IBM Software Group

IBM WebSphere® Application Server V5.1.1

Web Services Client Cache

business on demand.

Goals

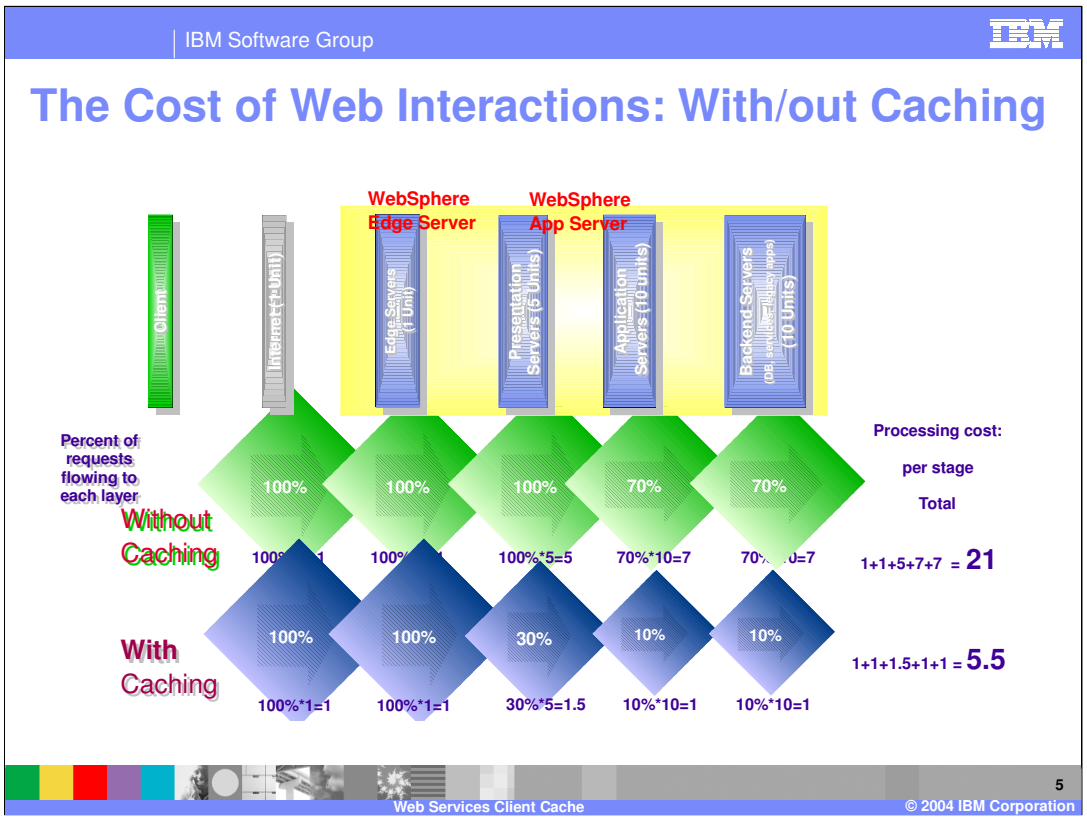
- Describe the implementation and administration of new Dynacache feature for Web Services in V5.1.1

Agenda

- Overview
- Architecture / Big-Picture
- Details
- Example
- References

Section

Overview

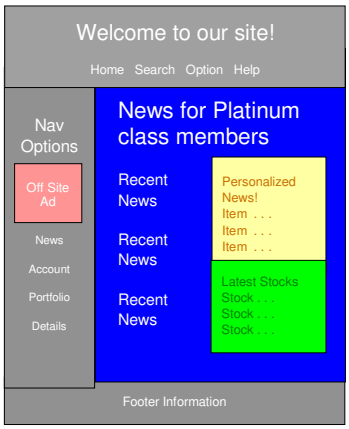


Here you have the classic example of performance benefits when using a cache.

Requests to all target endpoints without a cache policy will pay only a very minimal penalty because of a client cache in the flow. Target endpoint address is retrieved from the MessageContext and the cache handler will lookup for a cache policy in the configuration manager. If one is not found, the request moves to the next system handler in the chain.

