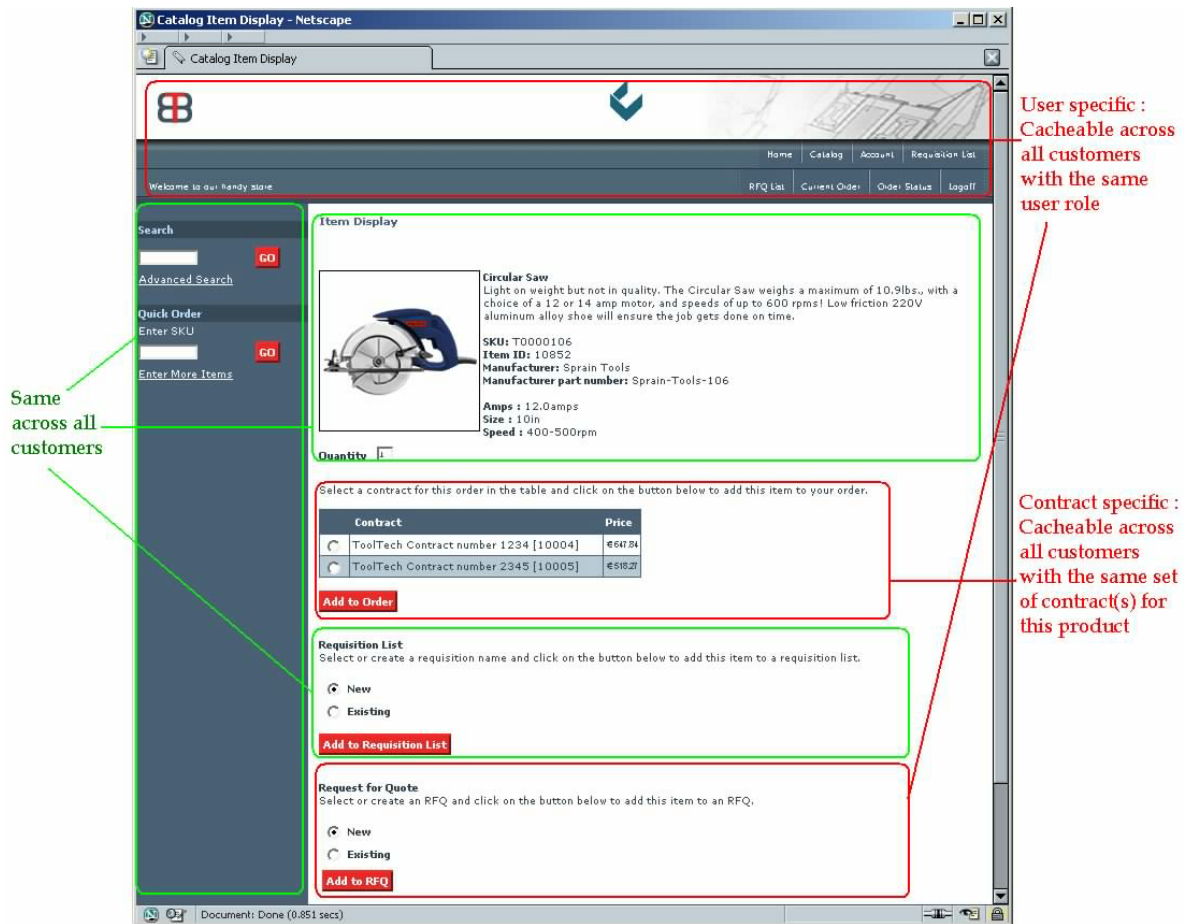


Figure 12 . Example of a personalized page fragmented for caching

In this case, all of the fragments become reusable or cacheable for a larger audience. Only fragments that are not cacheable need to be fetched from the backend, reducing server-side workload and improving performance.

5.1.3 Cache whole page excluding certain fragments

When there is small portion of a Web page that contains personalized information, for example, a personalized welcome message or a mini shopping cart, the *do-not-consume* property could become useful (**Note:** this property is only available with WebSphere Application Server version 5.0.2 + PQ81651 and later). To use this property, the parent entry would be marked with the property *consume-subfragments* and the child fragment that contains the personalization area would be marked with this *do-not-consume* property. With this combination, the performance gain of whole page caching remains intact for the entire page besides the child fragment where it will be cached separately apart from its parent.

The following Web page which is a common product page and its sidebar dynamically includes a fragment (MiniCartDisplay.jsp) that displays a personalized mini shopping cart.

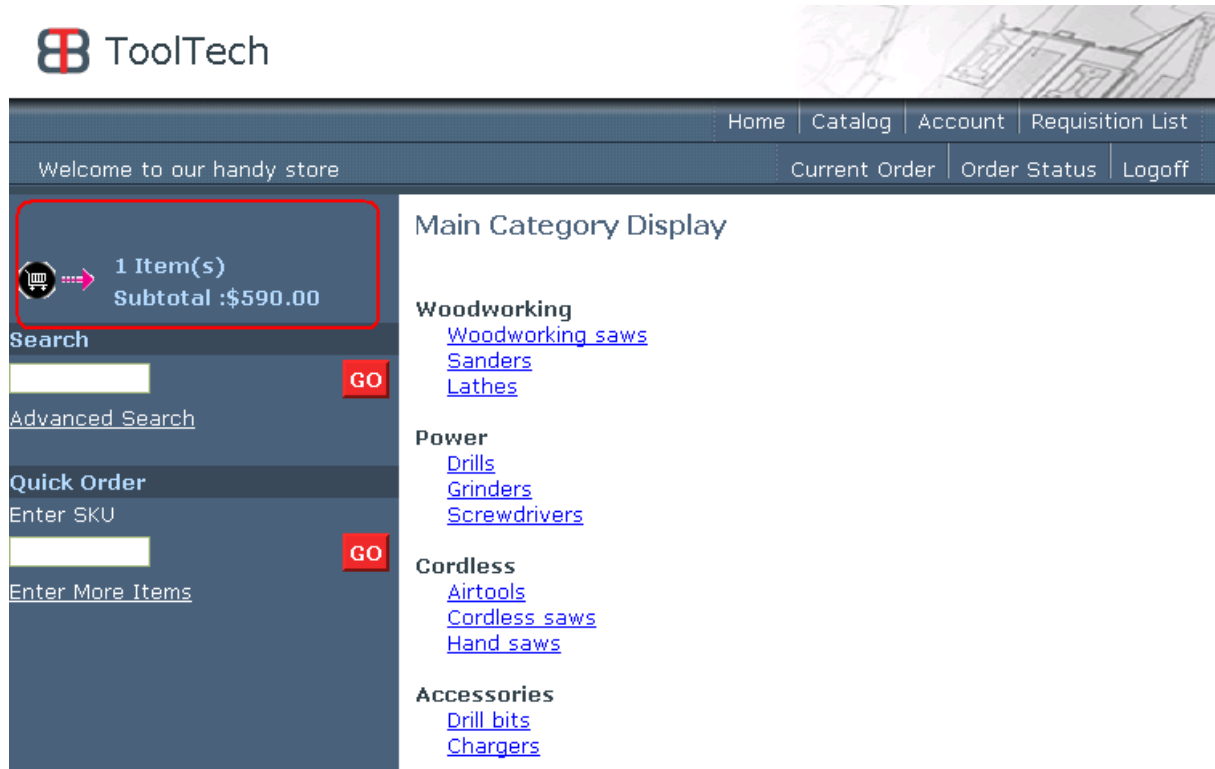


Figure 13. Product page with mini shop cart

In order to cache the product page and the mini shopping cart, we need to construct cache ID rules for them. Since the mini shopping cart is unique per user, therefore in order to cache it, we have to use the user's ID as the cache ID. WebSphere Commerce's cache filter (please refer to [Cache filter](#) section for more details) set up a request attribute *DC_userId*, so we can use that as the cache ID.

Here is an example of the cachespec.xml:

```
<cache-entry>
  <class>servlet</class>
  <name>/ToolTech/include/MiniShopCart.jsp</name>
  <property name="do-not-consume">true</property>
  <property name="save-attributes">false</property>

  <cache-id>
    <component id="DC_userId" type="attribute">
      <required>true</required>
    </component>
  </cache-id>
</cache-entry>
```

```

        </component>
    </cache-id>
</cache-entry>

<cache-entry>
    <class>servlet</class>
    <name>com.ibm.commerce.server.RequestServlet.class</name>
    <property name="store-cookies">false</property>
    <property name="save-attributes">false</property>
    <property name="consume-subfragments">true</property>

    <cache-id>
        <component id="" type="pathinfo">
            <required>true</required>
            <value>/TopCategoriesDisplay</value>
        </component>
        <component id="storeId" type="parameter">
            <required>true</required>
        </component>
        <component id="catalogId" type="parameter">
            <required>true</required>
        </component>
    </cache-id>
</cache-entry>

```

Here is what the cache entries will look like:

Current Cache Contents

Entries 1 through 2 of 2

< 0 >

Clear cache

Template	Cache ID	Timeout (sec)
/webapp/wcs/stores/ToolTech/include/MiniShopCart.jsp	/webapp/wcs/stores/ToolTech/include/MiniShopCart.jsp;DC_userId=2;ISO8859-1	0
/webapp/wcs/stores/servlet	/webapp/wcs/stores/servlet;pathinfo=/TopCategoriesDisplay;storeId=10001;catalogId=10001;ISO8859-1	0

Figure 14. Cache contents

5.1.4 Where caching should take place

Theoretically, caching should take place in the tier closest to the user. In reality, other factors such as security and user specific data may influence the choice of the best place to cache the content. To maximize the benefit of dynamic caching,

elements of a page should be fragmented as finely as possible so that they can be cached independently in different cache entries.

For example, the non-user specific, non-security sensitive fragments are generally useful to many users, and can be cached in a more public space and closer to the user. The security sensitive data should be cached behind the enterprise firewall.

5.1.5 How to invalidate the cached data

The hardest part of caching is to guarantee the accuracy of the content. The mechanism that is used to identify and update pages or fragments that are no longer valid is known as *invalidation*.

WebSphere Application Server dynamic cache service provides invalidation techniques that are rule-based, time-based, group-based, and programmatic. The service can also invalidate any remote caches that were configured externally.

6.1 Implementing Caching Strategy

After the caching strategy has been decided, you will need to implement it by creating a cache policy file. Before doing that, we need to review some topics that are relevant when creating a cache policy.

6.1.1 JSP include mechanisms

There are two JSP include mechanisms: the `@include` directive and the `jsp:include` action.

JSP Element	Syntax	Interpretation
<code>@include</code>	<code><%@include file="relative URL"%></code>	translation-time
<code>jsp:include</code>	<code><jsp:include page="{relativeURL <%expression %>}" flush="true"/></code>	request-time

The `@include` directive is static, in the sense that it is interpreted at compile time. It includes text from the referenced HTML or JSP file before the including JSP is compiled. Once the including JSP has been compiled, subsequent changes to the included file have no effect on the output produced by the including JSP, until it is recompiled.

On the other hand, the `jsp:include` action is dynamic. The referenced JSP is re-executed each time the including JSP is executed. The resulting dynamically generated text (referred to as a "JSP fragment") is included in the output of the including JSP.

When the WebSphere Application Server dynamic cache caches the output of a JSP, it also caches the JSP fragments produced by executing its `jsp:include` actions. Subsequent requests for the including JSP are served from the cache as long as all its included fragments can also be served from the cache. If any of its fragments are no longer in the cache (a "cache miss") the including JSP is re-executed. However, any fragments still in the cache are included, thus avoiding re-execution of the `jsp:include` actions that produced them.

6.1.2 Session dependent and session independent

Prior to WebSphere Commerce Version 5.5, WebSphere Commerce provided its own caching mechanism and Web page output was cached using one of two methods:

- Session independent (SI) caching - page output was cached based on URL parameters.
- Session dependent (SD) caching - page output was cached based on URL parameters and session information such as the user's language, preferred currency, parent organization, contract IDs, and member groups.

WebSphere Commerce generated the cache IDs for SI based on the URL parameters. SD cache IDs are based on URL parameters plus the user's language, preferred currency, parent organization, contract IDs, and member groups.

WebSphere Commerce Version 5.5 makes use of the WebSphere Application Server dynamic cache for serving dynamic content and caching data. Dynamic cache uses request parameters, HTTPSession, cookie, request attribute, header, pathinfo, servletpath and locale to construct the cache and dependency ID for servlet/JSP result caching. Therefore in order for dynamic cache to retrieve the session information to construct the cache ID, a servlet filter known as the *cache filter* is introduced. The cache filter is designed to set up request attributes from the session data, MemberGroupsCacheCmd, and store relationship information.

6.1.2.1 Cache filter

WebSphere Commerce's session data is set by the WebSphere Commerce Server run time, therefore the cache filter cannot retrieve enough information to set all the required request attributes until the second hit against the web site. Whenever the cache filter cannot retrieve the session information, the request attributes are not set, and the page is not retrieved.

6.1.2.1.1 Session data request attributes

The request attributes set by the *cache filter* are based on the session information and URL parameters. Here is the list of request attributes being set by the *cache filter* for session information:

Request Attributes	Description
DC_curr	User's preferred Currency
DC_lang	User's current Language
DC_porg	User's Parent Organization
DC_cont	User's current contract
DC_buyCont	User's eligible buyer's contract (for SupplyChain only)
DC_mg	User's explicit member groups
DC_storeId	Store ID
DC_userId	User ID

Since a user can be eligible for multiple contracts and can belong to multiple member groups, the request attributes *DC_cont* and *DC_mg* might contain multiple values. For such a user, the values are sorted and concatenated with a semicolon(";") as a separator. In addition, multiple contract and member group request attributes will be defined. (e.g. *DC_cont0*, *DC_cont1*, ..., *DC_contN* where *N* is the number of contracts the user is entitled to).

For example, if a user is eligible for contracts 10004 and 10005, then the following request attributes will be set up:

- *DC_cont* is 10004; 10005
- *DC_cont0* is 10004
- *DC_cont1* is 10005

For details about *DC_mg*, please refer to the *MemberGroupsCacheCmd* section of this whitepaper.

6.1.2.1.2 Store relationship request attributes

In order to facilitate sharing resources, including store assets, WebSphere Commerce introduces the concept of store relationship.

Store relationship information is needed for caching because when the store pages that are cached use shared catalogs, then whenever the shared catalog is changed then the store pages must be invalidated. In order for the cache entries to build up the relationship for cache invalidation, the *cache filter* needs to provide the store relationships information.

The *cache filter* sets up the store relationships by calling the *getStorePath()* and *getStoresForRelatedStore()* methods from the *StoreAccessBean* and sets up the request attributes as follows:

Store Relationship Type	Store Relationship Identifier	Request Attributes Name for <i>getStorePath()</i>	Request Attributes Name for <i>getStoresForRelatedStore()</i>
IBM commerce businessPolicy	-1	DC_busN	DC_bus_RS_N
IBM commerce tax	-2	DC_taxN	DC_tax_RS_N
IBM commerce campaigns	-3	DC_campN	DC_camp_RS_N
IBM commerce catalog	-4	DC_catN	DC_cat_RS_N
IBM commerce command	-5	DC_cmdN	DC_cmd_RS_N
IBM commerce hosted Store	-6	DC_hostN	DC_host_RS_N
IBM commerce price	-7	DC_prcN	DC_prc_RS_N
IBM commerce referral	-8	DC_refN	DC_ref_RS_N
IBM commerce	-9	DC_segN	DC_seg_RS_N

segmentation			
IBM commerce URL	-10	DC_urlN	DC_url_RS_N
IBM commerce view	-11	DC_viewN	DC_view_RS_N
IBM commerce inventory	-13	DC_invN	DC_inv_RS_N
IBM commerce base item	-14	DC_baseItemN	DC_baseItem_RS_N
IBM commerce channel store	-15	DC_chsN	DC_chs_RS_N
IBM commerce currency conversion	-17	DC_currConvN	DC_currConv_RS_N
IBM commerce currency format	-18	DC_currFmtN	DC_currFmt_RS_N
IBM commerce supported currency	-19	DC_supCurrN	DC_supCurr_RS_N
IBM commerce counter value currency	-20	DC_cterCurrN	DC_cterCurr_RS_N
IBM commerce measurement format	-21	DC_meaFmtN	DC_meaFmt_RS_X

Since the methods to determine the store relationship for each resource type return an array of store IDs and since dynamic cache does not support an array of request attributes, the *cache filter* will set up multiple request attributes when multiple store IDs are returned.

For details about store relationship, refer to the *WebSphere Commerce Store Development Guide*.

6.1.2.1.3 MemberGroupsCacheCmd

Since the member group information is not part of the session data, the *cache filter* must retrieve this information from the database based on the user ID. In order to prevent performance degradation due to repetitive database queries, a new command, *com.ibm.commerce.dynacache.command.MemberGroupsCacheCmd* extends WebSphere's *CacheableCommand* to cache the member groups each user belongs.

The request attributes defined by the *cache filter* are then used by dynamic cache by specifying as a component elements. These components can be specified in the *cache-id*, *dependency-id* or *invalidation* rules. Since the session information is being set up as *component* elements, then the user can customize to their need. For example, if a site only support one language and one currency, then *DC_lang* and *DC_curr* might not be needed to uniquely identify a cache entry.

Here is an example:

```
<cache-entry>
  <class>servlet</class>
  <name>...</name>
  ...
  <cache-id>
    ...
    <component id="DC_lang" type="attribute">
      <required>true</required>
    </component>
    <component id="DC_curr" type="attribute">
      <required>true</required>
    </component>
    <component id="DC_porg" type="attribute">
      <required>true</required>
    </component>
    <component id="DC_cont" type="attribute">
      <required>true</required>
    </component>
    <component id="DC_mg" type="attribute">
      <required>true</required>
    </component>
  </cache-id>
</cache-entry>
```

7.1 Edge Side Include (ESI) caching

Edge Side Include (ESI) is a simple markup language used to define Web page components for dynamic assembly and delivery of Web applications at the edge of the Internet.

For details of how to configure ESI caching, please refer to the “*Configuring Edge Side Include caching*” section in the *WebSphere Application Server Information Center*, available at the following URL:

<http://www.ibm.com/software/webservers/appserv/infocenter.html>

The ESI processor's cache can be monitored through the cache monitor application. In order for the ESI processor's cache to be visible in the cache monitor (refer to [Dynamic cache monitor](#) section for more details), the DynaCacheEsi application must be installed as described above and the *esiInvalidationMonitor* property must be set to *true* in the *plugin-cfg.xml* file.

For example,

```
<?xml version="1.0"?>
<Config>
  <Property Name="esiEnable" Value="true"/>
  <Property Name="esiMaxCacheSize" Value="1024"/>
  <Property Name="esiInvalidationMonitor" Value="true"/>
```

7.1.1 Caching static data

By default, static data such as images and HTML are cached by the ESI processor when served up by the WebSphere Application Server. The cached entries have a default time-out of 300 seconds. The time-out value can be changed by setting the system property *com.ibm.servlet.file.esi.timeOut* in your JVM.

For example:

```
-D com.ibm.servlet.file.esi.timeOut=60
```

By default, WebSphere Commerce does not cache static data. You can enable it by specifying a *cache-entry* in the *cachespec.xml* file. Refer to [Simple file servlet](#) section for more details.

7.1.2 Caching full pages using ESI

To mark an entry to be cached using ESI, use the property *EdgeCacheable*. This property also implies the property of *consume-subfragments*. That is, the page will be cached as a full page including all its sub fragments unless one of these sub fragments are specified to be cacheable separately (refer to [Caching fragments using ESI](#) for details). In order to cache pages with ESI, only the *parameter* and *cookie* component can be used to define the *cache* ID.

The following example shows the *cache-entry* that uses the ESI plugin:

```
<cache-entry>
  <class>servlet</class>
  <name>com.ibm.commerce.server.RequestServlet.class</name>
  <property name="store-cookies">>false</property>
  <property name="save-attributes">>false</property>
  <property name="EdgeCacheable">>true</property>

  <cache-id>
    <component id="" type="pathinfo">
      <required>true</required>
      <value>/StoreCatalogDisplay</value>
    </component>
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
  </cache-id>
</cache-entry>
```

7.1.3 Caching fragments using ESI

WebSphere Commerce uses the model-view-controller (MVC) programming model, where a calls to a controller servlet might include one or more child JSP files to construct the view. In order for the child JSP files to be edge cacheable, they have to be able to request these JSP files externally. WebSphere Commerce's JSPs use request attributes that are set by the controller servlet, so in order to cache these JSP files on the edge, we have to use the alternate URL support to provide an alternate controller servlet URL used to invoke the JSP file.

The following Web page which is a common product page and its sidebar dynamically includes a fragment (MiniCartDisplay.jsp) that displays a personalized mini shopping cart.

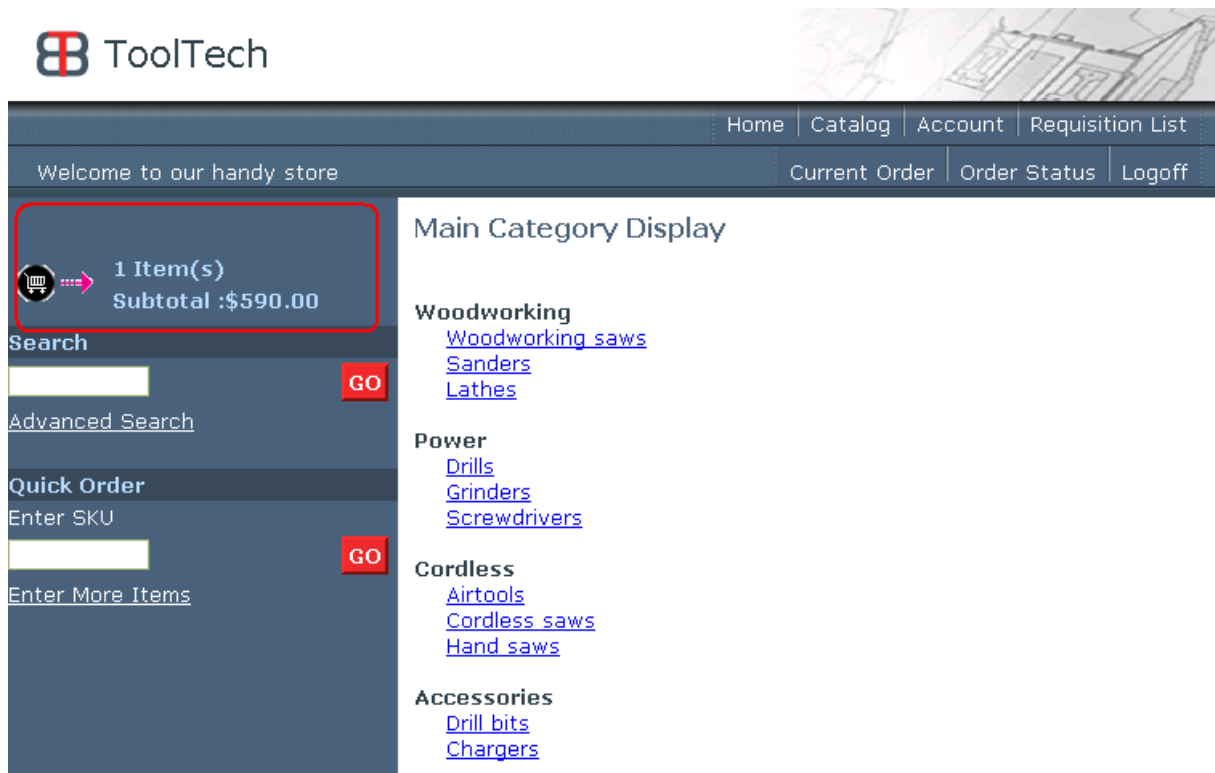


Figure 15. Product page with mini shop cart

In order to cache this product page and the mini shopping cart on the edge, we need to construct cache ID rules which only contain URL parameters or cookies. For the product page, it will not be a problem since all the information needed to cache it are on the URL. However, since the mini shopping cart is unique per user, in order to cache it on the edge, we have to use the user's ID as the cache ID.