Web caching with DynaCache

- WebSphere can cache fragments of pages, notably individual Servlet and JSP requests
- Reduces both load and response time
- Performance gains with:
  - Static Fragments
    - header JSPs, navigation bars, etc.
  - Dynamic Fragments/Pages
    - stock quotes, search results, levels of service
    - personalized pages using shared information (e.g. MyNews)
- Application developers control how fragments are cached
  - Define rules based on Servlet, URI, request/session variables, etc.
- Rule-based, time-based, and programmatic techniques for invalidating cache entries
- Can control external caches, e.g. IBM Edge Server



WebSphere's implementation of caching is called DynaCache. Caching entire pages is relatively easy, caching portions of a page is trickier.

Not done programmatically, all done through XML files. Makes it easier to do fine grained tuning on your application without having to change the application code.

Rules based, determines how long and when items will cached.

What's New?

- WebSphere V5.0 can cache
  - Servlets
  - JSPs
  - Java objects
  - Commands
  - Server-side web services
- V5.1.1 seeks to improve performance of JAX-RPC clients running within the Application Server by adding support for Dynacache to the Web Services client

Websphere 5.0 and 5.1 already contained capability to cache certain resources. 5.1.1 and 6.0 are adding the capability to apply a cache when WebSphere is running the client application.

Client Caching

- Increases the performance of Web Services clients by caching responses from remote Web Services
  - Once a response is cached, subsequent calls to the same Web Service with the same set of request parameters could be responded from cache
- Provided as a JAX-RPC handler that operates on a proxy server
  - Based on the policy specified in the cachespec.xml file

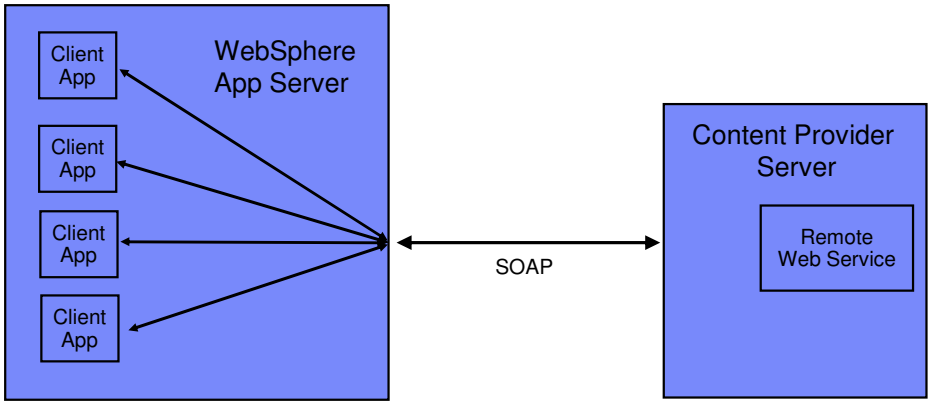
Cache JAX-RPC handler needs to identify a cache policy based on the target web service. Once a policy is found, all the cache id rules specified in the policy will be evaluated one by one until a valid rule is detected.

Once a response is returned from a remote Web service, the response is saved in the client cache on the application server. Any identical requests that are made to the same remote Web service are then responded to from the cache for a specified period of time. The Web services client cache relies primarily on time-based invalidations because the target web service may be outside of your enterprise network and unaware of your client caching. Therefore, you can specify the amount of time in the cache and the rules to build cache entry IDs in the cache in your client application.