Since only URL parameters and cookies can be used to define the cache ID rule, and the URL would not contain the user's information, the only other way is to use a custom cookie that contain the user's ID information and use that as the cache ID. In our testing, we have experimented with servlet filter chaining (please refer to the Servlet 2.3 specification for details) to chain up with WebSphere Commerce's cache filter (please refer to [Cache filter](#) section for details) to create a custom cookie named `WC_USERID` based on the request attribute `DC_userId` as created by the cache filter.

Here is an example of the cachespec.xml:

```
<cache-entry>
```

```

<class>servlet</class>
<name>/ToolTech/include/MiniShopCart.jsp</name>
<property name="EdgeCacheable">true</property>
<property
name="alternate_url">/servlet/ToolTech/include/MiniShopCart.jsp</property>
<property name="save-attributes">>false</property>
<property name="do-not-consume">>false</property>

<cache-id>
  <component id="WC_USERID" type="cookie">
    <required>true</required>
  </component>
</cache-id>
</cache-entry>

<cache-entry>
  <class>servlet</class>
  <name>com.ibm.commerce.server.RequestServlet.class</name>
  <property name="store-cookies">>false</property>
  <property name="save-attributes">>false</property>
  <property name="EdgeCacheable">true</property>

  <cache-id>
    <component id="" type="pathinfo">
      <required>true</required>
      <value>/TopCategoriesDisplay</value>
    </component>
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
  </cache-id>
</cache-entry>

```


The edge cache entries will be shown as follows:

Current Edge Cache Contents for Host: All Process: All

Entries 1 through 2 of 2

< 0 >

Refresh Contents

Clear cache

Cache ID	Host
GET_/webapp/wcs/stores/servlet/ToolTech/include/MiniShopCart.jsp_WC_USERID=2	barbwong
GET_/webapp/wcs/stores/servlet/TopCategoriesDisplay_storeId=10001:catalogId=10001	barbwong

Figure 16. Edge cache contents

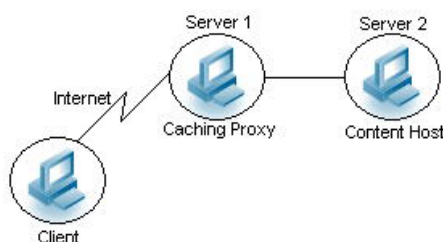
8.1 External cache groups

The dynamic cache has the ability to control caches outside of the application server, such as IBM Edge Server, non-z/OS IBM HTTP Server FRCA caches, and non-z/OS WebSphere HTTP Server plug-in ESI fragment processors. When external cache groups are defined, the dynamic cache matches externally cacheable entries with those groups, and pushes cache entries and invalidation out to those groups. This allows WebSphere to manage dynamic content beyond the application server. The content can then be served from the external cache, instead of the application server, improving savings in performance.

8.1.1 Edge components Caching Proxy

WebSphere Application Server edge components contains a Caching Proxy component that can cache static content and content dynamically generated by WebSphere Application Server. The Caching Proxy intercepts the data request from a client, retrieves the requested information from content-hosting machines, and delivers the content to the client. Most commonly, the requests are for documents stored on Web server machines (also called origin servers or content hosts) and delivered via HTTP.

The Caching Proxy stores cacheable content in a local cache before delivering it to the requester. This enables the Caching Proxy to satisfy subsequent requests for the same content by delivering it directly from the local cache, which is much quicker than retrieving it again from the back-end servers. WebSphere Commerce pages can be configured to make use of the Caching Proxy to greatly reduce the server response time. The following figure illustrates a basic Caching Proxy server network setup using three computer systems. This network binds the proxy to a dedicated content host (running IBM HTTP Server, WebSphere Application Server, WebSphere Commerce and DB2) located on Server 2. The following sections describe how an external cache group can be configured in dynamic cache with WebSphere Commerce to work with a Caching Proxy. Refer to the Appendix for information on installing and configuring the caching proxy.



8.1.1.1 External cache configuration at content host

1. Start the application server *server1*.
2. Log on to the Administrative Console *http://hostname:portNumber/admin*.

3. Click **Servers > Application Servers > WC_InstanceName**.
4. Scroll down and select **Dynamic cache Service > External Cache Groups**.
5. Click **New** and enter an external cache group name (e.g. IBM-WC-EDGE) and click **OK**. Note that this must be the same name as entered in the ExternalCacheManager directive in the ibmproxy.conf file (located in <EdgeDir>\cp\etc\en_US\ directory).
6. Now click on the external cache group name (e.g. IBM-WC-EDGE) and select **External Cache Group Members** and click on *New*.
7. Enter the fully qualified address or IP address of the Caching Proxy machine and enter the following in the Adapter Bean Name:
com.ibm.websphere.edge.dynacache.WteAdapter.
8. Click **OK** and save the changes.
9. Log out of the WebSphere Application Server Administrative Console.

8.1.1.2 Dynaedge-cfg.xml file modifications

1. Open the dynaedge-cfg.xml file located in *WASInstallDir/properties* directory.
2. Replace the line
`<EdgeServerCfg CacheManager="IBM-WAS">`
 with
`<EdgeServerCfg CacheManager=ExternalCacheGroupName>` .
 For example:
`<EdgeServerCfg CacheManager="IBM-WC-EDGE">`
3. After all of the commented sections, add the following two entries:
 - `<EdgeServer endpoint="http://<server2hostname>:80" invalidation-url="/WES_External_Adapter" URI-type="absolute" user="<userName>" userPasswd="<userPwd>"/>`
 - `<EdgeServer endpoint="https://<server2hostname>:443" invalidation-url="/invalidate-dynamic-contents" URI-type="absolute" user="<userName>" userPasswd="<userPwd>"/>`

The following is a sample entry showing how a cache entry in the cachespec.xml file is configured to cache externally on the Edge Components Caching Proxy. Unlike the ESI plugin (see section on ESI caching), only full page caching is supported at the Caching Proxy. The property EdgeCacheable allows parameters and cookies but not request attributes to be specified as components of a cache ID at the Caching Proxy. The ExternalCache property differentiates caching at the Cache Proxy from caching using ESI plugin. Note that the ExternalCache property must match the external cache group ID "IBM-WC-EDGE" as specified in previous steps.

```
<cache>
  <cache-entry>
    <class>servlet</class>
    <name>/ToolTech/.../StoreCatalogDisplay.jsp</name>
```

```
<cache-id>
  <property name="EdgeCacheable">true</property>
  <property name="ExternalCache">IBM-WC-EDGE</property>
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
</cache-id>
</cache-entry>
</cache>
```

To see Caching Proxy cache statistics go the Caching Proxy administration page at <http://<server1 hostname>:8008> -> Server Activity Monitor -> Cache Statistics.

9.1 Tutorial

The following tutorial will illustrate different approaches developing dynamic cache caching policies. The tutorial uses the B2BDirect sample store (referred to as the ToolTech store) included in WebSphere Commerce Version 5.5, however, the approaches used are general in nature and can be applied to any WebSphere Commerce store.

- How to define cache policies for caching as whole or page fragments.
- Defining cache policy.
- Examining JSPs for the ToolTech Store.

In the following discussion, we will be using the B2B Direct Business Model.

9.1.1 B2BDirect business model

After you have successfully created a WebSphere Commerce instance, you can publish a sample store. The following section describes how to publish the *B2BDirect* (ToolTech) store archive.

9.1.1.1 Publishing a store archive

Ensure that WebSphere Application Server for your WebSphere Commerce instance is started.

1. Launch the WebSphere Commerce Administration Console in a Web browser: <https://hostname.domain:8002/adminconsole>.

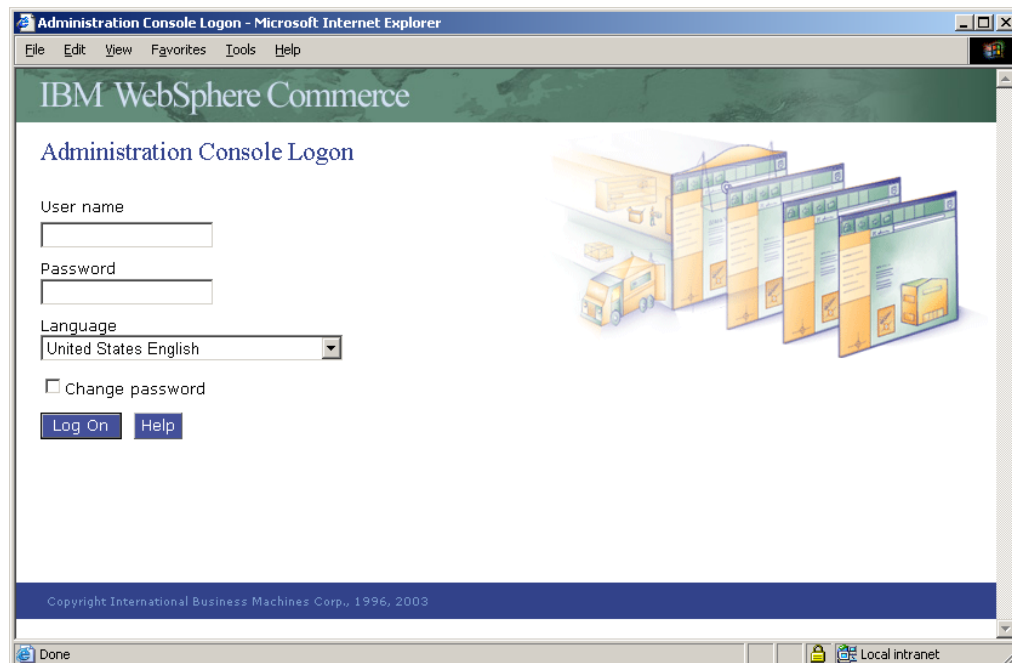


Figure 17. WebSphere Commerce's Administrative Console

2. The user name and password to access this site are *wcsadmin* and the site administration password that you set during installation.
3. Select **Site** and click **OK**.
4. From the Store menu, select **publish**.
5. On the Store Archive page, select **B2BDirect.sar**.
6. On the Parameters page, click **next**.
7. On the Options page, click **Finish**, then click **OK**.



Figure 18 . Publish store archive

8. After the store is published, click **Details** > **Launch Store** to launch the store.

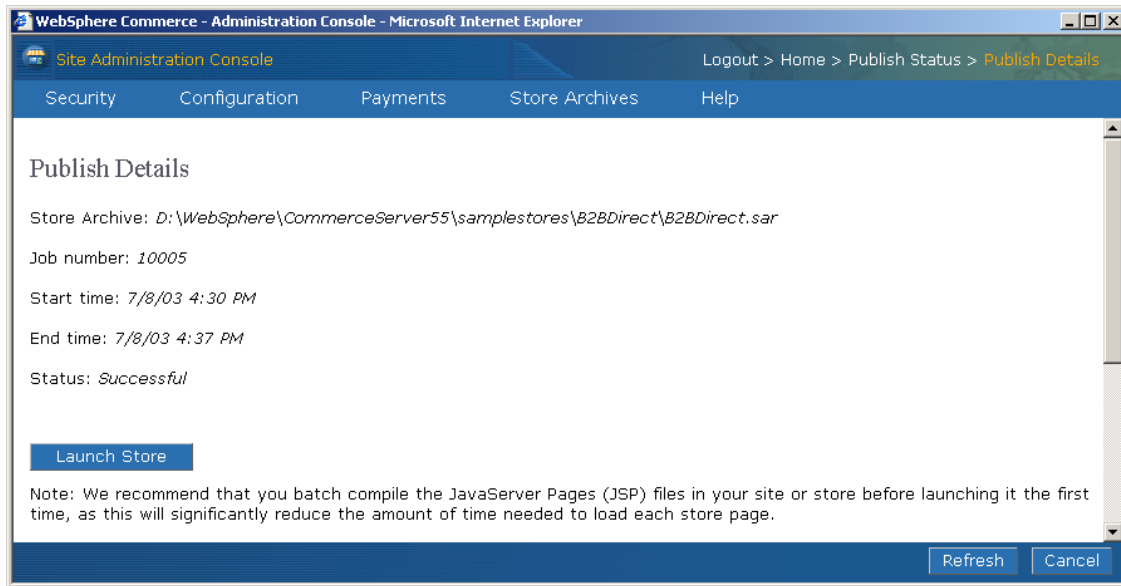


Figure 19 . Publish Details

The JSPs for the published ToolTech store are located in the following directory:
`WASInstallDir\installedApps\cell_name\WC_instanceName.ear\Stores.war\ToolTech`

9.1.2 Defining cache policy

In order for dynamic cache to cache servlets, *cache-entry* has to be defined in the cache policy configuration file (*cachespec.xml*).

For caching servlet and JSP result:

- The **class** element must be defined as *servlet*
- The **name** element must be defined and may include either the full URI of the JSP or the fully qualified class name of the servlet

For example:

```
<name>com.ibm.commerce.server.RequestServlet.class</name>
<name>/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/StoreCatalogDisplay.jsp</name>
```

Note:

WebSphere Commerce has only one controller servlet for the Stores.war named *com.ibm.commerce.server.RequestServlet.class*. Therefore, when caching with the servlet class, a filtering mechanism is needed to determine what needs to be cached. This can be accomplished by using the attribute type *pathinfo* in the *component* sub-element in the *cache-id* specification.

For example:

```
<component id="" type="pathinfo">
  <required>true</required>
  <value>/StoreCatalogDisplay</value>
</component>
```

9.1.2.1 Caching whole pages

If the Web page can be cached as a whole page, then the *cache-entry* that you specified is as follows. This example shows caching for the *StoreCatalogDisplay* command.

```
<cache-entry>
  <class>servlet</class>
  <name>com.ibm.commerce.server.RequestServlet</name>
  <property name="consume-subfragments">true</property>
  <property name="store-cookies">>false</property>
  <property name="save-attributes">>false</property>

  <cache-id>
    <component id="" type="pathinfo">
      <required>true</required>
      <value>/StoreCatalogDisplay</value>
    </component>
    ...
  </cache-id>
</cache-entry>
```

9.1.2.2 Caching page fragments

The following example shows how to specify a *cache-entry* for caching page fragments for *StoreCatalogDisplay.jsp*:

```
<cache-entry>
  <class>servlet</class>
  <name>>/ToolTech/ShoppingArea/CatalogSection/CategorySub
  section/StoreCatalogDisplay.jsp</name>
  <property name="save-attributes">>false</property>

  <cache-id>
    ...
  </cache-id>
</cache-entry>
```

9.1.2.3 Defining cache IDs

Each *cache-id* element defines a rule for caching an object and is composed of the sub-elements component, time-out, priority, and property. When defining cache

IDs for *cache-entry*, one should take into consideration how the servlet and JSP result output would be cached. If whole page caching is being used for the WebSphere Commerce command, then the command parameters will be the key factor for defining the cache ID rules. If the page output is to be cached by JSPs then the parameters for the JSPs are the key. Also in some cases, the request attributes set by the *cache filter* should also be used to define the rules. For example, if some session information like the user's preferred language and currency are a mandatory key to make the cache ID unique.

For example:

```
<cache-entry>
  <class>servlet</class>
  <name>/ToolTech/ShoppingArea/CatalogSection/CategorySubsect
ion/StoreCatalogDisplay.jsp</name>
  <property name="save-attributes">false</property>

  <cache-id>
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
    <component id="DC_lang" type="attribute">
      <required>true</required>
    </component>
  </cache-id>
</cache-entry>
```

For the above example if the URL

<http://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?langId=-1&storeId=10001&catalogId=10001> is invoked, the cache ID generated will be:
/webapp/wcs/stores/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/StoreCatalogDisplay.jsp:storeId=10001:catalogId=10001:DC_lang=-1:ISO-8859-1

As you can see each component defined becomes part of the cache ID generated.

9.1.2.4 Defining dependency IDs

A dependency ID is also known as grouping ID because it can be used to group multiple cache entries with the same group identifier. Dependency ID rules should be defined with cache invalidation in mind so that when a product is updated for a particular store, catalog, etc., the cache entries can be invalidated as a bundle.

For example:

```

<cache-entry>
  <class>servlet</class>
  <name>/ToolTech/ShoppingArea/CatalogSection/CategorySubs
ection/StoreCatalogDisplay.jsp</name>
  <property name="save-attributes">false</property>

  <cache-id>
    ...
  </cache-id>
  <dependency-id>storeId
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>
  <dependency-id>catalogId
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>
</cache-entry>

```

For the above example if the URL <http://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?langId=-1&storeId=10001&catalogId=10001> is invoked, the dependency IDs generated are:

- **storeId:10001**
- **catalogId:10001**

9.1.2.5 Store relationship considerations

If you are caching Web pages that involve resources shared, then you might have to set up your dependency IDs differently to define the resources relationship in order to support cache invalidation.

In order to explain this, we need to use Demand Chain business model. To do this publish the *DemandChain.sar* and create a hosted store (i.e. ResellerOne). The store relationships are defined in the *STOREREL* table. For details on creating a hosted store, please refer to the *WebSphere Commerce Sample Store Guide*. The ResellerOne store (10051) uses the CommercePlazaCatalog store (10002) as the catalog profile store.

The following table shows the store IDs for this example:

Store ID	Directory	Store Type
10001	CommercePlaza	Hosted Channel Plaza
10002	CommercePlazaCatalog	Catalog Profile
10003	CommercePlaza	Distributor Proxy
10004	ConsumerDirectResellerProfile	Reseller Profile

10051	ResellerOne	Reseller Hosted Store
-------	-------------	-----------------------

The following shows the store relationships:

Store ID	Relationship Type	getStorePath()	getStoresForRelatedStore()
10001	-1 (Business Policy) -4 (Catalog) -7 (Price) -17 (Currency Format) -19 (Currency Supported)	10002	N/A
10001	-6 (Hosted Store)	10051	n/a
10051	-1 (Business Policy) -14 (Base Item)	[10051, 10002, 10004]	[10051]
10051	-2 (Tax) -3 (Campaigns) -5 (Command) -10 (URL) -11 (View)	[10051, 10004]	[10051]
10051	-4 (Catalog) -7 (Price) -17 (Currency Conversion) -18 (Currency Format) -19 (Currency supported) -20 (Counter value currency) -21 (Measurement format)	[10051, 10002]	[10051]

The following request attributes will be set up by the *cache filter*:

Store Relationship	store ID 10051	store ID 10051	store ID 10001
-1 (Business Policy)	DC_bus0=10051 DC_bus1=10002 DC_bus2=10004	DC_bus_RS_o=10051 1	DC_bus0=10002 2
-2 (Tax)	DC_tax0=10051 DC_tax1=10004	DC_tax_RS_o=10051	
-4 (Catalog)	DC_cato=10051 DC_cat1=10002	DC_cat_RS_o=10051	DC_cato=10002
-6 (Hosted Store)	DC_host0=10051	DC_host_RS_o=10001	DC_host0=10051 1

In this example, whenever the catalog of the catalog profile store (storeId 10002) is changed, the catalog pages of ResellerOne (storeId 10051) must be invalidated. In order to do that, extra dependency IDs must be set up for this store relationship. So for *StoreCatalogDisplay*, the extra dependency IDs are set up as follows:

```
<!-- Start Store Relationship Dependency Ids -->
<!-- DC_cat1 is the catalog Profile Store ID -->
<dependency-id>storeId
  <component id="DC_cat1" type="attribute">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>storeId:catalogId
  <component id="DC_cat1" type="attribute">
    <required>true</required>
  </component>
  <component id="catalogId" type="attribute">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>StoreCatalogDisplay:storeId
  <component id="DC_cat1" type="attribute">
    <required>true</required>
  </component>
</dependency-id>
<!-- Ends Store Relationship Dependency Ids -->
```

The extra dependency IDs created are:

- **storeId:10002**
- **storeId:catalogId:10002:10051**
- **StoreCatalogDisplay:storeId:10002**

Since storeId 10002 is the catalog profile store, whenever there are changes to the store catalog, the catalog pages for the hosted store will also be invalidated.

9.1.3 Caching the ToolTech store catalog pages

Web page output of the ToolTech store is driven by JSP processing. Each page is composited of one or more JSP invocations. In order to cache the servlet or JSP results, we need to define a cache policy in the *cachespec.xml* file. Each cacheable object must be defined as a *cache-entry* element and each *cache-entry* must specify certain basic information that the dynamic cache uses.

Let us define a cache policy to cache the page output for the *StoreCatalogDisplay* command.

9.1.3.1 Caching as whole page or page fragments

The first thing to do is to determine if the page output should be cached as a whole page or as page fragments. When invoking the URL

https://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?langId=-1 & storeId=10051&catalogId=10051, one of following page will be displayed:

- StoreCatalogDisplay page output for guest user

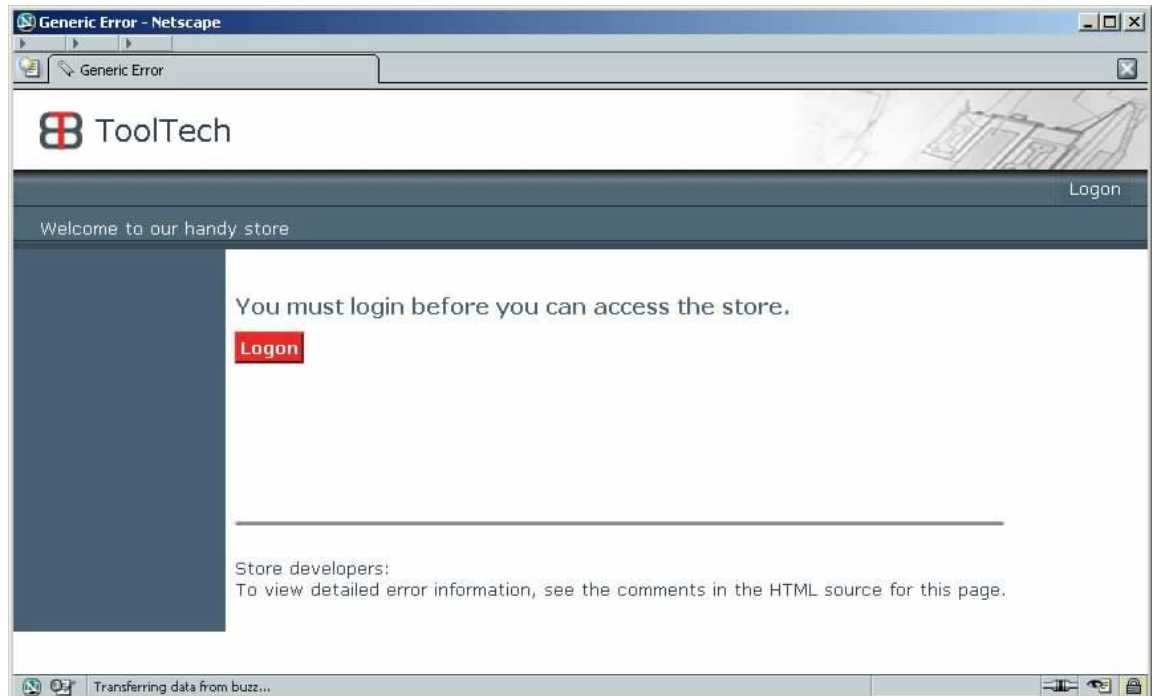


Figure 20 . ToolTech's StoreCatalogDisplay for guest user

- StoreCatalogDisplay page output for registered user

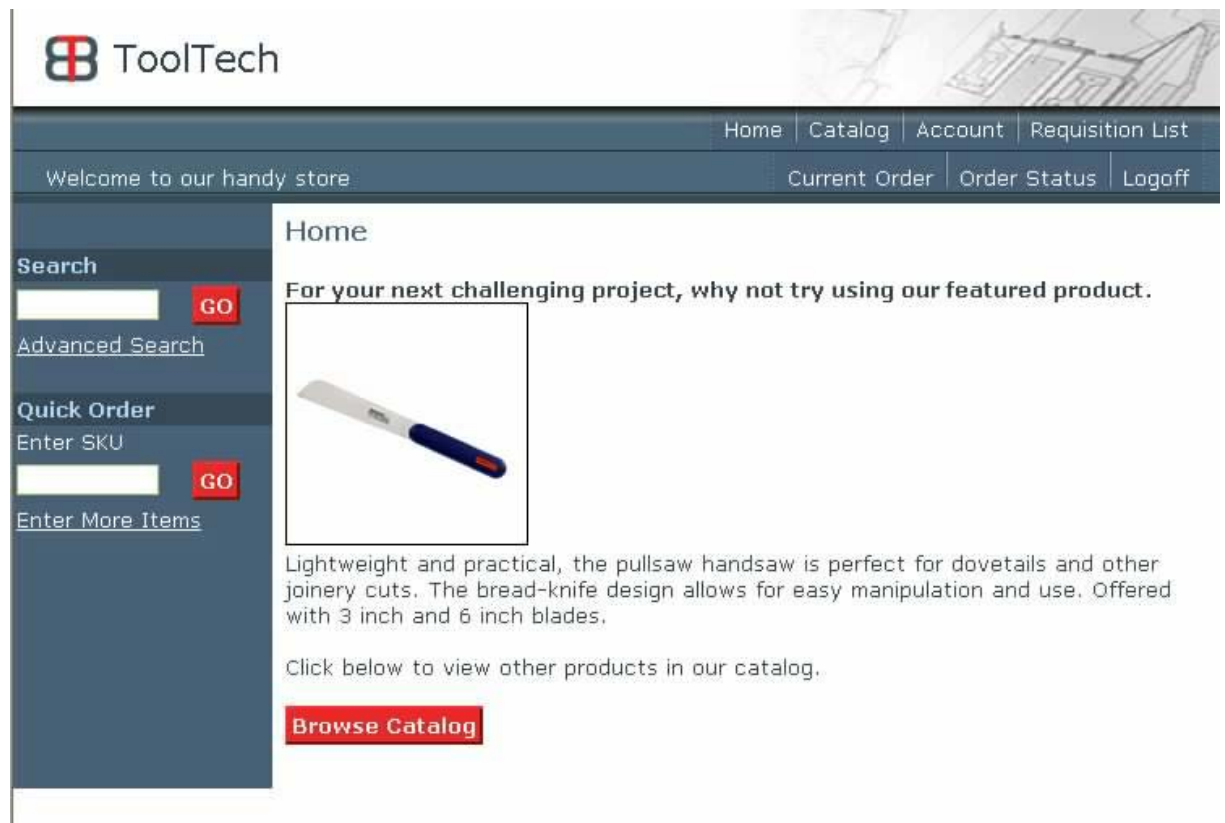


Figure 21 . ToolTech's StoreCatalogDisplay for registered user

- StoreCatalogDisplay page output for registered User with Buyer(Buyer-side) & Buyer Approver role

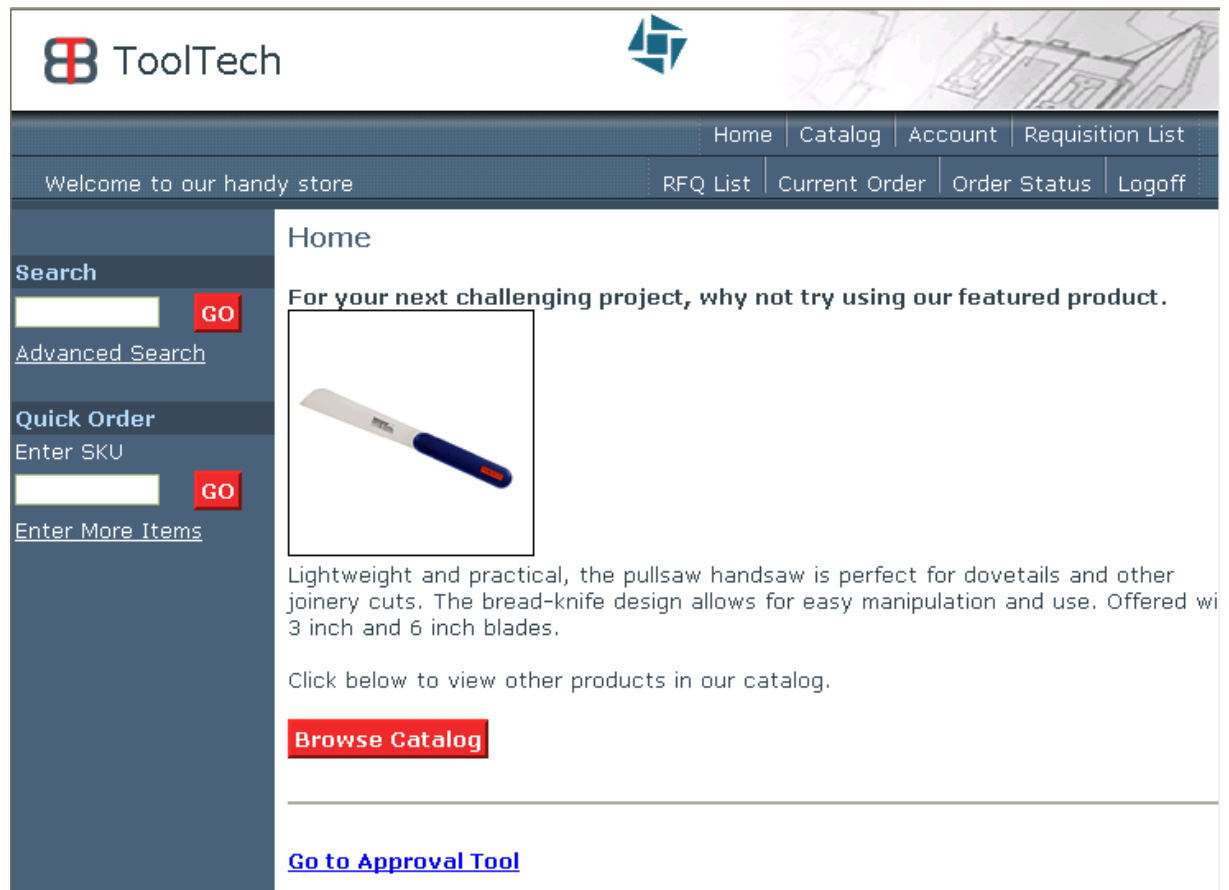


Figure 22 . ToolTech's StoreCatalogDisplay for registered user with Buyer(Buyer-Side) & Buyer Approver Role

For the ToolTech store, the page output for the *StoreCatalogDisplay* command is dependent on the user's information, therefore it is not possible to cache the page output as a whole, and fragment caching is used.

In order to cache the output as page fragments, we need to determine which JSP is being invoked by the *StoreCatalogDisplay* command. This can be done by looking up the *properties* column in the *VIEWREG* table.

Issue the following from the DB2 command line:

```
db2 select properties from viewreg where storeent_id=10051 and
viewname='StoreCatalogDisplayView'
```

You should see output like the following:

PROPERTIES

docname=ShoppingArea/CatalogSection/CategorySubsection/StoreCatalogDisplay.jsp

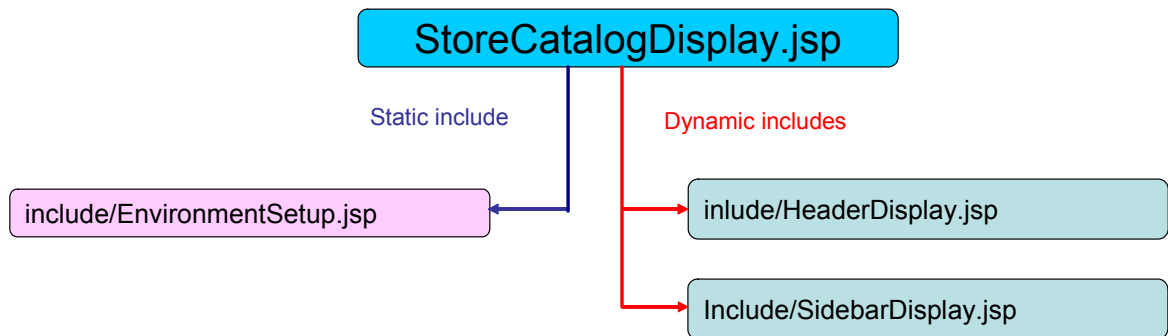
1 record(s) selected.

The JSP being invoked is
/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/StoreCatalogDisplay.jsp because the store directory is *ToolTech*.

Now examine the *StoreCatalogDisplay.jsp*. We can see that it includes the following JSPs:

- ToolTech/include/EnvironmentSetup.jsp
- ToolTech/include/HeaderDisplay.jsp
- ToolTech/include/SidebarDisplay.jsp

So in order to cache, we need to examine whether the JSPs can be cached and if so, we need to define a cache-entry for each JSP.



.Figure 23 . Original StoreCatalogDisplay.jsp includes

9.1.3.2 Defining cache policies for the JSPs

In order to define cache entries to cache the JSP result, the JSPs need to be examined in detail.

Note:

For details of the JSPs, refer to *original StoreCatalogDisplay.jsp*, *original HeaderDisplay.jsp* and *original SidebarDisplay.jsp* in [Sample JSPs](#) section of this document.

9.1.3.2.1 StoreCatalogDisplay.jsp

By examining *StoreCatalogDisplay.jsp*, we can see that it does the following:

- Display an E-Marketing spot *StoreHomePage*

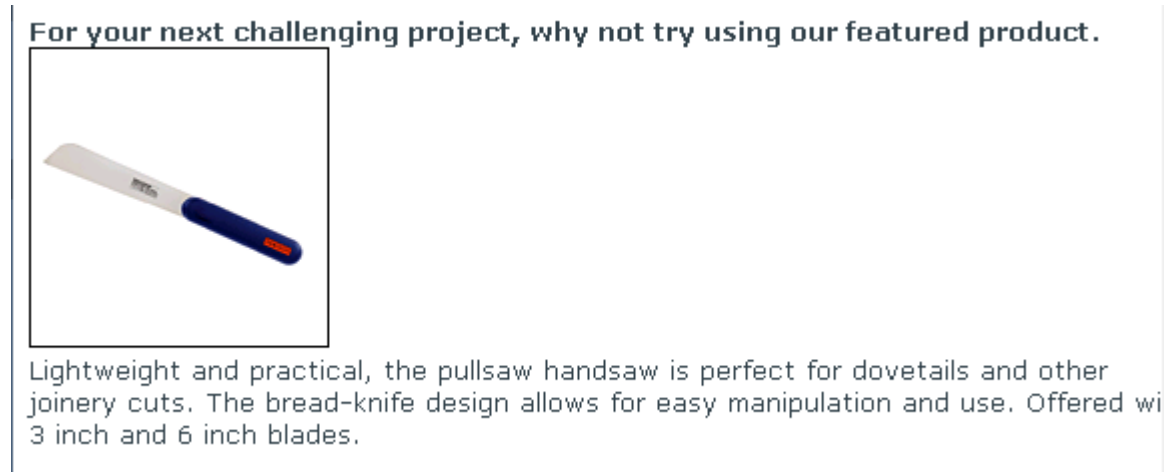


Figure 24. ToolTech's StoreCatalogDisplay StoreHomePage

- Display a *Browse Catalog* button



Figure 25 . ToolTech's StoreCatalogDisplay *Browse Catalog* Button

- Display a *Go to Approval Tool* link if the user is a *Buyer Administrator* (role id -21) or *Buyer Approver* (role id -22) and is a *Buyer (Buy-side)* (role id -24)

[Go to Approval Tool](#)

Figure 26 . ToolTech's StoreCatalogDisplay *Approval Tool* Link

- Redirects the user to the LogonForm if the user is a *Guest* user.

9.1.3.2.2 HeaderDisplay.jsp

Different variations are displayed based on user's information:

- Guest user



Figure 27 . ToolTech's HeaderDisplay for Guest User

- Registered user



Figure 28 . ToolTech's HeaderDisplay for Registered User

- Registered user with Buyer (Buy-side) - role id -24



Figure 29 . ToolTech's HeaderDisplay for Registered User with Buyer(Buy-side)

9.1.3.2.3 SidebarDisplay.jsp

For guest users the *SidebarDisplay.jsp* is blank and for registered users it displays the following:

Figure 30 . ToolTech's SidebarDisplay for registered User

As you can see, different variations of the page output from *StoreCatalogDisplay.jsp*, *HeaderDisplay.jsp* and *SidebarDisplay.jsp* are all based on the user's information, such as user's type, user's role, etc. Therefore if we want to cache the header, sidebar, and the rest of the *StoreCatalogDisplay* page, we should use the user's information as part of the cache ID.

9.1.3.3 Marketing campaign

In the ToolTech Store, you can create campaigns based on the accounts to which you belong. Please refer to the section *Creating a marketing campaign* under *B2B direct sample (ToolTech)*, in the *WebSphere Commerce Sample Store Guide* for details.

The *StoreCatalogDisplay.jsp* displays the *StoreHomePage* e-Marketing spot and since an e-Marketing spot should not really be cached, then you should change the JSP, move it to a separate JSP, and use an include action to include it. Also the e-Marketing spot bean *com.ibm.commerce.marketing.beans.EMarketingSpot* needs the e-Marketing Spot name and catalog ID, therefore we need to pass them as parameters.

```
<jsp:include page="../../../include/StoreCatalogProductESpot.jsp" flush="true">
  <jsp:param name="emsName" value="StoreHomePage" />
  <jsp:param name="catalogId" value="<%= catalogId %>" />
</jsp:include>
```

9.1.3.4 Rewriting JSPs for caching

If you examine the contents of the JSPs, you will notice that the logic to determine the user's information exists in all three JSPs. This information is needed every time when the *StoreCatalogDisplay* command is executed to determine what the page output should be. Therefore these JSP should be rewritten as follows:

1. Create a new JSP *CacheParametersSetup.jsp* which will be used to set the user's information into different variables. These variables can then be used as parameters to invoke the different variations of the JSP. The user information will be the following:
 - *userState* (String)
 - *userType* (String)
 - *bBuyerApprover* (boolean)
 - *bBuyerAdmin* (boolean)
 - *bBuyerBuySide* (boolean)
 - *rfqLinkDisplayed* (boolean)
 - *displayApproverLink* (boolean)
 - *bRegisteredUser* (boolean)
 - *bGuestUser* (boolean)
2. Create three new JSPs based on the original JSPs

- *CachedHeaderDisplay.jsp*
- *CachedSidebarDisplay.jsp*
- *CachedStoreCatalogDisplay.jsp*

Remove the logic to get the user's information and change the code to use the input parameters instead.

3. Modify *StoreCatalogDisplay.jsp* to include the new JSPs that were created in steps 1 and 2. *CacheParametersSetup.jsp* should be statically included and the rest should be dynamically included.

For example,

```
<%@ include file="../../include/CacheParametersSetup.jsp"%>
<%
    String incfile;
    incfile = includeDir + "CachedHeaderDisplay.jsp";
%>
<jsp:include page="<%=incfile%>" flush="true">
    <jsp:param name="storeId" value="<%= storeId %>" />
    <jsp:param name="catalogId" value="<%= catalogId %>" />
    <jsp:param name="langId" value="<%= languageId %>" />
    <jsp:param name="userType" value="<%= userType %>" />
    <jsp:param name="liveHelp" value="<%= liveHelp %>" />
    <jsp:param name="rfqLinkDisplayed" value="<%= rfqLinkDisplayed
%>" />
</jsp:include>
```

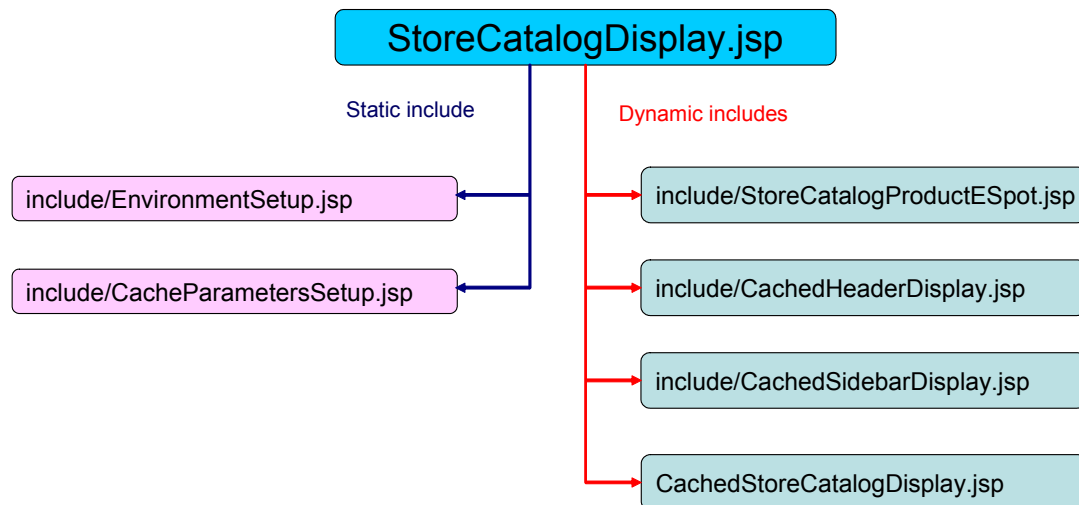


Figure 31 . StoreCatalogDisplay.jsp includes

Note:

For details of the JSPs, please refer to *StoreCatalogDisplay.jsp*, *CacheParametersSetup.jsp*, *CachedHeaderDisplay.jsp*,

CachedSidebarDisplay.jsp and *CachedStoreCatalogDisplay.jsp* in [Sample JSPs](#) section of this document.

9.1.3.5 Defining cache entry and cache ID

Now we have finished changing the JSPs, we can define cache entries to get these JSPs cached. *StoreCatalogDisplay.jsp* should not be cached since it contains logic to determine the user's information. So the pages we should be caching are:

- *CachedHeaderDisplay.jsp*
- *CachedSidebarDisplay.jsp*
- *CachedStoreCatalogDisplay.jsp*

9.1.3.5.1 Cache entry for *CachedHeaderDisplay.jsp*

For the *CachedHeaderDisplay.jsp*, since the output is based on the JSP parameters, the *cache-entry* should be set up as follows:

```
<cache-entry>
  <class>servlet</class>
  <name>/ToolTech/include/CachedHeaderDisplay.jsp</name>
  <property name="save-attributes">false</property>

  <cache-id>
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
    <component id="userType" type="parameter">
      <required>true</required>
    </component>
    <component id="liveHelp" type="parameter">
      <required>true</required>
    </component>
    <component id="rfqLinkDisplayed" type="parameter">
      <required>true</required>
    </component>
    <component id="DC_lang" type="attribute">
      <required>true</required>
    </component>
  </cache-id>
</cache-entry>
```

The request attribute *DC_lang* is defined as part of the cache ID because the ToolTech store supports multiple languages. However, we cannot use the parameter *langId* because it is not a mandatory parameter.

9.1.3.5.2 Cache entries for CachedSidebarDisplay.jsp and CachedStoreCatalogDisplay.jsp

Similarly cache entries can be defined for both *CachedSidebarDisplay.jsp* and *CachedStoreCatalogDisplay.jsp*:

```
<cache-entry>
  <class>servlet</class>
  <name>/ToolTech/include/CachedSidebarDisplay.jsp</name>
  <property name="save-attributes">false</property>

  <cache-id>
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
    <component id="userType" type="parameter">
      <required>true</required>
    </component>
    <component id="userState" type="parameter">
      <required>true</required>
    </component>
    <component id="liveHelp" type="parameter">
      <required>true</required>
    </component>
    <component id="DC_lang" type="attribute">
      <required>true</required>
    </component>
  </cache-id>
</cache-entry>

<cache-entry>
  <class>servlet</class>
  <name>/ToolTech/ShoppingArea/CatalogSecion/CategorySubsection/CachedStoreCatalogDisplay.jsp</name>
  <property name="save-attributes">false</property>

  <cache-id>
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
    <component id="displayApproverLink" type="parameter">
      <required>true</required>
    </component>
  </cache-id>
</cache-entry>
```

```

        </component>
        <component id="DC_lang" type="attribute">
            <required>true</required>
        </component>
    </cache-id>
</cache-entry>

```

9.1.3.6 Defining dependency ID

After the cache entries are defined, then we need to define dependency IDs for the entries. Dependency ID rules should be defined with cache invalidation in mind. That is, consider what could cause the cached JSP content to become invalid and which JSPs need to be invalidated as a result. Examples of events that could cause cache content to become invalid include:

- the store is opened, closed, suspended, etc.
- new entries are added, removed or updated in the catalog
- contracts are deployed, suspended, resumed or canceled

For example:

If the store administrator wants all the cached JSPs to be invalidated when the store is closed, then all the JSPs would need to have a grouping ID (dependency ID). When the command *com.ibm.commerce.store.commands.StoreCloseCmdImpl* is invoked, it will invalidate all the cache entries associated with that dependency ID.

The logical dependency ID to be created in this case would be:

```

<dependency-id>storeId
    <component id="storeId" type="parameter">
        <required>true</required>
    </component>
</dependency-id>

```

Similar dependency IDs can be created for catalog information, like the catalog ID, category ID and product ID, etc.

Now let us look at contract. For the default ToolTech store, there are four contracts deployed when the store is published, two for each organization. In this example, the user can only be eligible for a maximum two contracts. The dependency IDs are created as follows with *DC_cont0* and *DC_cont1* as the attribute IDs.

```

<dependency-id>contracts
    <component id="DC_cont" type="attribute" ignore-value="true">
        <required>true</required>
    </component>
</dependency-id>

```

```

<dependency-id>contractId
  <component id="DC_conto" type="attribute">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>contractId
  <component id="DC_cont1" type="attribute">
    <required>true</required>
  </component>
</dependency-id>

```

9.1.4 Displaying cache information

After defining the cache entries, we can use the *cache monitor* to:

- View the cache policies information

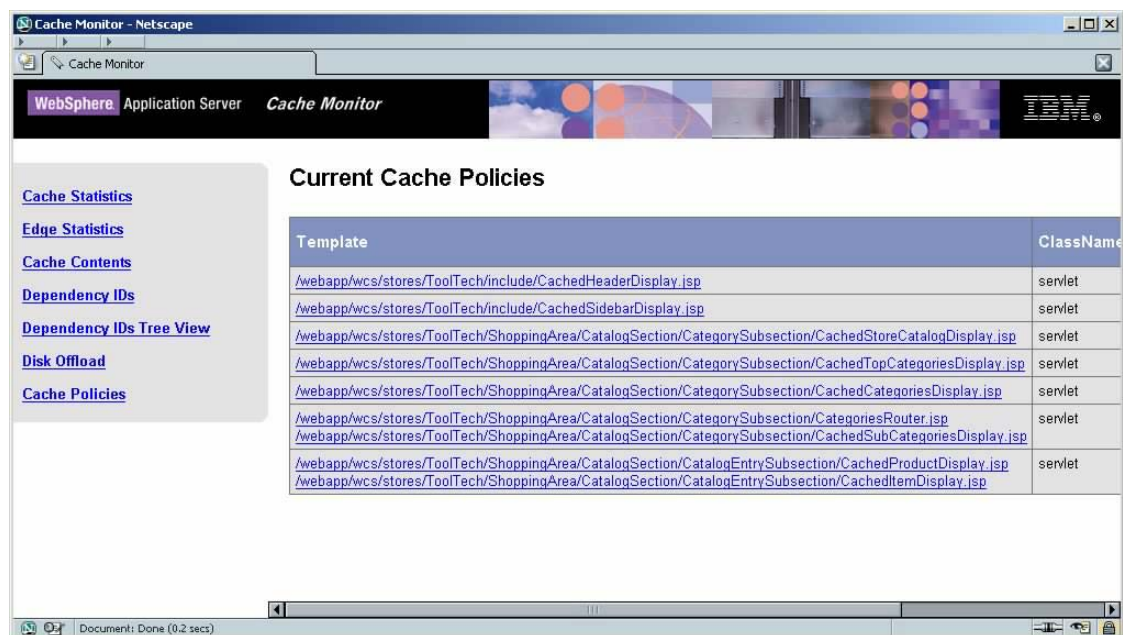


Figure 32. Cache policies as viewed by cache monitor

Cache Monitor - Netscape

Cache Monitor

WebSphere Application Server Cache Monitor

Cache Policy Detail for /webapp/wcs/stores/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/CachedStoreCatalogDisplay.jsp

Cache ID Rule: 0 timeout: 0 priority: 0 idGenerator: null m+toDataGenerator: null properties: 0

Component	ID	Type	Ignore Value	Method	Required	Values	Not-Values
Component 0	storeId	parameter	true	all	true		
Component 1	catalogId	parameter	true	all	true		
Component 2	displayProcessorLink	parameter	true	all	true		
Component 3	DC_tag	attribute	true	all	true		
Component 4	DC_catr	attribute	true	all	true		
Component 5	DC_posg	attribute	true	all	true		
Component 6	DC_posr	attribute	true	all	true		
Component 7	DC_posg	attribute	true	all	true		

Dependency: ID Rule 0 Base Name: storeId

Component	ID	Type	Ignore Value	Method	Required	Values	Not-Values
Component 0	storeId	parameter	true	all	true		

Dependency: ID Rule 1 Base Name: catalogId

Component	ID	Type	Ignore Value	Method	Required	Values	Not-Values
Component 0	catalogId	parameter	true	all	true		

Dependency: ID Rule 2 Base Name: storeId/catalogId

Component	ID	Type	Ignore Value	Method	Required	Values	Not-Values
Component 0	storeId	parameter	true	all	true		
Component 1	catalogId	parameter	true	all	true		

Dependency: ID Rule 3 Base Name: StoreCatalogDisplay;storeId

Component	ID	Type	Ignore Value	Method	Required	Values	Not-Values
Component 0	storeId	parameter	true	all	true		

Document: Done (0.441 secs)

Figure 33. Cache policy for CachedStoreCatalogDisplay.jsp

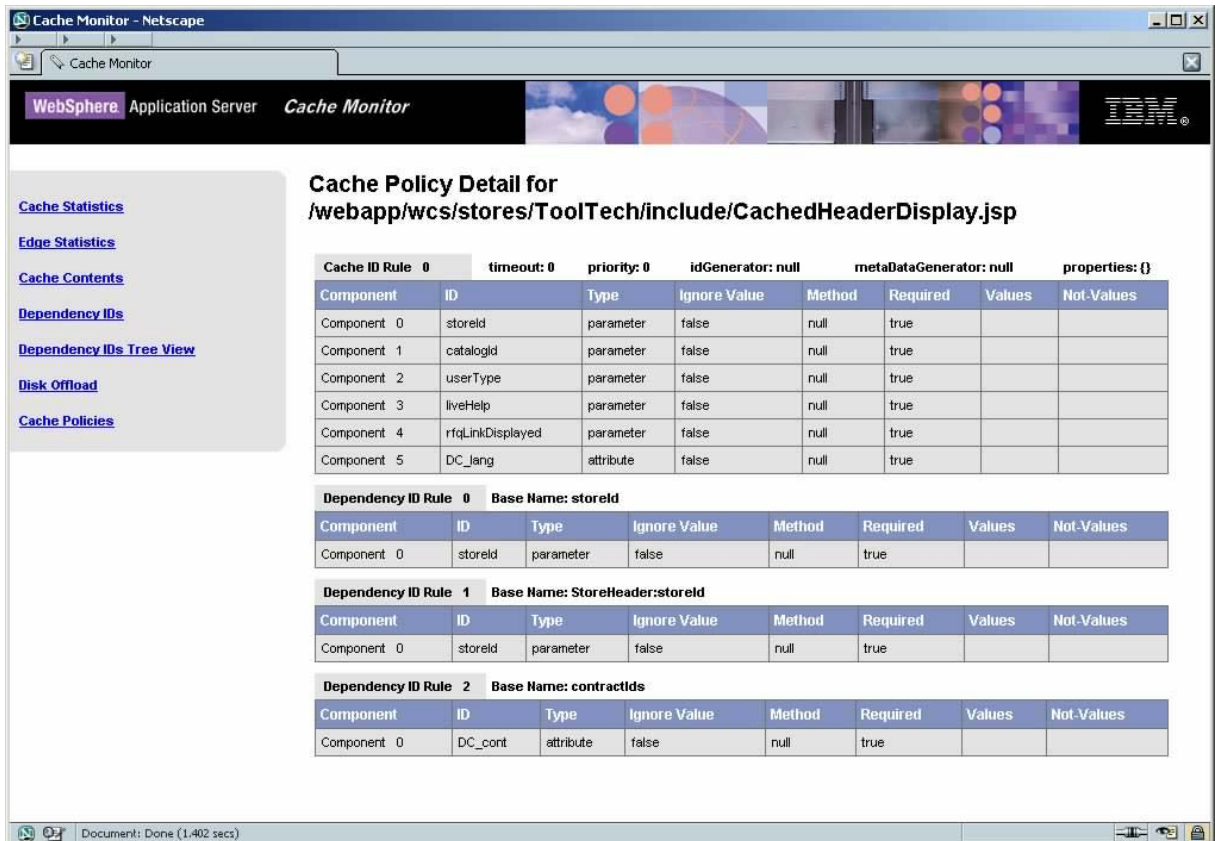


Figure 34. Cache Policy for CachedHeaderDisplay.jsp

- View the cache content

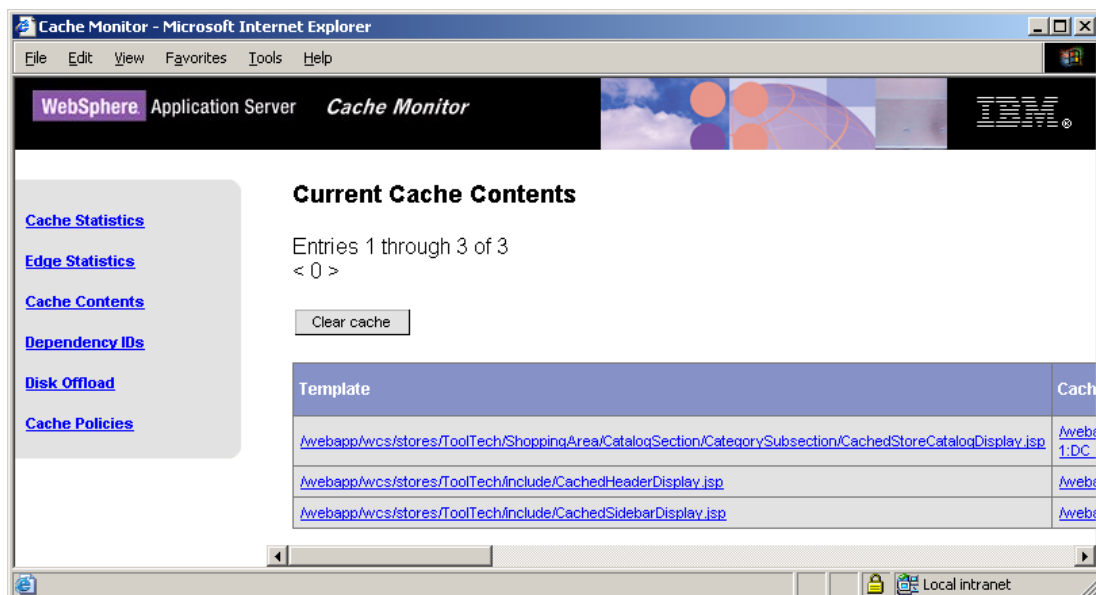


Figure 35. Cache contents as viewed by cache monitor

Cache
Entry: /webapp/wcs/stores/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/CachedStoreCatalogDisplay.jsp?storeId=10051:catalogId=10051:displayApproverLink=false:1:DC_curr=USD:DC_porg=2001:DC_cont=10004:DC_mg=999:IS08859.1



Figure 36. Cache entry for CachedStoreCatalogDisplay.jsp

The following is the sample *cachespec.xml* file provided for the ToolTech store located at *WCInstallDir/samples/dynacache/B2BDirect/cachespec.xml*:

```
<?xml version="1.0" ?>
<!DOCTYPE cache SYSTEM "cachespec.dtd">
<cache>

<!-- Starts B2BDirect Store Pages -->
<cache-entry>
  <class> servlet </class>
  <name> /ToolTech/include/CachedHeaderDisplay.jsp </name>
  <property name="save-attributes"> false </property>

  <cache-id>
    <component id="storeId" type="parameter">
      <required> true </required>
    </component>
    <component id="catalogId" type="parameter">
      <required> true </required>
    </component>
    <component id="userType" type="parameter">
      <required> true </required>
    </component>
    <component id="liveHelp" type="parameter">
      <required> true </required>
    </component>
    <component id="rfqLinkDisplayed" type="parameter">
      <required> true </required>
    </component>
    <component id="DC_lang" type="attribute">
      <required> true </required>
    </component>
  </cache-id>
</cache-entry>
</cache>
```

```

    </component>
</cache-id>

<!-- Starts Dependency Ids -->
<dependency-id>storeId
    <component id="storeId" type="parameter">
        <required>true</required>
    </component>
</dependency-id>

<dependency-id>StoreHeader:storeId
    <component id="storeId" type="parameter">
        <required>true</required>
    </component>
</dependency-id>

<dependency-id>contractIds
    <component id="DC_cont" type="attribute">
        <required>true</required>
    </component>
</dependency-id>
<!-- Ends Dependency Ids -->
</cache-entry>

<cache-entry>
    <class>servlet</class>
    <name>/ToolTech/include/CachedSidebarDisplay.jsp</name>
    <property name="save-attributes">>false</property>

    <cache-id>
        <component id="storeId" type="parameter">
            <required>true</required>
        </component>
        <component id="catalogId" type="parameter">
            <required>true</required>
        </component>
        <component id="userType" type="parameter">
            <required>true</required>
        </component>
        <component id="userState" type="parameter">
            <required>true</required>
        </component>
        <component id="liveHelp" type="parameter">
            <required>true</required>
        </component>
        <component id="DC_lang" type="attribute">
            <required>true</required>
        </component>
    </cache-id>

    <!-- Starts Dependency Ids -->
    <dependency-id>storeId
        <component id="storeId" type="parameter">

```

```

        <required>true</required>
    </component>
</dependency-id>

<dependency-id>catalogId
    <component id="catalogId" type="parameter">
        <required>true</required>
    </component>
</dependency-id>

<dependency-id>StoreSidebar:storeId
    <component id="storeId" type="parameter">
        <required>true</required>
    </component>
</dependency-id>
<!-- Ends Dependency Ids -->
</cache-entry>

<cache-entry>
    <class>servlet</class>
    <name>/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/CachedStoreCatalogDispl
ay.jsp</name>
    <property name="save-attributes">>false</property>

    <cache-id>
        <component id="storeId" type="parameter">
            <required>true</required>
        </component>
        <component id="catalogId" type="parameter">
            <required>true</required>
        </component>
        <component id="displayApproverLink" type="parameter">
            <required>true</required>
        </component>
        <component id="DC_lang" type="attribute">
            <required>true</required>
        </component>
        <component id="DC_curr" type="attribute">
            <required>true</required>
        </component>
        <component id="DC_porg" type="attribute">
            <required>true</required>
        </component>
        <component id="DC_cont" type="attribute">
            <required>true</required>
        </component>
        <component id="DC_mg" type="attribute">
            <required>true</required>
        </component>
    </cache-id>

    <!-- Starts Dependency Ids -->
    <dependency-id>storeId

```

```

    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>

  <dependency-id>catalogId
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>

  <dependency-id>storeId:catalogId
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>

  <dependency-id>StoreCatalogDisplay:storeId
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>

  <dependency-id>storeId:contracts
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="DC_cont" type="attribute" ignore-value="true">
      <required>true</required>
    </component>
  </dependency-id>

  <dependency-id>contracts
    <component id="DC_cont" type="attribute" ignore-value="true">
      <required>true</required>
    </component>
  </dependency-id>

  <dependency-id>contractId
    <component id="DC_cont0" type="attribute">
      <required>true</required>
    </component>
  </dependency-id>

  <dependency-id>contractId
    <component id="DC_cont1" type="attribute">
      <required>true</required>
    </component>
  </dependency-id>
<!-- Ends Dependency Ids -->

```

```

</cache-entry>

<cache-entry>
  <class> servlet</class>
  <name> /ToolTech/ShoppingArea/CatalogSection/CategorySubsection/CachedTopCategoriesDisplay.jsp</name>
  <property name=" save-attributes"> false</property>

  <cache-id>
    <component id=" storeId" type=" parameter">
      <required> true</required>
    </component>
    <component id=" catalogId" type=" parameter">
      <required> true</required>
    </component>
    <component id=" DC_lang" type=" attribute">
      <required> true</required>
    </component>
    <component id=" DC_curr" type=" attribute">
      <required> true</required>
    </component>
    <component id=" DC_porg" type=" attribute">
      <required> true</required>
    </component>
    <component id=" DC_cont" type=" attribute">
      <required> true</required>
    </component>
    <component id=" DC_mg" type=" attribute">
      <required> true</required>
    </component>
  </cache-id>

  <!-- Starts Dependency Ids -->
  <dependency-id> storeId
    <component id=" storeId" type=" parameter">
      <required> true</required>
    </component>
  </dependency-id>

  <dependency-id> catalogId
    <component id=" catalogId" type=" parameter">
      <required> true</required>
    </component>
  </dependency-id>

  <dependency-id> storeId:catalogId
    <component id=" storeId" type=" parameter">
      <required> true</required>
    </component>
    <component id=" catalogId" type=" parameter">
      <required> true</required>
    </component>
  </dependency-id>

```

```

<dependency-id>TopCategoriesDisplay:storeId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>TopCategoriesDisplay:storeId:catalogId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="catalogId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>TopCategoriesDisplay:storeId:categoryId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="categoryId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>storeId:contracts
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="DC_cont" type="attribute" ignore-value="true">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>contracts
  <component id="DC_cont" type="attribute" ignore-value="true">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>contractId
  <component id="DC_cont0" type="attribute">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>contractId
  <component id="DC_cont1" type="attribute">
    <required>true</required>
  </component>
</dependency-id>
<!-- Ends Dependency Ids -->

```



```

</cache-entry>

<cache-entry>
  <class> servlet</class>
  <name> /ToolTech/ShoppingArea/CatalogSection/CategorySubsection/CachedSubCategoriesDisplay.jsp</name>
  <name> /ToolTech/ShoppingArea/CatalogSection/CategorySubsection/CategoriesRouter.jsp</name>
  <property name=" save-attributes"> false</property>

  <cache-id>
    <component id=" storeId" type=" parameter">
      <required> true</required>
    </component>
    <component id=" catalogId" type=" parameter">
      <required> true</required>
    </component>
    <component id=" categoryId" type=" parameter">
      <required> true</required>
    </component>
    <component id=" parent_category_rr" type=" parameter">
      <required> false</required>
    </component>
    <component id=" DC_lang" type=" attribute">
      <required> true</required>
    </component>
    <component id=" DC_curr" type=" attribute">
      <required> true</required>
    </component>
    <component id=" DC_porg" type=" attribute">
      <required> true</required>
    </component>
    <component id=" DC_cont" type=" attribute">
      <required> true</required>
    </component>
    <component id=" DC_mg" type=" attribute">
      <required> true</required>
    </component>
  </cache-id>

  <!-- Starts Dependency Ids -->
  <dependency-id> storeId
    <component id=" storeId" type=" parameter">
      <required> true</required>
    </component>
  </dependency-id>

  <dependency-id> catalogId
    <component id=" catalogId" type=" parameter">
      <required> true</required>
    </component>
  </dependency-id>

```

```

<dependency-id>categoryId
  <component id="categoryId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>storeId:catalogId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="catalogId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>storeId:categoryId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="categoryId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>CategoryDisplay:storeId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>CategoryDisplay:storeId:catalogId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="catalogId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>CategoryDisplay:storeId:categoryId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="categoryId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>storeId:contracts
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="DC_cont" type="attribute" ignore-value="true">

```

```

        <required>true</required>
    </component>
</dependency-id>

<dependency-id>contracts
    <component id="DC_cont" type="attribute" ignore-value="true">
        <required>true</required>
    </component>
</dependency-id>

<dependency-id>contractId
    <component id="DC_cont0" type="attribute">
        <required>true</required>
    </component>
</dependency-id>

<dependency-id>contractId
    <component id="DC_cont1" type="attribute">
        <required>true</required>
    </component>
</dependency-id>
<!-- Ends Dependency Ids -->
</cache-entry>

<cache-entry>
    <class>servlet</class>
    <name>/ToolTech/ShoppingArea/CatalogSection/CatalogEntrySubsection/CachedProductDisplay.jsp</name>
    <name>/ToolTech/ShoppingArea/CatalogSection/CatalogEntrySubsection/CachedItemDisplay.jsp</name>
    <property name="save-attributes">false</property>

    <cache-id>
        <component id="storeId" type="parameter">
            <required>true</required>
        </component>
        <component id="catalogId" type="parameter">
            <required>true</required>
        </component>
        <component id="productId" type="parameter">
            <required>true</required>
        </component>
        <component id="userState" type="parameter">
            <required>false</required>
        </component>
        <component id="registeredUser" type="parameter">
            <required>false</required>
        </component>
        <component id="rfqLinkDisplayed" type="parameter">
            <required>false</required>
        </component>
        <component id="DC_lang" type="attribute">
            <required>true</required>

```

```

</component>
<component id="DC_curr" type="attribute">
  <required>true</required>
</component>
<component id="DC_porg" type="attribute">
  <required>true</required>
</component>
<component id="DC_cont" type="attribute">
  <required>true</required>
</component>
<component id="DC_mg" type="attribute">
  <required>true</required>
</component>
</cache-id>

<!-- Starts Dependency Ids -->
<dependency-id>storeId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>catalogId
  <component id="catalogId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>productId
  <component id="productId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>storeId:catalogId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="catalogId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>productId:catalogId
  <component id="productId" type="parameter">
    <required>true</required>
  </component>
  <component id="catalogId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<dependency-id>ProductDisplay:storeId

```

```

    <component id="storeId" type="parameter">
      <required> true</required>
    </component>
  </dependency-id>

  <dependency-id>ProductDisplay:storeId:catalogId
    <component id="storeId" type="parameter">
      <required> true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required> true</required>
    </component>
  </dependency-id>

  <dependency-id>ProductDisplay:storeId:productId
    <component id="storeId" type="parameter">
      <required> true</required>
    </component>
    <component id="productId" type="parameter">
      <required> true</required>
    </component>
  </dependency-id>

  <dependency-id>storeId:contracts
    <component id="storeId" type="parameter">
      <required> true</required>
    </component>
    <component id="DC_cont" type="attribute" ignore-value="true">
      <required> true</required>
    </component>
  </dependency-id>

  <dependency-id>contracts
    <component id="DC_cont" type="attribute" ignore-value="true">
      <required> true</required>
    </component>
  </dependency-id>

  <dependency-id>contractId
    <component id="DC_cont0" type="attribute">
      <required> true</required>
    </component>
  </dependency-id>

  <dependency-id>contractId
    <component id="DC_cont1" type="attribute">
      <required> true</required>
    </component>
  </dependency-id>
  <!-- Ends Dependency Ids -->
</cache-entry>
<!-- Ends B2BDirect Store Pages -->
</cache>

```


The cache ID and dependency IDs generated for the URL

<https://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?langId=-1&storeId=10051&catalogId=10051> are:

JSP	Cache ID	Dependency ID
/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/CachedStoreCatalogDisplay.jsp	webapp/wcs/stores/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/CachedStoreCatalogDisplay.jsp:storeId=10051:catalogId=10051:displayApproverLink=false:DC_lang=-1:DC_curr=USD:DC_porg=70000000000000000052:DC_cont=10004;10005:DC_mg=0:ISO8859-1	<ul style="list-style-type: none"> • storeId:10051 • catalogId:10051 • storeId:catalogId:10051:10051 • storeId:contracts:10051 • contracts • contractId:10005 • ContractId:10004 • StoreCatalogDisplay:storeId:10051
/ToolTech/include/CacheHeaderDisplay.jsp	/webapp/wcs/stores/ToolTech/include/CachedHeaderDisplay.jsp:storeId=10051:catalogId=10051:userType=R:liveHelp=false:rfqLinkDisplayed=false:DC_lang=-1:ISO8859-1	<ul style="list-style-type: none"> • storeId:10051 • StoreHeader:storeId:10051 • contracts
/ToolTech/include/CacheSidebarDisplay.jsp	/webapp/wcs/stores/ToolTech/include/CachedSidebarDisplay.jsp:storeId=10051:catalogId=10051:userType=R:userState=1:liveHelp=false:DC_lang=-1:ISO8859-1	<ul style="list-style-type: none"> • storeId:10051 • StoreSideBar:storeId:10051 • catalogId:10051

10.1 Cache invalidation

WebSphere Application Server dynamic cache provides the following types of invalidation mechanisms:

- Rule-based
- Time-based
- Group-based
- Programmatic

10.1.1 Rule-based

Two types of invalidation rules can be specified in the *cachespec.xml* file, namely, servlet-based and command-based invalidation rules. They allow specific defined events to trigger the invalidation automatically.

10.1.2 Servlet-based invalidation

Servlet-based invalidation means that invalidation rules are triggered when the servlet passes through its service method. Since WebSphere Commerce only supplies a single servlet (*com.ibm.commerce.server.RequestServlet*), the *pathinfo* component must be used as a filtering mechanism to catch a specific URI call.

Example of servlet-based invalidation:

```
<cache-entry>
  <class>servlet</class>
  <name>com.ibm.commerce.server.RequestServlet.class</name>
  <invalidation>listId
    <component id="" type="pathinfo" ignore-value="true">
      <required>true</required>
      <value>/InterestItemDelete</value>
    </component>
    <component id="listId" type="parameter">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>
```

The example above specifies a policy to invalidate any cache entries that can be identified by the same *listId*. It intercepts calls to the *InterestItemDelete* URI which deletes catalog entries from an interest item list and generates the invalidation ID based on the *listId* parameter value provided to the URI. Upon execution of the servlet, dynamic cache compares the invalidation IDs with each of the dependency IDs for the same identifier and value. Any cache entries that are associated with such dependency IDs are removed.

10.1.3 Command-based invalidation

Command-based invalidation means that the invalidation rules are executed upon execution of a command which extends from the WebSphere Commerce Command Framework API. Invalidation IDs for command-based invalidation are constructed based on methods and fields provided by the commands.

Example of command-based invalidation:

```
<cache-entry>
  <class>command</class>
  <name>com.ibm.commerce.catalogmanagement.commands.CatalogEntryUpdateCmdImpl</name>

  <invalidation>productId
    <component id="getCatentryId" type="method">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>
```

The example above specifies a policy to invalidate any cache entries that can be identified by the same productId. It intercepts calls to the CatalogEntryUpdate command which update a catalog entry and generates the invalidation ID based on the value returned by the getCatentryId method. Upon execution of the servlet, dynamic cache compares the invalidation IDs with each of the dependency IDs for the same identifier and value. Any cache entries associated with such dependency IDs are removed.

When building the command-based invalidation policies in *cachespec.xml*, keep in mind the following restrictions:

- Only the methods invoked by the command that return the input instance variables can be used in the method component.
- All the methods that are used to construct the invalidation IDs, must be provided in the command interface and be implemented.
- The methods to be used in the method component must only return single value.
- The request attributes component type cannot be used.

10.1.4 Time-based invalidation

Time-based invalidation can be accomplished by specifying the `<timeout>value</timeout>` sub-element with a *cache-entry* in the *cachespec.xml* file. *Value* is the amount of time, in seconds, the cache entry is kept in the cache. This method is useful when the cache entries cannot be invalidated by any other mechanism, or they should be refreshed after a set period.

An example in which time-based invalidation makes sense to use is caching of a e-Marketing spot. In general e-Marketing spots should not be cached because the page output is generated dynamically based on personalized data. But in the case

that the store admin is willing to sacrifice function for performance, then the e-Marketing spot JSPs can be cached with a *timeout* sub-element, so that the output can be reused for a certain period of time.

10.1.5 Group-based invalidation

You can specify additional cache group identifiers to associate multiple cache entries the same group identifier in *cachespec.xml*. There is no limit on the number of such identifiers -- dependency IDs, that can be defined in a cache entry. That is, you can define more than one dependency ID on the same cache entry and the same dependency identifier can be reused on another cache entry. This mechanism provides a convenient way to remove all related cache entries at the same time by means of a single rule.

The following shows an example in which the same dependency ID and storeId, has been defined on each of the catalog page cache entries. Also, at the end, an invalidation rule is provided in which the invalidation ID is generated in a way that maps to that of the dependency ID when the rule intercepts the command call to StoreStyleUpdateCmd. At execution time, after dynamic cache generates the invalidation ID, it will compare the ID to each of the dependency IDs for the same identifier and value from the cache entries. All catalog page cache entries that are grouped under the dependency ID will be removed.

Example of group-based invalidation:

```
<cache-entry>
  <class>servlet</class>
  <name>/Tooltech/ShoppingArea/CatalogSection/CategorySubsection
/CachedStoreCatalogDisplay.jsp</name>
  ...
  <dependency-id>storeId
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>
</cache-entry>

<cache-entry>
  <class>servlet</class>
  <name>/Tooltech/ShoppingArea/CatalogSection/CategorySubsection
/CachedTopCategoriesDisplay.jsp</name>
  ...
  <dependency-id>storeId
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>
</cache-entry>
```

```

<cache-entry>
  <class>servlet</class>
  <name>/Tooltech/ShoppingArea/CatalogSection/CategorySubsection
/CachedCategoryDisplay.jsp</name>
  <name>/Tooltech/ShoppingArea/CatalogSection/CategorySubsection
/CategoriesRouter.jsp</name>
  ...
  <dependency-id>storeId
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>
</cache-entry>

<cache-entry>
  <class>servlet</class>
  <name>/Tooltech/ShoppingArea/CatalogSection/CategorySubsection
/CachedProductDisplay.jsp</name>
  <name>/Tooltech/ShoppingArea/CatalogSection/CategorySubsection
/CachedItemDisplay.jsp</name>
  ...
  <dependency-id>storeId
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>
</cache-entry>

<cache-entry>
  <class>command</class>
  <name>com.ibm.commerce.tools.devtools.store.commands.StoreStyleUpdateCmdImpl</name>
  ...
  <invalidation>storeId
    <component id="getStoreId" type="method">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>

```

10.1.6 Programmatic invalidation

When updates are made directly to the database, or in more complex scenarios that cause changes to the cache entries, you might need to write custom applications to invalidate the cache entries. WebSphere Application Server dynamic cache provides the following APIs to support programmatic invalidation:

- `com.ibm.websphere.cache.invalidateById`
- `com.ibm.websphere.cache.invalidateByTemplate`

10.1.6.1 Invalidation through the cache monitor

Manual invalidation of the cache entries can be accomplished through the cache monitor tool. Cache entries can be invalidated selectively, or as a bundle grouped by dependency IDs or the entire cache can be cleared. One application of this method is that it allows users to remove unwanted cache entries immediately in emergency case.

10.1.6.2 DynaCacheInvalidation command

WebSphere Commerce provides a *DynaCacheInvalidation* command which is called by the scheduler periodically to process the records in the *CACHEIVL* table and then call the WebSphere Application Server dynamic cache invalidation APIs to invalidate the specified cache entries. By default, the schedule interval is every ten minutes. Refer to the WebSphere Commerce's documentation on *Scheduler* in the *WebSphere Commerce's Administration Guide* on how to edit, delete and schedule a new job.

Syntax:

`http://hostname/webapp/wcs/stores/servlet/DynaCacheInvalidation`

Access control:

Site and Store Administrator

10.1.6.3 CACHEIVL table

The *CACHEIVL* table is processed by the *DynaCacheInvalidation* command and the rules are :

- If the *TEMPLATE* column is set then the *DynaCacheInvalidation* command will call the **`com.ibm.websphere.cache.invalidateByTemplate`** API and use the *TEMPLATE* column content as the parameter
- If the *DATAID* column is set and the *TEMPLATE* column is not set, then the *DynaCacheInvalidation* command will call the **`com.ibm.websphere.cache.invalidateById`** API using the *DATAID* column content as the parameter

10.1.6.4 Database Triggers

The *CACHEIVL* table can also be populated by database triggers, a sample of the triggers can be found in the `WCInstallDir/samples/dynacache/triggers/dbType/cacheTriggers.sql`. The triggers can be created as follows:

DB2:

1. db2 connect to *db_name* user *db_username* using *db_password*

2. db2 -vtd# -f *cacheTriggers.sql*

Oracle:

1. Open a SQLPlus window
2. Connect as *dba*
3. Type *@cacheTriggers.sql*

With these sample database triggers and if the *STORECAT* table is updated, the *CACHEIVL* table will contains the followings:

```
db2 select substr(dataid, 1, 40), inserttime from cacheivl
```

1	INSERTTIME
-----	-----
storeId:10001	2003-06-02-11.50.50.921000
catalogId:10001	2003-06-02-11.50.50.921000
storeId:catalogId:10001:10001	2003-06-02-11.50.50.921000

3 record(s) selected.

So when the DynaCacheInvalidation command is being executed by the scheduler, it will call the **com.ibm.websphere.cache.invalidateById** API and with the data Ids *storeId:10001*, *catalogId:10001* and *storeId:catalogId:10001:10001*.

10.1.7 Building invalidation policies in WebSphere Commerce

Invalidation rules can be built in different ways using a combination of the available invalidation mechanisms. Sometimes compromise is required when choosing between the mechanisms due to limitation of one mechanism or to business requirement.

10.1.7.1 Invalidation rules in *cachespec.xml*

Invalidation rules can be specified independently of cache ID generation. That is, all invalidation rules separately executes one by one even if the cache ID is not generated.

10.1.7.2 Servlet-based versus command-based invalidation

In WebSphere Commerce, a URI servlet call may trigger one or multiple command calls. By defining the invalidation policies at the individual command level instead of on the URI, it allows the users to refine the rules so that more granular and precise cache invalidation can be performed upon the execution of a command. On the other hand, since majority of business logic that might cause changes to cache entries are modularized either inside the controller or the task commands, users should be able to find out quickly what impact a command is on the current cache entries by examining the logic in the command.

While Command-based invalidation provides a lot of advantages, its support is limited with WebSphere Application Server V5.

For example, the method component does not support return of an array from the method nor accept any parameters input to the method. Also unlike Servlet-based invalidation, the request attributes are not available to Command-based invalidation. If a user attempts to construct a invalidation Id with the request attribute like the following example, it will cause dynamic cache to issue error of

DYNA0038E: Unrecognized component type attribute for command...

The following shows an example which is not allowed:

```
<cache-entry>
  <class>command</class>
  <name>com.ibm.commerce.catalogmanagement.commands.CatalogEntryUpdateCmdImpl</name>

  <invalidation>storeId
    <component id="DC_storeId" type="attribute">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>
```

10.1.7.3 Building dependency IDs for invalidation

In order to invalidate the cache entries, the task is not merely to set up the invalidation rules, it also involves building proper dependency ids on the cache entries so that the invalidation can take place feasibly and efficiently.

10.1.7.3.1 Dependency IDs

Dependency IDs cannot be specified independently of cache ID generation. That is if a cache ID is not successfully generated, all the associated dependency Ids are not generated.

A dependency ID is basically composed of two parts:

- Identifier base name
- Identifier value

The identifier base name is a constant string that can be made up of one or more string tokens. The string tokens can be separated by a special characters e.g. a colon. It identifies a dependency ID rule. In order for the invalidation rule to invalidate the cache entries through the dependency IDs, the identifier defined on the invalidation rules must first match that of the dependency Ids.

There are many ways to define the dependency IDs for invalidation use. Here are several examples that can be applied in WebSphere Commerce:

Example 1:

Generate the ID with a specific component value (e.g. contractId)

```
<cache-entry>
...
<dependency-id>contractId
  <component id="DC_cont" type="attribute">
    <required>true</required>
  </component>
</dependency-id>
</cache-entry>
```

The above example is useful when you need to invalidate only the cache entries under a specific contract ID.

Example 2:

Generate the ID with a generic name (e.g. Contracts) that only takes into account of the component id, but not the component value .

```
<cache-entry>
...
<dependency-id>contracts
  <component id="DC_cont" type="attribute"
    ignore-value="true">
    <required>true</required>
  </component>
</dependency-id>
</cache-entry>
```

The above example is useful when it needs to invalidate all the cache entries related to any contracts.

Example 3:

A combination of examples 1 and 2

```
<cache-entry>
...
<dependency-id>storeId:contracts
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="DC_cont" type="attribute"
    ignore-value="true">
    <required>true</required>
  </component>
</dependency-id>
</cache-entry>
```

The above example is useful when it needs to invalidate all the cache entries related to any contracts under a specific store.

10.1.7.3.2 Multiple dependency IDs

Multiple events can cause a cache entry to change under different scenarios. For example, when a contract is updated in a store, it might cause the contract prices be changed on the products and therefore the amount that were displayed on the cached product pages will become invalid. Similarly, any change to the product description or to the look of the storefront might as well changes the content of the product page. In both scenarios, if the product display page is being cached, their cache entries should be invalidated. One approach to invalidate those cache entries is to set up multiple dependency IDs.

Since dynamic cache allows us to define multiple dependency IDs along with a cache entry ID for each cache entry, using the example from above, it means that users can define one set of dependency IDs for the contract update event and another set for the product description update on the same product display cache entry. Then for each event, construct one or more invalidation rules to intercept the execution of the event and define the invalidation identifiers that maps to the corresponding dependency ID. At runtime, the ID generated by the invalidation rule will invalidate all the cache entries that have the same dependency IDs.

The following sample *cachespec.xml* illustrates the above example:

Example:

```
<cache-entry>
  <class>servlet</servlet>
  <name>/ToolTech/ShoppingArea/CatalogSection/CategoryEntrySubsection/CachedProductDisplay.jsp</name>
  . . .
  <!-- ***** -->
  <!-- Dependency ids for Invalidation triggered by store commands -->
  <!-- ***** -->
  <dependency-id>storeId
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
  </dependency-id>

  <!-- ***** -->
  <!-- Dependency ids for Invalidation triggered by product -->
  <!-- commands -->
  <!-- ***** -->
```



```

<dependency-id>productId
  <component id="productId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<!-- ***** -->
<!-- Dependency ids for Invalidation triggered by contract commands
-->
<!-- ***** -->
<dependency-id>storeId:contracts
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="DC_cont" type="attribute"
    ignore-value="true">
    <required>true</required>
  </component>
</dependency-id>
</cache-entry>

<!-- ***** -->
<!-- Invalidate the product page cache entries in a specific store -->
<!-- when the store is updated -->
<!-- ***** -->
<cache-entry>
  <class>command</servlet>
  <name>com.ibm.commerce.tools.devtools.store.commands.StoreTextUpdateCmdImpl</name>
  <name>com.ibm.commerce.tools.devtools.store.commands.StoreLogoUpdateCmdImpl</name>
  <name>com.ibm.commerce.tools.devtools.store.commands.StoreBannerUpdateCmdImpl</name>

  <invalidation>storeId
    <component id="getStoreId" type="method">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>

<!-- ***** -->
<!-- Invalidate the cache entries of a particular product when the -->
<!-- product is updated -->
<!-- ***** -->
<cache-entry>
  <class>command</servlet>

```

```

<name>com.ibm.commerce.catalogmanagement.commands.A
ddCatalogEntryDescCmdImpl</name>
<name>com.ibm.commerce.catalogmanagement.commands.u
pdateCatalogEntryDescCmdImpl</name>

<invalidation>productId
  <component id="getCatentryId" type="method">
    <required>true</required>
  </component>
</invalidation>
</cache-entry>

<!-- ***** -->
<!-- Invalidate the product page cache entries in a specific store -->
<!-- when a contract is imported or deployed to the store -->
<!-- ***** -->
<cache-entry>
  <class>command</servlet>
  <name>com.ibm.commerce.contract.commands.ContractImp
ortApprovedVersionCmdImpl</name>
  <name>com.ibm.commerce.contract.commands.ContractDepl
oyCmdImpl</name>
  <invalidation>storeId:contracts
    <component id="getStoreId" type="method">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>

```

This table summarizes the mapping of the dependency ids and the invalidation rules which intercept the update events that trigger the invalidation of the *CachedProductDisplay.jsp* cache entry.

dependency-id	invalidation rules	events
storeId	store id returned by getStoreId	Update to the StoreFront look
productId	product id returned by getCatentryId	Update to the product information
storeId:contracts	store id returned by getStoreId	Update to the contract used in the store

10.1.7.3.3 Cache grouping

Dependency ID also allows us to logically group the cache entries into one or more than one categories. The benefit of it has been discussed in an earlier section under “Group-based invalidation”. It is important to note that a

cache-entry having the same dependency identifier name as the other cache-entry does not mean that they are logically be grouped.

When a dependency ID is generated, it concatenates the dependency identifier base name with the component elements.

This following shows an example in that two dependency IDs having the same identifier but each of them is comprised of different component elements. While the dependency ID on the left hand side is composed of one component elements: `storeId`, the one on the right hand side is composed of two component elements: `storeId` and `DC_cont`. Therefore they can never be equivalent.

Example:

<pre><dependency-id>storeId:contract <component id="storeId" type="parameter"> <required>true</required> </component> <component id="DC_cont" type="attribute" ignore-value="true"> <required>true</required> </component> </dependency-id></pre>	<pre><dependency-id>storeId:contract <component id="storeId" type="parameter"> <required>true</required> </component> <component id="DC_cont" type="attribute"> <required>true</required> </component> </dependency-id></pre>
--	---

10.1.8 Cache invalidation verification

Users can verify if the invalidation rules that they have defined are actually executed and have invalidated the cache entries as expected by examining the result from cache monitor (remember to refresh browser) and the trace.

If the invalidation rules are executed successfully, the cache entries that were originally seen on the cache monitor and now are supposed get invalidated by the rules, should disappear.

Let's say if we have the following caching and invalidation policies defined in *cachespec.xml*:

```
<cache-entry>
  <class>servlet</class>
  <name>/ToolTech/ShoppingArea/CatalogSection/CatalogEntrySub
section/CachedProductDisplay.jsp</name>
  <name>/ToolTech/ShoppingArea/CatalogSection/CatalogEntrySub
section/CachedItemDisplay.jsp</name>
  <property name="save-attributes">false</property>
```

```

...
<dependency-id>productId
  <component id="productId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>
...
</cache-entry>

<!-- ***** -->
<!-- Invalidate the cache entries of a particular product when -->
<!-- the product is updated -->
<!-- ***** -->
<cache-entry>
  <class>command</servlet>
  <name>com.ibm.commerce.catalogmanagement.commands.
UpdateCatalogEntryDescCmdImpl</name>

  <invalidation>productId
    <component id="getCatentryId" type="method">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>

```

When a cache miss is encountered on the CachedProductDisplay.jsp, the caching policy from the above xml causes the cache entry be generated and be displayed on the cache monitor as follows:

WebSphere Application Server Cache Monitor

[Cache Statistics](#)
[Edge Statistics](#)
[Cache Contents](#)
[Dependency IDs](#)
[Disk Offload](#)
[Cache Policies](#)

Current Cache Contents

Entries 1 through 1 of 1
 < 0 >

Template

</webapp/wcs/stores/ToolTech/ShoppingArea/CatalogSection/CatalogEntrySubsection/ProductDisplay.jsp>



From the cache monitor, you can see that one of the dependency IDs, productId has been defined on the cache entry. When the invalidation rule in the XML is evaluated, the invalidation ID, productId will map to the dependency ID and cause the cache entry be removed from the cache space upon the execution of UpdateCatalogEntryDescCmd. At the end, the entry should disappear from the cache monitor.

If the cache entries are not removed as expected, users can examine the trace. The default location of the trace is inside `${SERVER_LOG_ROOT}/trace.log`. The diagnostic trace service on the WebSphere Application Server's dynamic cache group has to be turned on in order to see the full trace information.

The following trace shows that the cache entry of `/webapp/wcs/stores/ToolTech/ShoppingArea/CatalogSection/CatalogEntrySubsection/ProductDisplay.jsp` is being removed by means of the invalidation Id of `productId:10001`.

Example:

```
[5/29/03 17:46:13:938 EDT] 654b6594 ConfigManager d getCacheEntry()null null
[5/29/03 17:46:13:938 EDT] 654b6594 ConfigManager d Searching cache entries for 1of2null
[5/29/03 17:46:13:938 EDT] 654b6594 ConfigManager d Searching cache entries for
2of2com.ibm.commerce.catalogmanagement.commands.UpdateCatalogEntryCmdImpl.class
```

```
[5/29/03 17:46:13:938 EDT] 654b6594 ConfigManager d using cache entry: [CacheEntry]
className    : command
name        :
com.ibm.commerce.catalogmanagement.commands.CatalogEntryDeleteCmdImpl.class
all names    :
[com.ibm.commerce.catalogmanagement.commands.CatalogEntryDeleteCmdImpl.class,
com.ibm.commerce.catalogmanagement.commands.DeleteCatalogEntryCmdImpl.class,
com.ibm.commerce.catalogmanagement.commands.CatalogEntryUpdateCmdImpl.class,
com.ibm.commerce.catalogmanagement.commands.UpdateCatalogEntryCmdImpl.class]
sharingPolicy : 1
```

```

properties : {}
[Invalidation 0]
baseName: productId
Inval. Component 0
type      : method
id        : getCatentryId
ignoreValue: false
method    : null
required  : true
values    : {}

```

```

[5/29/03 17:46:13:938 EDT] 654b6594 ConfigManager > getCacheProcessor
[5/29/03 17:46:13:938 EDT] 654b6594 ConfigManager d iClassName = 2
[5/29/03 17:46:13:938 EDT] 654b6594 ConfigManager < getCacheProcessor:
com.ibm.ws.cache.command.CommandCacheProcessor@77cae5ac
[5/29/03 17:46:13:938 EDT] 654b6594 EntryInfo    d set id=null
[5/29/03 17:46:13:938 EDT] 654b6594 Cache        > Cache.internalInvalidateById:
productId:10001
[5/29/03 17:46:13:938 EDT] 654b6594 CacheStatisti d CACHE: Removing id = productId:10001
[5/29/03 17:46:13:938 EDT] 654b6594 CacheStatisti d CACHE: Removing id =
/webapp/wcs/stores/ToolTech/ShoppingArea/CatalogSection/CatalogEntrySubsection/ProductDispl
ay.jsp:storeId=10001:catalogId=10001:productId=10001:DC_lang=-1:DC_curr=USD:DC_porg=-2
001:DC_cont=10003:DC_mg=-999:ISO8859-1
[5/29/03 17:46:13:953 EDT] 654b6594 Cache        > returnEntryToFreeList:
/webapp/wcs/stores/ToolTech/ShoppingArea/CatalogSection/CatalogEntrySubsection/ProductDispl
ay.jsp:storeId=10001:catalogId=10001:productId=10001:DC_lang=-1:DC_curr=USD:DC_porg=-2
001:DC_cont=10003:DC_mg=-999:ISO8859-1, false
[5/29/03 17:46:13:953 EDT] 654b6594 Cache        < Cache.internalInvalidateById:
productId:10001
[5/29/03 17:46:13:953 EDT] 654b6594 ExternalCache d ExternalCacheServices.invalidate id:
productId:10001
[5/29/03 17:46:13:953 EDT] 654b6594 ESIProcessor > invalidateIds
[5/29/03 17:46:13:953 EDT] 654b6594 ESIProcessor > invalidateId 'productId:10001' in
buzz.torolab.ibm.com-2344
[5/29/03 17:46:13:953 EDT] 654b6594 ESIProcessor d writeInt 5
[5/29/03 17:46:13:953 EDT] 654b6594 ESIProcessor d writeString productId:10001
[5/29/03 17:46:13:953 EDT] 654b6594 ESIProcessor < invalidateId 'productId:10001' in
buzz.torolab.ibm.com-2344
[5/29/03 17:46:13:953 EDT] 654b6594 ESIProcessor < invalidateIds

```

If the trace does not show that the invalidation rules have been executed or the cache entries have been removed as a result of an event, some of the possible reasons would be:

- The event executed does not map to the one defined in the invalidation policy.
- The invalidation id generated does not map exactly to the dependency ID.
- Syntax error in the policies.

11.1 Conclusion

Considering caching at the development and architecture time can improve the ability to cache and is highly recommended.

11.1.1 What should be cached

In general a Web page that is being accessed frequently and is stable over a long enough time for meaningful reuse to occur is a candidate for caching. Therefore catalog and product pages are good candidates for caching since catalog information's are not change very often. Caching fragments on shopping pages like headers, footers and sidebars can also improve performance if they hold enough wait or require database access.

All store pages output is driven by JSP processing. Each screen is a composite of the multiple invocations of JSPs. So in general, cache policy are created based on how the JSPs are written.

WebSphere Commerce provides sample cache policies for the different business models which are located in the *WCInstallDir/samples/dynacache/**BusinessModel**/cachespec.xml* files. The sample cache policies provided are for the following WebSphere Commerce commands:

- StoreCatalogDisplay
- TopCategoriesDisplay
- CategoryDisplay
- ProductDisplay

Just because the samples only cache this pages you are not limited to these pages and you should look at registration or shopping pages.

11.1.2 Where caching should take place

It is very important to take the time to determine where you should place your cached object as it can affect the overall performance and scalability of the system. One of the deciding factors in making this decision is if request attribute are required or will values come from cookies or URL parameters. If request attributes are required then the ESI cache is not an option as it only support cookies and URL parameters. Servlet level cache should be the next best options as it works with request attributes, cookies and URL parameters. The last option is to cache at the fragment level.

11.1.3 How to invalidate the cached data

Cache invalidation planning should not be done on its own. After having a high level plan on what to be cached, the process to determine what exactly are the fragments to be cached, the design on how to cache them and how to invalidate the

cache should be carried out at the same time since one might affect the other and vice versa. For instance, it is not always that we are able to invalidate every fragments that we plan to cache or the fragments get invalidated so often that it offsets the benefit of caching and sometimes even degrade the performance and therefore decision has to be made if those fragments are to be cached. Other factors such as business requirement etc. would also determine the entire caching and invalidation scheme.

Normally cache fragments that get high reuse rate, low invalidation rate and cost, and few variations is a good candidate for caching, therefore one big exercise in planning for cache invalidation is to investigate on how to invalidate the cache fragments that have been planned to cache and evaluate the effectiveness of the invalidation scheme.

Steps for this task:

- Sort out what events will cause the cache fragments to be changed.
- Decide which invalidation vehicles to use.
- Put together the invalidation policies.
- Measure the effectiveness of the invalidation scheme.

11.1.3.1 Sort out what events causes the fragment change

In WebSphere Commerce, basically there are a number of ways that can change the business data and thus potentially affects the cache content.

Examples:

- Update made with WebSphere Commerce tools such as the WebSphere Commerce Accelerator, Administration Console, Organization Administration Console, Catalog Manager Web Editor or Product Management Tool, etc. e.g. An administrator updates the description on a product with the Product Management Tool, and the product page in cache displays the product information.
- Update made during a shopping flow, e. g. An user adds a new item to the shopping cart that was previously cached.
- Update made by the staging server, e. g. Seasonal massive update to the catalog through the staging server to the production server that currently caches the catalog pages.
- Database updates made by MassLoader.
- Direct /ad hoc database updates, e. g. Increase the prices on a product in response to a sudden market change, and the product page was cached with the old price figures.
- Update to the JSPs, gifs etc. e. g. A store changes its store logo which is displayed on all the store header pages that have been cached.
- Changes to business logic, e. g. Store developer changes the logic to calculate the price as required by new business requirement and the item page was cached with the original price.

11.1.3.2 Decide which invalidation vehicle to use

Considering all the possible events that can cause a cache entry to change, the next step is to look into what is the best available invalidation method to intercept those events and trigger the invalidation. In some situation, the impact of the event might become so big that it could cause almost everything to change, one option that the users can consider is to clear up the whole cache.

Designing on how to invalidate very often depends on how an event occurs. For instance, when a product description is updated via the Catalog Management Tool, WebSphere Commerce internally invokes the commands, AddCatalogEntryDescCmd or UpdateCatalogEntryDescCmd to actually make the changes. Therefore users can choose the command-based invalidation approach to automate the invalidation of the cache entries that have been created on the product page by intercepting these two commands.

Example:

```
<cache-entry>
  <class> servlet</class>
  <name> /Tooltech/ShoppingArea/CatalogSection/CategoryEntrySubsection/CachedProductDisplay.jsp</name>
  <name> /ToolTech/ShoppingArea/CatalogSection/CatalogEntrySubsection/CachedItemDisplay.jsp</name>
  <property name="save-attributes"> false</property>

  <cache-id>
    <component id="storeId" type="parameter">
      <required> true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required> true</required>
    </component>
    <component id="productId" type="parameter">
      <required> true</required>
    </component>
    <component id="userState" type="parameter">
      <required> false</required>
    </component>
    <component id="registeredUser" type="parameter">
      <required> false</required>
    </component>
    <component id="rfqLinkDisplayed" type="parameter">
      <required> false</required>
    </component>
    <component id="DC_lang" type="attribute">
      <required> true</required>
    </component>
    <component id="DC_curr" type="attribute">
```

```

        <required>true</required>
    </component>
    <component id="DC_porg" type="attribute">
        <required>true</required>
    </component>
    <component id="DC_cont" type="attribute">
        <required>true</required>
    </component>
    <component id="DC_mg" type="attribute">
        <required>true</required>
    </component>
</cache-id>

<dependency-id>productId
    <component id="productId" type="parameter">
        <required>true</required>
    </component>
</dependency-id>
</cache-entry>

<cache-entry>
    <class>command</class>
    <name>com.ibm.commerce.catalogmanagement.commands.AddCatalogEntryDescCmdImpl</name>
    <name>com.ibm.commerce.catalogmanagement.commands.UpdateCatalogEntryDescCmplImpl</name>

    <invalidation>productId
        <component id="getCatentryId" type="method">
            <required>true</required>
        </component>
    </invalidation>
</cache-entry>

```

11.1.3.3 Put together the invalidation policies

Previous sections already describe examples on how to put together the invalidation policies. It is also important to understand that the nature of the cache data can affect the way the policies are constructed.

One common example in WebSphere Commerce are the user-specific fragments. If the caching plan is to cache a specific fragment in which each user requires its own version, it might end up that more cache spaces will be required and thus affecting the efficiency of the cache. However if we know that these kind of cache data would only be reused within the user session, we can combine several invalidation mechanisms to remove the cache entries in a more timely manner when the session is over or after a certain time period. By doing that, cache space can be more effectively recycled for other more demanded data.

The following example shows how to invalidate a user's requisition list cache object. This combines several approaches: time-based invalidation rules, and priority setting to time out the cache entries; Rule-based invalidation rule to remove the cache entries when the user logs out or the requisition list is updated.

Example:

```
<cache-entry>
  <class> servlet</class>
  <name>com.ibm.commerce.server.RequestServlet.class</name>
  <property name="store-cookies"> false</property>
  <property name="save-attributes"> false</property>
  <property name="consume-subfragments"> true</property>

<!-- ***** -->
<!-- cache RequisitionListDetailDisplay URI for each user in a store -->
<!-- ***** -->
<cache-id>
  <component id="" type="pathinfo">
    <required> true</required>
    <value> /RequisitionListDetailDisplay</value>
  </component>
  <component id="requisitionListId" type="parameter">
    <required> true</required>
  </component>
  <component id="DC_storeId" type="attribute">
    <required> true</required>
  </component>
  <component id="DC_userId" type="attribute">
    <required> true</required>
  </component>
  <component id="DC_lang" type="attribute">
    <required> true</required>
  </component>
  <component id="DC_curr" type="attribute">
    <required> true</required>
  </component>
  <component id="DC_porg" type="attribute">
    <required> true</required>
  </component>
  <component id="DC_cont" type="attribute">
    <required> true</required>
  </component>
  <component id="DC_mg" type="attribute">
    <required> true</required>
  </component>

<!-- ***** -->
```

```

    <!-- allow output stay in cache for 1 cycle through the LRU algorithm -->
    <!-- ***** -->
    <priority>1</priority>
    <timeout>3600</timeout>
</cache-id>

<!-- ***** -->
<!-- Used for invalidating the cache entry that contains a specific list -->
<!-- ***** -->
<dependency-id>requisitionListId
    <component id="" type="pathinfo" ignore-value="true">
        <required>true</required>
        <value>/RequisitionListDetailDisplay</value>
    </component>
    <component id="requisitionListId" type="parameter">
        <required>true</required>
    </component>
</dependency-id>

<!-- ***** -->
<!-- Used for invalidating all the cache entries that belong to the user -->
<!-- in the store -->
<!-- ***** -->
<dependency-id>storeId:userId:requisitionLists
    <component id="" type="pathinfo" ignore-value="true">
        <required>true</required>
        <value>/RequisitionListDetailDisplay</value>
    </component>
    <component id="DC_storeId" type="attribute">
        <required>true</required>
    </component>
    <component id="DC_userId" type="attribute">
        <required>true</required>
    </component>
    <component id="requisitionListId" type="parameter" ignore-value="true">
        <required>true</required>
    </component>
</dependency-id>

<!-- ***** -->
<!-- Invalidate the cache entry that contains a specific list when the list -->
<!-- is updated -->
<!-- ***** -->
<invalidation>requisitionListId
    <component id="" type="pathinfo" ignore-value="true">
        <required>true</required>
        <value>/RequisitionListItemUpdate</value>
        <value>/RequisitionListDelete</value>
        <value>/RequisitionListUpdate</value>

```

```

    </component>
    <component id="requisitionListId" type="parameter">
      <required>true</required>
    </component>
  </invalidation>

<!-- ***** -->
<!-- Invalidate all the list cache entries that belong to the user in the store -->
<!-- when the user is logoff -->
<!-- ***** -->
<invalidation>storeId:userId:requisitionLists
  <component id="" type="pathinfo" ignore-value="true">
    <required>true</required>
    <value>/Logoff</value>
  </component>
  <component id="DC_storeId" type="attribute">
    <required>true</required>
  </component>
  <component id="DC_userId" type="attribute">
    <required>true</required>
  </component>
</invalidation>
</cache-entry>

```

11.1.3.4 Measure the effectiveness of the invalidation scheme

One way to verify the effectiveness of the caching scheme is to measure the ratio of invalidation hits to the cache hits for each of cache entry. A high ratio value might indicate that the entry is not recommended to be cached or the invalidation policies have not been defined properly, for instance, they might be invalidating more than they need to. There is no rule of thumb to determine what should not be cached or how they should be invalidated, sometimes it all depends on the business requirement.

12.1 Simple file servlet

By default WebSphere Commerce does not cache static data such as images and HTML. That is, the system property *com.ibm.servlet.file.esi.timeOut* to zero. If the user wants to enable images and HTML caching, they could either

- Modify the system property *com.ibm.servlet.file.esi.timeOut* by setting to a non-zero timeout value
- Remove the *com.ibm.servlet.file.esi.timeOut* System property, by default static contents are cached and timeout is default to 300 seconds
- Set up the *cache-entry* in the *cachespec.xml* file for the *SimpleFileServlet*.

Setting up the *cache-entry* in the *cachespec.xml* is a better approach because controlling what is being cached from WebSphere Application Server also gives the you the ability to invalidate the cache entries when needed. One reason of why the user needs to invalidate some images file is that for some stores, the logo and banner can be modified at will. Therefore if the logo and banner images are cached then they will need to be invalidated.

Note: When storing all possible images as cache entries at the application server which may compromise performance when the maximum number of cache entries is reached. It is recommended that you consider setting a lower priority when specifying *SimpleFileServlet*.

Here is the *cache-entry* for caching static content with the Simple file servlet. If this *cache-entry* is put into the *cachespec.xml* in the *Stores.war* then all the images and HTML files that has the context root of */webapp/wcs/stores* will be cached.

Example:

```
<cache-entry>
  <class>servlet</class>
  <name>com.ibm.ws.webcontainer.servlet.SimpleFileServlet.class</name>
  <property name="EdgeCacheable">true</property>

  <cache-id>
    <component id="" type="pathinfo">
      <required>true</required>
    </component>
  </cache-id>

  <dependency-id>
    <component id="" type="pathinfo">
      <required>true</required>
    </component>
  </dependency-id>
  <timeout>1800</timeout>
</cache-entry>
```

Here is an example of the invalidation policies for these images, so if the *StoreLogoUpdateCmdImpl* or *StoreBannerUpdateCmdImpl* are executed successfully then the images */ToolTech/images/logo.gif* and */ToolTech/images/banner.gif* are invalidated:

```
<cache-entry>
  <class> command</class>
  <name>com.ibm.commerce.tools.devtools.store.commands.StoreLogoUpdateCmdImpl</name>

  <invalidation> ToolTech/images/logo.gif</invalidation>
</cache-entry>

<cache-entry>
  <class> command</class>
  <name>com.ibm.commerce.tools.devtools.store.commands.StoreBannerUpdateCmdImpl
  </name>

  <invalidation> ToolTech/images/banner.gif</invalidation>
</cache-entry>
```

12.1.1 Selective caching

Another alternative would be cache the images selectively, i. e. Do not cache the logo and banner images so that we do not have to worry about invalidation. Here is an example of the *cache-entry*:

```
<cache-entry>
  <class> servlet</class>
  <name> com.ibm.ws.webcontainer.servlet.SimpleFileServlet.class</name>
  <property name="EdgeCacheable"> true</property>

  <cache-id>
    <component id="" type="pathinfo">
      <required> true</required>
      <not-value> ToolTech/images/logo.gif</not-value>
      <not-value> ToolTech/images/banner.gif</not-value>
    </component>
  </cache-id>
</cache-entry>
```

13.1 Problem determination

Tracing is a very good problem determination tool. Trace files show the time and sequence of methods called which can be used to determine the failure. This section provides information on how to enable the trace for WebSphere Application Server's dynamic cache, WebSphere Commerce's cache filter and Web Server plugin:

13.1.1 WebSphere Application Server trace

WebSphere Application Server trace for dynamic cache can be turned on from the WebSphere Application Server's Administrative Console.

Here is how to turn on the trace for the dynamic cache component:

1. Log onto the WebSphere Application Server Administrative Console.
2. Click **Troubleshooting > Logs and Trace > server > Diagnostic trace.**
3. Check the *Enable Trace* check box.
4. Enter *com.ibm.ws.cache.*=all=enabled* in the *Trace Specification* box.
5. Click *OK* and save your configuration.
6. Restart the Application Server.

13.1.2 WebSphere Commerce cache trace

WebSphere Commerce trace is also incorporated into WebSphere Application Server's trace service, therefore tracing for the *cache filter* can be enabled in the same way. Here are the steps to enable the trace for the WebSphere Commerce CACHE component:

1. Log onto the WebSphere Application Server Administrative Console.
2. Click **Troubleshooting > Logs and Trace > server > Diagnostic trace.**
3. Check the *Enable Trace* check box.
4. Enter *com.ibm.websphere.commerce.WC_CACHE=all=enabled* in the *Trace Specification* box.
5. Click *OK* and save your configuration.
6. Restart the Application Server.

Note : Multiple components can be specified by a : (colon) as a separator.

For example:

```
com.ibm.websphere.commerce.WC_CACHE=all=enabled:com.ibm.ws.cache.*=all=enabled
```

Please refer to the end of this document for an example of the WebSphere Commerce Cache trace output.

13.1.3 Trace strings

Trace strings must conform to a specific grammar for processing by the trace service. The specification of this grammar follows:

```
TRACESTRING=COMPONENT_TRACE_STRING[:COMPONENT_TRACE_STRING]*  
COMPONENT_TRACE_STRING=COMPONENT_NAME=LEVEL=STATE[,LEVEL=STATE]*  
LEVEL = all | entryExit | debug | event  
STATE = enabled | disabled  
COMPONENT_NAME = COMPONENT | GROUP
```

13.1.4 Web server plug-in trace

The Web server plug-in trace can be enabled by changing the *LogLevel* in the *plugin-cfg.xml* file.

Here is an example of the line in *plugin-cfg.xml* file:

```
<Log LogLevel="Error" Name="D:\WebSphere\AppServer5\logs\http_plugin.log"/>
```

- **Name**
the fully qualified path to the log file to which the plug-in will write error messages
- **LogLevel**
The level of detail of the log messages that the plug-in would write to the log
Values for this attribute is on of the followings:
 - Trace
 - Warn
 - Error (Default)

13.1.5 Trace output for Application Server

By default, the trace are directly to a file `${SERVER_LOG_ROOT}/trace.log`.

Original StoreCatalogDisplay.jsp

14.1 Sample JSPs

14.1.1 Original StoreCatalogDisplay.jsp

```
<%
//*****
//*-----
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001, 2002
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//*
//*-----
//*
//*
%>
...
<%@ include file="../../../include/EnvironmentSetup.jsp"%>

<%
//Parameters may be encrypted. Use JSPHelper to get URL parameter instead of request.getParameter().
JSPHelper jhelper = new JSPHelper(request);
String storeId = jhelper.getParameter("storeId");
String catalogId = jhelper.getParameter("catalogId");
String languageId = jhelper.getParameter("langId");

//Create URL for RFQ
String host = request.getServerName();

String StoresWebPath = ConfigProperties.singleton().getValue("WebServer/StoresWebPath");
Vector contextPath = ConfigProperties.singleton().getAllValues("Websphere/WebModule/Module/contextPath");
Vector urlMappingPath = ConfigProperties.singleton().getAllValues("Websphere/WebModule/Module/urlMappingPath");
String orgadminPort = ConfigProperties.singleton().getValue("Websphere/OrgAdminPort");

StringBuffer orgAdminConsoleURL = new StringBuffer();
orgAdminConsoleURL.append("https://");
orgAdminConsoleURL.append(host);
orgAdminConsoleURL.append(":");
orgAdminConsoleURL.append(orgadminPort);
orgAdminConsoleURL.append(((String)contextPath.elementAt(3));
orgAdminConsoleURL.append(((String)urlMappingPath.elementAt(3));
orgAdminConsoleURL.append("/BuyAdminConsoleView?XMLFile=buyerconsole.BuySiteAdminConsole&webPathToUse=");
orgAdminConsoleURL.append(StoresWebPath);
orgAdminConsoleURL.append("&langId=");
orgAdminConsoleURL.append(languageId);
orgAdminConsoleURL.append("&storeIdToUse=");
orgAdminConsoleURL.append(storeId);
OrgAdminConsoleURL.append("&buyer=true");
String ApprovalToolLinkURL = response.encodeURL(orgAdminConsoleURL.toString());

String BrowserVerErrorURL = response.encodeURL("https://" + host + "/webapp/wcs/stores/servlet/BrowserVerErrorView");
%>
<flow:ifEnabled feature="customerCare">
<%
// Set header type needed for this JSP for LiveHelp. This must
// be set before HeaderDisplay.jsp
request.setAttribute("liveHelpPageType", "personal");
%>
</flow:ifEnabled>
<%
UserRegistrationDataBean bnRegUser = new UserRegistrationDataBean();
com.ibm.commerce.beans.DataBeanManager.activate(bnRegUser, request);
```

Original StoreCatalogDisplay.jsp

```
Integer [] userRoles = bnRegUser.getRoles();

//Get User's Role and determine whether to display the Buyer Approver link
boolean bBuyerApprover = false;
boolean bBuyerBuySide = false;
boolean bBuyerAdmin = false;
for (int i=0; i < userRoles.length; i++) {
    RoleDataBean dbRole = new RoleDataBean();
    // Using setInitKey_RoleId here instead of setRokeId because the databean
    // won't activate properly. setInitKey_RoleId is from RoleAccessBean. To comply
    // with models and access control, we do not use RoleAccessBean.
    dbRole.setInitKey_RoleId (userRoles[i].toString());
    DataBeanManager.activate(dbRole, request);
    if (dbRole.getRoleId().equals ("-22")) {
        bBuyerApprover = true;
    }
    else if (dbRole.getRoleId().equals ("-24")) {
        bBuyerBuySide = true;
    }
    else if (dbRole.getRoleId().equals ("-21")) {
        bBuyerAdmin = true;
    }
}
String userState = cmdcontext.getUser().getState();

String regURL = null;
String pageToInclude = "";
boolean bInclude = false;
if (!(bBuyerBuySide && (bBuyerApprover || bBuyerAdmin)) && (userState.equals("0"))){
    // If the user is not a buyer approve, and pending state, redirect to account
    pageToInclude = "UserArea/AccountSection/UserAccountDisplay.jsp";
    bInclude = true;
}
else if (!bnRegUser.findUser()){
    // If the user is a guest shopper
    pageToInclude = "UserArea/AccountSection/LogonSubsection/UserLogonForm.jsp";
    bInclude = true;
}
if (bInclude) {
%>
    <jsp:include page="<%=pageToInclude%>" flush="true"/>
<% } %>

<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-transitional.dtd">
<HEAD>
    .
    .
    .
</HEAD>

<BODY MARGINHEIGHT="0" MARGINWIDTH="0" LEFTMARGIN="0" TOPMARGIN="0">
<%
    String incfile;
    incfile = includeDir + "HeaderDisplay.jsp";
%>
<jsp:include page="<%=incfile%>" flush="true"/>

<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" height="99%" width="790">
<TR>
    <TD VALIGN="top" BGCOLOR="#4c6178" WIDTH="160">
    <%
        incfile = includeDir + "SidebarDisplay.jsp";
    %>
    <jsp:include page="<%=incfile%>" flush="true"/>
    </TD>
    <TD valign="top" width="630">

    <!--MAIN CONTENT STARTS HERE-->
```

Original StoreCatalogDisplay.jsp

```
<%
// This page should only appear to user under the following conditions
// User State is Approved - show the regular catalog
// OR
// User State is null but user is registered - show the regular catalog
%>

<!-- Start Main JSP Content -->
<TABLE cellpadding="8" cellspacing="0" width="580" border="0" align="left">
<tr>
<td align="left" valign="top" colspan="3" class="categoryspace">
<H1><%=tooltechtext.getString("Header_Home")%></H1>
<!--START PROMO -->
<%
// create the e-Marketing Spot
EMarketingSpot eMarketingSpot = new EMarketingSpot();

// IMPORTANT - set the correct name here
eMarketingSpot.setName("StoreHomePage");

// instantiate the bean
DataBeanManager.activate(eMarketingSpot, request);
EMarketingSpot.CatalogEntry[] productResults = eMarketingSpot.getCatalogEntries();
if (productResults != null && productResults.length > 0) {
    for (int i = 0; i < productResults.length; i++) {
        EMarketingSpot.CatalogEntry catalogBean = productResults[i];
        CatalogEntryDescriptionAccessBean catalogDescriptionBean = catalogBean.getDescription();
%>
        <B><%=tooltechtext.getString("Home_ImageText")%></B>
        <BR>
        <a href="ProductDisplay?catalogId=<%=catalogId%>&storeId=<%= storeId%>&productId=<%=
catalogBean.getCatalogEntryID()%>&langId=<%=languageId%>">
        " width="150" height="150" border="0">
        </a><br>
        <font class="text"><%= catalogDescriptionBean.getLongDescription()%>&nbsp;  </font><BR><BR>
    }
}
%>
<!--END PROMO -->

<TABLE border="0" cellspacing="0" cellpadding="0">
...
</TABLE>

<P>
<%
if (bBuyerBuySide && (bBuyerApprover | bBuyerAdmin)) { %>
<script language="JavaScript">
    checkBrowser(); //calls checkBrowser function for ApprovalToolLink() below
</script>
<hr width="580" noshade align="left">
<P>
<b><font color=#ffffff><A CLASS="catalog" HREF="javascript: ApprovalToolLink();">
    <%= tooltechtext.getString ("Home_Link1") %> </A></font>
</b>
<%
    }
%>
<P>
</TD>
</TR>
</TABLE>
<br>
</TD>
</TR>
```

Original StoreCatalogDisplay.jsp

```
</TABLE>  
</BODY>  
</HTML>
```

Original HeaderDisplay.jsp

14.1.2 Original HeaderDisplay.jsp

```
<%
//*****
//*-----
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001, 2002
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//*
//*-----
//*
//*
%>
...
<jsp:useBean id="sdb" class="com.ibm.commerce.common.beans.StoreDataBean" scope="request">
  <% com.ibm.commerce.beans.DataBeanManager.activate(sdb, request); %>
</jsp:useBean>

<%
try {
  CommandContext cmdcontext = (CommandContext) request.getAttribute(ECConstants.EC_COMMANDCONTEXT);
  Locale locale = cmdcontext.getLocale();
  String lastCmdName = cmdcontext.getCommandName().trim();

  //Parameters may be encrypted. Use JSPHelper to get URL parameter instead of request.getParameter().
  JSPHelper jhelper = new JSPHelper(request);

  String storeId = cmdcontext.getStoreId().toString();
  String languageId = cmdcontext.getLanguageId().toString();
  String cmpCategoryId = jhelper.getParameter("categoryId");

  // Setup Store directories
  String storeDir = (String) request.getAttribute("storeDir");
  String fileDir = (String) request.getAttribute("fileDir");
  String includeDir = (String) request.getAttribute("includeDir");
  String bundleDir = (String) request.getAttribute("bundleDir");
  String storeName = "";

  if (storeDir == null) {
    storeDir = sdb.getJspPath();
    fileDir = sdb.getFilePath();
    includeDir = storeDir + "include" + "/";
    bundleDir = sdb.getDirectory();
    storeName = sdb.getDescription(cmdcontext.getLanguageId()).getDisplayName();
    request.setAttribute("storeName", storeName);
    request.setAttribute("storeDir", storeDir);
    request.setAttribute("includeDir", includeDir);
    request.setAttribute("fileDir", fileDir);
    request.setAttribute("bundleDir", bundleDir);
  }
  ResourceBundle tooltechtext = (ResourceBundle) request.getAttribute("ResourceText");
  if (tooltechtext == null) {
    tooltechtext = ResourceBundle.getBundle(bundleDir + "/tooltechtext", locale);
    request.setAttribute("ResourceText", tooltechtext);
  }
  CatalogDataBean Catalogs[] = sdb.getStoreCatalogs();
  String catalogId = Catalogs[0].getCatalogId();
  if (cmpCategoryId == null)
    cmpCategoryId = "";
}
%>

<jsp:useBean id="userRegistrationDataBean" class="com.ibm.commerce.user.beans.UserRegistrationDataBean" scope="page">
```

Original HeaderDisplay.jsp

[illegible]

Original HeaderDisplay.jsp

```
        </tr>
    </table>

<%
    }
%>
    </td>
</table>

<!--END HEADER-->
<%
}
catch (Exception e) {
    out.println(e);
}
%>
```


Original SidebarDisplay.jsp

14.1.3 Original SidebarDisplay.jsp

```
<%
//*****
//*-----
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001, 2002
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//*-----
//*
//*>
...
<%@ include file="EnvironmentSetup.jsp"%>

<jsp:useBean id="sdbSidebar" class="com.ibm.commerce.common.beans.StoreDataBean" scope="request">
  <% com.ibm.commerce.beans.DataBeanManager.activate(sdbSidebar, request); %>
</jsp:useBean>

<%
try {
  //Parameters may be encrypted. Use JSPHelper to get URL parameter instead of request.getParameter().
  JSPHelper jhelper = new JSPHelper(request);

  String storeId = jhelper.getParameter("storeId");
  String languageId = jhelper.getParameter("langId");

  CatalogDataBean Catalogs[] = sdbSidebar.getStoreCatalogs();
  String catalogId = Catalogs[0].getCatalogId();

  //Create URL for RFQ
  String host = request.getServerName();
  String IEencodedURL = response.encodeURL("https://" + host +
"/webapp/wcs/stores/servlet/MerchantCenterView?XMLFile=utf.utfCSA&storeId=" + storeId);
  String BrowserVerErrorURL = response.encodeURL("https://" + host + "/webapp/wcs/stores/servlet/BrowserVerErrorView");

  UserRegistrationDataBean bnRegUser = new UserRegistrationDataBean();
  com.ibm.commerce.beans.DataBeanManager.activate(bnRegUser, request);
  Integer [] userRoles = bnRegUser.getRoles();
  String userState = cmdcontext.getUser().getState();
  String userType = bnRegUser.getRegisterType().trim();
}%>
<%
if (!userType.equals("G") && (userState.equals("1") || userState.equals(""))) {
}%>
  <TABLE BORDER="0" CELLPADDING="2" CELLSPACING="0" WIDTH="160" BGCOLOR="#4c6178">
    ...
    <flow:ifEnabled feature="customerCare">
      <%
        //Link for Live Help
        if (com.ibm.commerce.collaboration.livehelp.beans.LiveHelpShopperConfiguration.isEnabled()) {
      %>
        ...
      <%
        }
      %>
    </flow:ifEnabled>
  </TABLE>
}
}%>
<br>
```

Original SidebarDisplay.jsp

```
<!--END SIDEBAR NAVIGATION-->
<%
}
catch (Exception e) {
    out.println("Exception:" + e);
}
%>
```

StoreCatalogDisplay.jsp

14.1.4StoreCatalogDisplay.jsp

```
<%
//*****
//*-----
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001, 2002
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//*
//*-----
//*
//*
%>
...
<%@ include file="../../include/EnvironmentSetup.jsp"%>
<%@ include file="../../include/CacheParametersSetup.jsp"%>

<%
//Parameters may be encrypted. Use JSPHelper to get URL parameter instead of request.getParameter().
JSPHelper jhelper = new JSPHelper(request);
String storeId = jhelper.getParameter("storeId");
String catalogId = jhelper.getParameter("catalogId");
String languageId = jhelper.getParameter("langId");

//Create URL for RFQ
String host = request.getServerName();

String StoresWebPath = ConfigProperties.singleton().getValue("WebServer/StoresWebPath");
Vector contextPath = ConfigProperties.singleton().getAllValues("Websphere/WebModule/Module/contextPath");
Vector urlMappingPath = ConfigProperties.singleton().getAllValues("Websphere/WebModule/Module/urlMappingPath");
String orgadminPort = ConfigProperties.singleton().getValue("Websphere/OrgAdminPort");

StringBuffer orgAdminConsoleURL = new StringBuffer();
orgAdminConsoleURL.append("https://");
orgAdminConsoleURL.append(host);
orgAdminConsoleURL.append(":");
orgAdminConsoleURL.append(orgadminPort);
orgAdminConsoleURL.append((String)contextPath.elementAt(3));
orgAdminConsoleURL.append((String)urlMappingPath.elementAt(3));
orgAdminConsoleURL.append("/BuyAdminConsoleView?XMLFile=buyerconsole.BuySiteAdminConsole&webPathToUse=");
orgAdminConsoleURL.append(StoresWebPath);
orgAdminConsoleURL.append("&langId=");
orgAdminConsoleURL.append(languageId);
orgAdminConsoleURL.append("&storeIdToUse=");
orgAdminConsoleURL.append(storeId);
orgAdminConsoleURL.append("&buyer=true");
String ApprovalToolLinkURL = response.encodeURL(orgAdminConsoleURL.toString());

String BrowserVerErrorURL = response.encodeURL("https://" + host + "/webapp/wcs/stores/servlet/BrowserVerErrorView");
%>
<flow:ifEnabled feature="customerCare">
<%
// Set header type needed for this JSP for LiveHelp. This must
// be set before HeaderDisplay.jsp
request.setAttribute("liveHelpPageType", "personal");
%>

<%
String regURL = null;
String pageToInclude = "";
boolean bInclude = false;
if (!(bBuyerBuySide && (bBuyerApprover || bBuyerAdmin)) && (userState.equals("0"))){
```

StoreCatalogDisplay.jsp

```
// If the user is not a buyer approve, and pending state, redirect to account
pageToInclude = "UserArea/AccountSection/UserAccountDisplay.jsp";
bInclude = true;
}
else if (!bnRegUser.findUser()){
    // If the user is a guest shopper
    pageToInclude = "UserArea/AccountSection/LogonSubsection/UserLogonForm.jsp";
    bInclude = true;
}
if (bInclude) {
%>
    <jsp:include page="<%=pageToInclude%>" flush="true"/>
<% } %>

<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml11-transitional.dtd">
<HEAD>
    .
    .
    .
</HEAD>

<BODY MARGINHEIGHT="0" MARGINWIDTH="0" LEFTMARGIN="0" TOPMARGIN="0">

<%
    String incfile;
    incfile = includeDir + "CachedHeaderDisplay.jsp";
%>
    <jsp:include page="<%=incfile%>" flush="true">
        <jsp:param name="storeId" value="<%= storeId %>" />
        <jsp:param name="catalogId" value="<%= catalogId %>" />
        <jsp:param name="langId" value="<%= languageId %>" />
        <jsp:param name="userType" value="<%= userType %>" />
        <jsp:param name="liveHelp" value="<%= liveHelp %>" />
        <jsp:param name="rfqLinkDisplayed" value="<%= rfqLinkDisplayed %>" />
    </jsp:include>

    <TABLE CELLPADDING="0" CELSPACING="0" BORDER="0" height="99%" width="790">
    <TR>
    <TD VALIGN="top" BGCOLOR="#4c6178" WIDTH="160">
    <%
        incfile = includeDir + "CachedSidebarDisplay.jsp";
    %>
        <jsp:include page="<%=incfile%>" flush="true">
            <jsp:param name="storeId" value="<%= storeId %>" />
            <jsp:param name="catalogId" value="<%= catalogId %>" />
            <jsp:param name="langId" value="<%= languageId %>" />
            <jsp:param name="userType" value="<%= userType %>" />
            <jsp:param name="userState" value="<%= userState %>" />
            <jsp:param name="liveHelp" value="<%= liveHelp %>" />
        </jsp:include>
    </TD>

    <TD valign="top" width="630">
    <!--MAIN CONTENT STARTS HERE-->
    <%
        // This page should only appear to user under the following conditions
        // User State is Approved - show the regular catalog
        // OR
        // User State is null but user is registered - show the regular catalog
    %>
        <jsp:include page="CachedStoreCatalogDisplay.jsp" flush="true">
            <jsp:param name="storeId" value="<%= storeId %>" />
            <jsp:param name="catalogId" value="<%= catalogId %>" />
            <jsp:param name="langId" value="<%= languageId %>" />
            <jsp:param name="displayApproverLink" value="<%= displayApproverLink %>" />
        </jsp:include>
    <br>
```

StoreCatalogDisplay.jsp

```
</TD>  
</TR>  
</TABLE>  
</BODY>  
</HTML>
```

CachedParametersSetup.jsp

14.1.5CachedParametersSetup.jsp

```
<%
//*****
//*-----
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001, 2002
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//*
//*-----
//*
//*
%>

<%@ page import="com.ibm.commerce.beans.DataBeanManager" %>
<%@ page import="com.ibm.commerce.user.beans.UserRegistrationDataBean" %>
<%@ page import="com.ibm.commerce.user.beans.RoleDataBean" %>

<%
// LiveHelp
boolean liveHelp = com.ibm.commerce.collaboration.livehelp.beans.LiveHelpShopperConfiguration.isEnabled();
%>

<jsp:useBean id="userRegistrationDataBean" class="com.ibm.commerce.user.beans.UserRegistrationDataBean" scope="request">
    <% com.ibm.commerce.beans.DataBeanManager.activate(userRegistrationDataBean, request); %>
</jsp:useBean>

<%
// User Information
Long   userId = null;
String userState = null;
String userType = null;
Integer[] userRoles = null;
boolean bBuyerApprover = false;
boolean bBuyerBuySide = false;
boolean bBuyerAdmin = false;
boolean rfqLinkDisplayed = false;
boolean displayApproverLink = false;
boolean bRegisteredUser = false;
boolean bGuestUser = false;

if (cmdcontext != null) {
    userId      = cmdcontext.getUserId();
    userState    = cmdcontext.getUser().getState();
    userType     = cmdcontext.getUser().getRegisterType();
    userRoles    = userRegistrationDataBean.getRoles();

    if (userRegistrationDataBean.findUser()) {
        bRegisteredUser = true;
        bGuestUser      = false;
    }
    else {
        bRegisteredUser = false;
        bGuestUser      = true;
    }
}

for (int i=0; i < userRoles.length; i++) {
    com.ibm.commerce.user.beans.RoleDataBean dbRole = new com.ibm.commerce.user.beans.RoleDataBean();
    dbRole.setInitKey_RoleId (userRoles[i].toString());
    DataBeanManager.activate(dbRole, request);
    if (dbRole.getRoleId().equals ("-22")) {
        bBuyerApprover = true;
    }
}
```

CachedParametersSetup.jsp

```
    }
    else if (dbRole.getRoleId().equals ("-24")) {
        bBuyerBuySide = true;
    }
    else if (dbRole.getRoleId().equals ("-21")) {
        bBuyerAdmin = true;
    }

    if ( (bBuyerBuySide) &&  (! rfqLinkDisplayed) ) {
        rfqLinkDisplayed = true;
    }

    if (bBuyerBuySide && (bBuyerApprover || bBuyerAdmin) ) {
        displayApproverLink = true;
    }
}
}
%>
```

CachedStoreCatalogDisplay.jsp

14.1.6CachedStoreCatgalogDisplay.jsp

```
<%
//*****
//*-----
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001, 2002
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//*
//*-----
//*
//*
%>
...
<%@ include file="../../include/EnvironmentSetup.jsp"%>
<%
//Parameters may be encrypted. Use JSPHelper to get URL parameter instead of request.getParameter().
JSPHelper jhelper = new JSPHelper(request);
String storeId = jhelper.getParameter("storeId");
String catalogId = jhelper.getParameter("catalogId");
String languageId = jhelper.getParameter("langId");
String displayApproverLink = jhelper.getParameter("displayApproverLink");
%>

<TABLE cellpadding="8" cellspacing="0" width="580" border="0" align="left">
  <tr>
    <td align="left" valign="top" colspan="3" class="categoryspace">
      <H1><%=tooltechtext.getString("Header_Home")%></H1>

      <!-- PROMO CONTENT STARTS HERE -->
      <jsp:include page="../../include/StoreCatalogProductESpot.jsp" flush="true">
        <jsp:param name="emsName" value="StoreHomePage" />
        <jsp:param name="catalogId" value="<%= catalogId %>" />
      </jsp:include>
      <!-- PROMO CONTENT ENDS HERE -->

      <TABLE border="0" cellspacing="0" cellpadding="0">
        ...
      </TABLE>

    <P>
    <%
    if (displayApproverLink!=null && displayApproverLink.equalsIgnoreCase("true")) { %>
      <script language="JavaScript">
        checkBrowser(); //calls checkBrowser function for ApprovalToolLink() below
      </script>
      <hr width="580" noshade align="left">
      <P>
      <b><font color=#ffffff><A CLASS="catalog" HREF="javascript: ApprovalToolLink();">
        <%= tooltechtext.getString ("Home_Link1") %> </A></font>
      </b>
    <%
    }
    %>
    <P>
    </TD>
  </TR>
</TABLE>
<br>
</TD>
</TR>
</TABLE>
```


CachedStoreCatalogDisplay.jsp

```
</BODY>  
</HTML>
```

StoreCatalogProductESpot.jsp

14.1.7StoreCatalogProductEspot.jsp

```
<%
//*****
//*-----
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001, 2002
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//*
//*-----
//*
//*
%>

<%
    // setup the command context and store data bean in this page
    com.ibm.commerce.command.CommandContext emsCommandContext = (com.ibm.commerce.command.CommandContext)
request.getAttribute(com.ibm.commerce.server.ECConstants.EC_COMMANDCONTEXT);
    com.ibm.commerce.common.beans.StoreDataBean storeDataBean = new com.ibm.commerce.common.beans.StoreDataBean();
    storeDataBean.setStoreId(emsCommandContext.getStoreId().toString());
    com.ibm.commerce.beans.DataBeanManager.activate(storeDataBean, request);

    // setup the resource bundle to be used in this page
    java.util.ResourceBundle tooltechtext = (java.util.ResourceBundle) request.getAttribute("ResourceText");
    if (tooltechtext == null) {
        tooltechtext = java.util.ResourceBundle.getBundle(storeDataBean.getDirectory() + "/tooltechtext",
emsCommandContext.getLocale());
        request.setAttribute("ResourceText", tooltechtext);
    }

    // setup the local variables to be used in this page
    String storeImagePath = storeDataBean.getFilePath();
    String productESpotName = request.getParameter("emsName");
    String productESpotCatalogId = request.getParameter("catalogId");

    // create the e-Marketing Spot
    com.ibm.commerce.marketing.beans.EMarketingSpot productESpot = new
com.ibm.commerce.marketing.beans.EMarketingSpot();

    // IMPORTANT - set the correct name here
    if (productESpotName == null) {
        productESpotName = "defaultESpotName";
    }
    productESpot.setName(productESpotName);

    // instantiate the bean
    com.ibm.commerce.beans.DataBeanManager.activate(productESpot, request);
    com.ibm.commerce.marketing.beans.EMarketingSpot.CatalogEntry[] productResults = productESpot.getCatalogEntries();
    if (productResults != null && productResults.length > 0) {
        for (int i=0; i<productResults.length; i++) {
            com.ibm.commerce.marketing.beans.EMarketingSpot.CatalogEntry catalogBean = productResults[i];
            com.ibm.commerce.catalog.objects.CatalogEntryDescriptionAccessBean catalogDescriptionBean =
catalogBean.getDescription();
        }
    }
%>
<B><%= tooltechtext.getString("Home_ImageText") %></B>
<BR>
<a href="ProductDisplay?catalogId=<%= productESpotCatalogId %>&storeId=<%= emsCommandContext.getStoreId().toString()
%>&productId=<%= catalogBean.getCatalogEntryID() %>&langId=<%= emsCommandContext.getLanguageId().toString()
%>">
" width="150" height="150" border="0">
</a><br>
```

StoreCatalogProductESpot.jsp

```
<font class="text"><%= catalogDescriptionBean.getLongDescription() %>&nbsp;  </font><BR><BR>
<%
    }
%>
```

CachedHeaderDisplay.jsp

14.1.8CachedHeaderDisplay.jsp

```
<%
//*****
//*-----
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001, 2002
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//*
//*-----
//*
//*
%>
...
<%@ include file="EnvironmentSetup.jsp"%>

<%
// This JSP is called from HeaderDisplay.jsp and is cached based on the parameters
// passed in and defined in the cachespec.xml file.

try {
    //Parameters may be encrypted. Use JSPHelper to get URL parameter instead of request.getParameter().
    JSPHelper jhelper = new JSPHelper(request);

    String storeId = jhelper.getParameter("storeId");
    String languageId = jhelper.getParameter("langId");

    String catalogId = jhelper.getParameter("catalogId");
    String userType = jhelper.getParameter("userType");
    String rfqLinkDisplayed = jhelper.getParameter("rfqLinkDisplayed");

    // Determine the store Logo from the Trading Agreement
    TradingAgreementAccessBean[] currentTradingACBean = cmdcontext.getCurrentTradingAgreements();
    String attachmentURL = "";
    String accountId = "";
    if (currentTradingACBean != null) {
        accountId = (currentTradingACBean[currentTradingACBean.length-1].getAccountId());
        if (accountId != "") {
            AccountDataBean dbAccount = new AccountDataBean();
            dbAccount.setDataBeanKeyAccountId(accountId);
            com.ibm.commerce.beans.DataBeanManager.activate(dbAccount, request);
            AttachmentAccessBean abAttachment = dbAccount.getDisplayCustomizationTCAttachment();
            if (abAttachment != null)
                attachmentURL = fileDir + abAttachment.getAttachmentURL();
        }
    }
}
%>

<!--START HEADER-->
<table cellpadding="0" cellspacing="0" border="0" width="100%">
  <tr>
    ...
  </tr>
  <tr>
    <td background="<%=fileDir%>images/row1_tile.gif">
<%
    if (userType.equals("G")) {
%>
      <table cellpadding="0" cellspacing="0" border="0" width="100%" height="28">
        ...
      </table>
<%
```

CachedHeaderDisplay.jsp

```
}  
else {  
    // Full Navbar  
  
%>  
        <table cellpadding="0" cellspacing="0" border="0" width="100%" height="28">  
            ...  
        </table>  
    </td>  
</tr>  
<tr>  
    <td background="<%=fileDir%>images/row2_tile.gif">  
        <table cellpadding="0" cellspacing="0" border="0" width="100%" height="32">  
            <tr>  
                <td nowrap><font face="Verdana" color="#ffffff" size="-1">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~<br>                    <%=tooltechtex<td align="right">  
                        <table cellpadding="0" cellspacing="0" border="0">  
                            <tr>  
                                ...  
                            </tr>  
                        </table>  
                    }  
                </td>  
            </tr>  
        </table>  
    </td>  
</tr>  
</table>  
  
}  
  
%>  
  
catch (Exception e) {  
    out.println(e);  
}
```

CachedSidebarDisplay.jsp

14.1.9CachedSidebarDisplay.jsp

```
<%
//*****
//*-----
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001, 2002
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//*
//*-----
//*
//*
%>
...

<%@ include file="EnvironmentSetup.jsp"%>

<%
// This JSP is called from SidebarDisplay.jsp and is cached based on the parameters
// passed in and defined in the cachespec.xml file.

try {
    //Parameters may be encrypted. Use JSPHelper to get URL parameter instead of request.getParameter().
    JSPHelper jhelper = new JSPHelper(request);

    String storeId = jhelper.getParameter("storeId");
    String languageId = jhelper.getParameter("langId");
    String catalogId = jhelper.getParameter("catalogId");
    String liveHelp = jhelper.getParameter("liveHelp");
    String userState = jhelper.getParameter("userState");
    String userType = jhelper.getParameter("userType");
%>

<%
    if (!userType.equals("G") && (userState.equals("1") || userState.equals(""))) {
%>
        <TABLE BORDER="0" CELLPADDING="2" CELLSPACING="0" WIDTH="160" BGCOLOR="#4c6178">
            ...
            <flow:ifEnabled feature="customerCare">
                <%
                    //Link for Live Help
                    if (liveHelp.equalsIgnoreCase("true")) {
                %>
                    ...
                <%
                    }
                %>
            </flow:ifEnabled>
        </TABLE>
    }
%>
<br>
<!--END SIDEBAR NAVIGATION-->
<%
}
catch (Exception e) {
    out.println("Exception:" + e);
}
%>
```

WebSphere Commerce Cache Trace Output

15.1 Sample of WebSphere Commerce Cache Trace

```
[6/3/03 18:26:59:412 EDT] 488c3560 WC_CACHE > com.ibm.commerce.dynacache.filter.CacheFilter$1 doFilter() Entry
[6/3/03 18:26:59:432 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter$1 doFilter() Filter parameter - Encoding is null; ServletName = Stores
[6/3/03 18:26:59:432 EDT] 488c3560 WC_CACHE > com.ibm.commerce.browseradapter.CacheSessionData initializeSession Entry
[6/3/03 18:26:59:602 EDT] 488c3560 WC_CACHE d com.ibm.commerce.browseradapter.CacheSessionData initializeSession DKTEST Retrieved UserSession
[6/3/03 18:26:59:612 EDT] 488c3560 WC_CACHE d com.ibm.commerce.browseradapter.CacheSessionData initializeSession No contracts specified on URL.
[6/3/03 18:26:59:612 EDT] 488c3560 WC_CACHE d com.ibm.commerce.browseradapter.CacheSessionData initializeSession Current contracts set to 10003;10004
[6/3/03 18:26:59:612 EDT] 488c3560 WC_CACHE < com.ibm.commerce.browseradapter.CacheSessionData initializeSession Exit
[6/3/03 18:26:59:612 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter$1 doFilter() strUserId=4; strStoreId=10001; strLang=-8; strCurrency=TWD;
strParentOrg=7000000000000000003; strCurrentContracts=10003;10004; strEligibleContracts=10003.10004;strContracts=10003;10004
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.businessPolicy
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_bus the storeId are 10001;
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001; relType=com.ibm.commerce.tax
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_tax the storeId are 10001;
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.campaigns
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_camp the storeId are 10001;
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.catalog
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_cat the storeId are 10001;
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.command
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_cmd the storeId are 10001;
[6/3/03 18:26:59:632 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.hostedStore
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_host the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.price
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_prc the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.referral
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_ref the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.segmentation
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_seg the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.URL
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_url the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.view
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_view the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.inventory
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_inv the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.storeitem
```

WebSphere Commerce Cache Trace Output

[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_baseItem the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.channelStore	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_chs the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.currency.conversion	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_currConv the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.currency.format	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_currFmt the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.currency.supported	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_supCurr the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.currency.countervalue	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_cterCurr the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStorePath() StorePath for storeId=10001;
relType=com.ibm.commerce.measurement.format	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter setRequestAttribute() For DC_meaFmt the storeId are 10001;
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.businessPolicy	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.tax	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.campaigns	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.catalog	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.command	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.hostedStore	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.price	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.referral	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.segmentation	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.URL	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.view	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.inventory	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.storeitem	
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE	d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.channelStore	

WebSphere Commerce Cache Trace Output

```
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.currency.conversion
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.currency.format
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.currency.supported
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.currency.countervalue
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStoresForRelatedStores() getStoresForRelatedStore for storeId=10001;
relType=com.ibm.commerce.measurement.format
[6/3/03 18:26:59:642 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter getStoreStatus() Store Status for storeId 10001 is 1
[6/3/03 18:26:59:812 EDT] 488c3560 WC_CACHE > com.ibm.commerce.dynacache.commands.MemberGroupsCacheCmdImpl setOutputProperties Entry
[6/3/03 18:26:59:812 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.commands.MemberGroupsCacheCmdImpl setOutputProperties CACHE HIT: userId=4;
memberGroups=[Ljava.lang.String;@5b78b544
[6/3/03 18:26:59:812 EDT] 488c3560 WC_CACHE < com.ibm.commerce.dynacache.commands.MemberGroupsCacheCmdImpl setOutputProperties Exit
[6/3/03 18:26:59:812 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter setMemberGroupAttributes() UserId=4; MemberGroups=0
[6/3/03 18:26:59:952 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter -
requestURL=https://barbwong02/webapp/wcs/stores/servlet/StoreCatalogDisplay; requestURI=/webapp/wcs/stores/servlet/StoreCatalogDisplay; pathInfo=/StoreCatalogDisplay;
servletPath=/servlet;
queryString=krypto=H6tSfK3MjYdn8WqHDwkae9AbzZKe9yq1npJ%2FfLprY4%2Ff0E54vVnEYO7Idq6Wi%2FAiXIoCvTgH945M%0D%0ALme6ZIm3dAqSgP6RXPPK8N%2BIhOYA%2Byl6s
3hzL8tfvPLYDdS7Uftiy745Leuk6Ew0lxhcesGKVA%3D%3D&ddkey=Logon
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request parameter ddkey=Logon
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request parameter
krypto=H6tSfK3MjYdn8WqHDwkae9AbzZKe9yq1npJ/flprY4/f0E54vVnEYO7Idq6Wi/AiXIoCvTgH945M
Lme6ZIm3dAqSgP6RXPPK8N+lhOYA+yl6s3hzL8tfvPLYDdS7Uftiy745Leuk6Ew0lxhcesGKVA==
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_ref0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute
com.ibm.servlet.engine.webapp.dispatch_type=forward
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_storeId=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_inv0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_seg0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_mg=0
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_baseItem0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_view0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_chs0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_bus0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_cont=10003;10004
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_curr=TWD
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_tax0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_currFmt0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_currConv0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_camp0=10001
DC_porg=70000000000000000000
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_storeStatus=1
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_cmd0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_meaFmt0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_prc0=10001
```

WebSphere Commerce Cache Trace Output

```
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute
CommandContext=com.ibm.commerce.command.CommandContextImpl@2bd0b544
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_oterCurr0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_userId=4
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_cat0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_url0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_lang=-8
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute jspStoreDir=ToolTech
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_cont1=10004
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_cont0=10003
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processRequestProperties() Pre-Filter - Request attribute
RequestProperties=storeId=10001;jspStoreDir=ToolTech;reLogonURL=LogonForm;logonPassword=*****;ddkey=Logon;catalogId=10001;logonId=buyerTW;
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processRequestProperties() Pre-Filter - Request attribute
DecryptedReqProp=storeId=10001;jspStoreDir=ToolTech;reLogonURL=LogonForm;logonPassword=*****;ddkey=Logon;catalogId=10001;logonId=buyerTW;
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute
SessionContainer=com.ibm.commerce.browseradapter.WCSessionContainer@1513f547
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_host0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute DC_supCurr0=10001
[6/3/03 18:26:59:962 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Pre-Filter - Request attribute EncodingAlreadySet=Big5
[6/3/03 18:26:59:973 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter$1 doFilter() Exit Pre-Filter
[6/3/03 18:27:05:711 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter -
requestURL=https://barbwong02/webapp/wcs/stores/servlet/StoreCatalogDisplay; requestURI=/webapp/wcs/stores/servlet/StoreCatalogDisplay; pathInfo=/StoreCatalogDisplay;
servletPath=/servlet;
queryString=krypto=H6tSfK3MjYdn8WqHDwkae9AbzZKe9yq1npJ%2FfLprY4%2Ff0E54vVnEYO7Idq6Wi%2FAiXIoCvTgH945M%0D%0ALme6ZIm3dAqSgP6RXPPK8N%2BlhOYA%2Byl6s
3hzL8tfvPlyDdS7Uftiy745Leuk6Ew0lxhcesGKVA%3D%3D&ddkey=Logon
[6/3/03 18:27:05:721 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request parameter ddkey=Logon
[6/3/03 18:27:05:721 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request parameter
krypto=H6tSfK3MjYdn8WqHDwkae9AbzZKe9yq1npJ/fLprY4/f0E54vVnEYO7Idq6Wi/AiXIoCvTgH945M
Lme6ZIm3dAqSgP6RXPPK8N+lhOYA+yl6s3hzL8tfvPlyDdS7Uftiy745Leuk6Ew0lxhcesGKVA==
[6/3/03 18:27:05:721 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute fileDir=/wcsstore/ToolTech/
[6/3/03 18:27:05:731 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
sdb=com.ibm.commerce.common.beans.StoreDataBean@125fb545
[6/3/03 18:27:05:741 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
storeId=[Ljava.lang.String;@2ca9f544
[6/3/03 18:27:05:741 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
com.ibm.servlet.engine.webapp.dispatch_nested=true
[6/3/03 18:27:05:741 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
javax.servlet.include.request_uri=null
[6/3/03 18:27:05:741 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_ref0=10001
[6/3/03 18:27:05:741 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
com.ibm.servlet.engine.webapp.dispatch_type=include
[6/3/03 18:27:05:761 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_storeId=10001
[6/3/03 18:27:05:761 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_inv0=10001
[6/3/03 18:27:05:771 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_seg0=10001
[6/3/03 18:27:05:771 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
requestURIPath=/webapp/wcs/stores/servlet/StoreCatalogDisplay
[6/3/03 18:27:05:791 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_mg=0
[6/3/03 18:27:05:791 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_baseItem0=10001
```

WebSphere Commerce Cache Trace Output

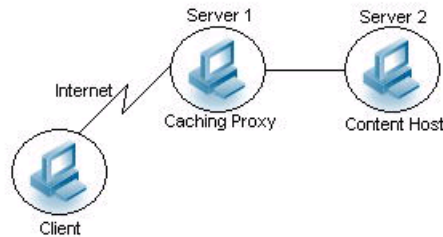
```
[6/3/03 18:27:05:791 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
scf=com.ibm.commerce.tools.devtools.flexflow.scf.impl.SCFImpl@31d2b5bc
[6/3/03 18:27:05:791 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_view0=10001
[6/3/03 18:27:05:791 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
catalogId=[Ljava.lang.String;@2ba63544
[6/3/03 18:27:05:791 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_chs0=10001
[6/3/03 18:27:05:811 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
logonId=[Ljava.lang.String;@2bb5b544
[6/3/03 18:27:05:881 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute bundleDir=ToolTech
[6/3/03 18:27:05:891 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_bus0=10001
[6/3/03 18:27:05:891 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_logonId=buyerTW
[6/3/03 18:27:05:901 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
javax.servlet.include.context_path=null
[6/3/03 18:27:05:941 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_cont=10003;10004
[6/3/03 18:27:05:941 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_curr=TWD
[6/3/03 18:27:05:941 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
reLogonURL=[Ljava.lang.String;@2b6db544
[6/3/03 18:27:05:941 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_tax0=10001
[6/3/03 18:27:05:941 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute storeDir=/ToolTech/
[6/3/03 18:27:05:941 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
logonPassword=[Ljava.lang.String;@2b597544
[6/3/03 18:27:05:971 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_currFmt0=10001
[6/3/03 18:27:06:011 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_currConv0=10001
[6/3/03 18:27:06:061 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_camp0=10001
[6/3/03 18:27:06:061 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
DC_porg=70000000000000000003
[6/3/03 18:27:06:061 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_storeStatus=1
[6/3/03 18:27:06:121 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_jspStoreDir=ToolTech
[6/3/03 18:27:06:161 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_cmd0=10001
[6/3/03 18:27:06:161 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
javax.servlet.include.path_info=null
[6/3/03 18:27:06:161 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute storeName=???
[6/3/03 18:27:06:242 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_meaFmt0=10001
[6/3/03 18:27:06:242 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_reLogonURL=LogonForm
[6/3/03 18:27:06:242 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_prc0=10001
[6/3/03 18:27:06:242 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
CommandContext=com.ibm.commerce.command.CommandContextImpl@65bcf544
[6/3/03 18:27:06:292 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
ResourceText=com.ibm.commerce.server.JSPResourceBundle@795ab55a
[6/3/03 18:27:06:302 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_cterCurr0=10001
[6/3/03 18:27:06:302 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_userId=4
[6/3/03 18:27:06:302 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute requestServletPath=/servlet
[6/3/03 18:27:06:302 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_cat0=10001
[6/3/03 18:27:06:302 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute includeDir=/ToolTech/include/
[6/3/03 18:27:06:322 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
DC_docname=ShoppingArea%2FCatalogSection%2FCategorySubsection%2FStoreCatalogDisplay.jsp
[6/3/03 18:27:06:322 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_url0=10001
```

WebSphere Commerce Cache Trace Output

```
[6/3/03 18:27:06:332 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
javax.servlet.include.servlet_path=null
[6/3/03 18:27:06:332 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_lang=-8
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute jspStoreDir=ToolTech
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_cont1=10004
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_cont0=10003
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processRequestProperties() Post-Filter- Request attribute
RequestProperties=docname=ShoppingArea/CatalogSection/CategorySubsection/StoreCatalogDisplay.jsp;storeId=10001;jspStoreDir=ToolTech;reLogonURL=LogonForm;logonPasswor
d=*****;langId=-8;logonId=buyerTW;catalogId=10001;
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processRequestProperties() Post-Filter- Request attribute
DecryptedReqProp=storeId=10001;jspStoreDir=ToolTech;reLogonURL=LogonForm;logonPassword=*****;catalogId=10001;logonId=buyerTW;
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
SessionContainer=com.ibm.commerce.browseradapter.WCSessionContainer@1513f547
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_host0=10001
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_catalogId=10001
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute
javax.servlet.include.query_string=null
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_supCurr0=10001
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute DC_langId=-8
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE d com.ibm.commerce.dynacache.filter.CacheFilter processDebug() Post-Filter - Request attribute EncodingAlreadySet=Big5
[6/3/03 18:27:06:342 EDT] 488c3560 WC_CACHE < com.ibm.commerce.dynacache.filter.CacheFilter$1 doFilter() Exit
```

Appendix A

Caching Proxy Server Installation and Configuration



To build the caching proxy server using Edge Components Version 5 and configure it as a reverse proxy, complete the following steps. Note that these instructions are based on a Windows 2000 Caching Proxy server but similar steps can be performed on other Caching Proxy server platforms.

Install the Caching Proxy Server

1. Start the WebSphere Application Server 5.0 Edge Components for Windows 2000 setup.
2. Select and install the Caching Proxy and Document (optional) components only.
3. After installation, start the caching proxy configuration wizard: Start>Programs>IBM WebSphere>Edge Components>Caching Proxy>Configuration Wizard.
4. Follow the menu instructions to setup a reverse proxy server. You will be asked to specify the ports (default 80) and target machine which will be server 2 in this case. Enter the fully qualified server path for server 2.
5. Choose a username and password for the Caching Proxy Administrator, this password will be needed to access the proxy's configuration and administrative pages.
6. Finish the setup and restart the "IBM Caching Proxy" service from the services control panel.
7. Verify that you can access the Caching Proxy Front Page by visiting `http://<server1 hostname>:8008/`

Configuring SSL

1. In order for the caching proxy to operate correctly with SSL, you must ensure that all keyrings and self signed certificates are the same as those used in the IBM HTTP Server running on the content host (server 2). The easiest way to complete this is to copy the entire ssl directory containing the keyfiles and certificates from the IBM HTTP Server directory (location is <IHS>\ssl) to the caching proxy directory (location is <EdgeDir>).

2. Now open a browser and start the Caching Proxy Server Configuration Page located at: `http://<server1 hostname>:8008/`
3. Select "Configuration and Administration Forms" and enter the username and password you created in step 5 of "Installing the caching proxy server"
4. From the left hand pane, select Proxy Configuration>SSL Settings
5. Select "Enable SSL" and "Attempt to cache content on a secure request" make sure the SSL V3 timeout is 1000 seconds.
6. Enter the locations for the "Key Ring Database" and "Key Ring Database Password" files from step 1. Typical entries are "<EdgeDir>\ssl\keyfile.kdb" and "<EdgeDir>\ssl\keyfile.sth" respectively. Ensure that there are no spaces in the path entries.
7. Click "Submit" and exit the Caching Proxy Server Configuration Page.

Configuring Proxy Settings:

Now the following manual modifications must be made to the `ibmproxy.conf` file (location is `<EdgeDir>\cp\etc\en_US\` directory). For more information about the directives described below, refer to the "Caching Proxy Administration Guide Version 5.0":

1. Search for the "HOSTNAME" directive and ensure it contains the fully qualified path for the server 1 caching proxy as entered in step 4 of "Installing the caching proxy server"
2. Uncomment the line
`"Service /WES_External_Adapter
 <EdgeDir>\cp\lib\plugins\dynacache\dyna_plugin.dll:exec_dynacmd"`
3. Find the line
`"Proxy /* server2hostname/* :80"`
 and replace it with
`"ReversePass https://server2hostname/* https://server1hostname/* "`
 (where `server2hostname` and `server1hostname` are the fully qualified paths for the server 2 and server 1 machines respectively)
4. Add the following two directives before the line
`"Pass /* <EdgeDir>\cp\server_root\pub\en_US*" :`
`"ProxyWAS /* https://wserver2hostname/* :443"`
`"ProxyWAS /* http://server2hostname/* :80"`
5. Search for the "ServerConnPool" directive and set it to ON.

Appendix B

WebSphere Application Server Level Requirement

Note: These levels will change, please always use the latest WEBSPPHERE APPLICATION SERVER fixpack or cumulative fixpacks

WAS Version	efixes, ifixes, cumulative fixpacks
5.0.0.0	WAS_Dynacache_07-16-2003_5.0.0_cumulative
5.0.1.0	WAS_Dynacache_09-26-2003_5.0.1_cumulative
5.0.2.0	PQ79710 PQ81651
5.0.2.1 5.0.2.2 5.0.2.3	Note: 5.0.2.1 denotes 5.0.2.0 with cumulative fixpack 1, etc

Important: In addition to the efixes above, please also install WAS_Plugin_02-03-2004_5.0.X_cumulative for the latest plugin updates.

Notices

Note to U.S. Government Users -- Documentation related to restricted rights -- Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp. This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independent created programs and other programs

(including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Lab Director
IBM Canada Ltd. Laboratory
8200 Warden Avenue
Markham, Ontario
L6G 1C7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This document may contain information about other companies' products, including references to such companies' Internet sites. IBM has no responsibility for the accuracy, completeness, or use of such information.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 1996, 2004. All rights reserved.

Trademarks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX
CICS
CrossWorlds

DB2
DB2 Universal Database
Domino
Encina
eServer
IBM
IBM (logo)
iSeries
Lotus
OS/390
OS/400
QuickPlace
S/390
SecureWay
WebSphere
z/OS
zSeries
400

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation, in the United States, other countries or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.