Scenarios

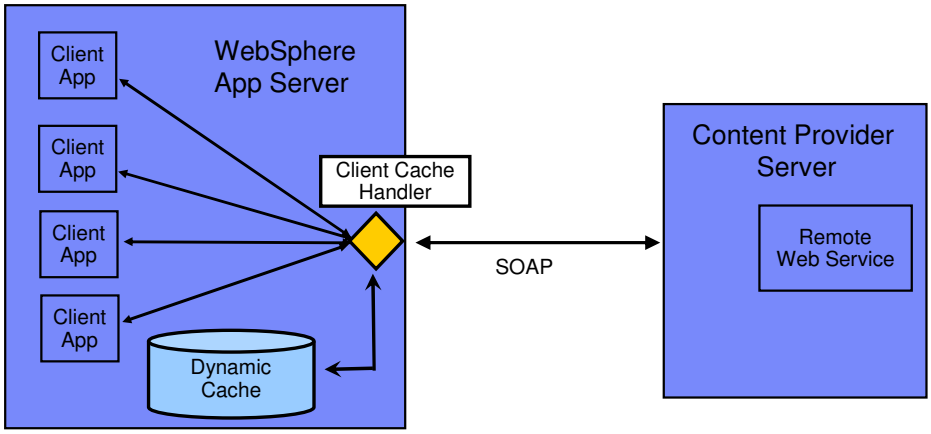
- Advantages of using cache in an Enterprise environment
- Split-Tier setup
  - Both client and server running WAS
  - Using a Reverse Proxy Server

External Content Provider Scenario no Cache

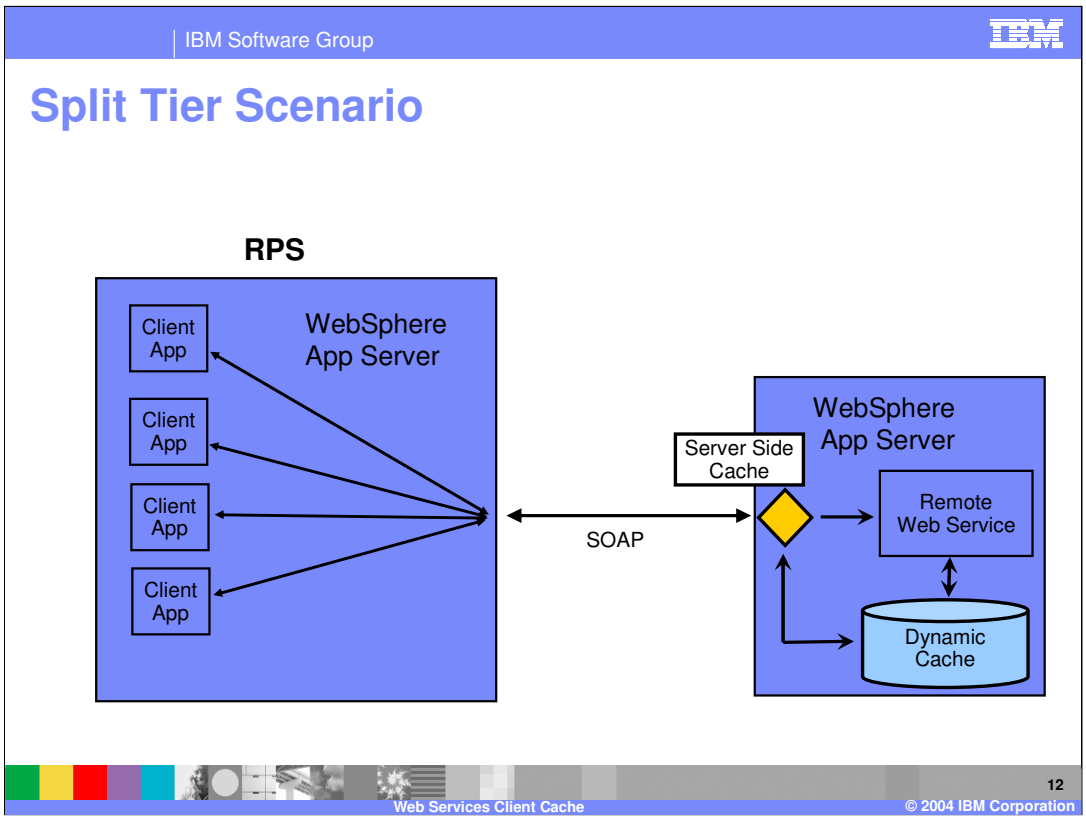


In this example the service provider is on the internet. Without caching all calls must travel over the internet. Thus calls to the service can be expensive. Depending on how many repetitive calls are being made, the performance cost could be very high.

External Content Provider Scenario with Cache



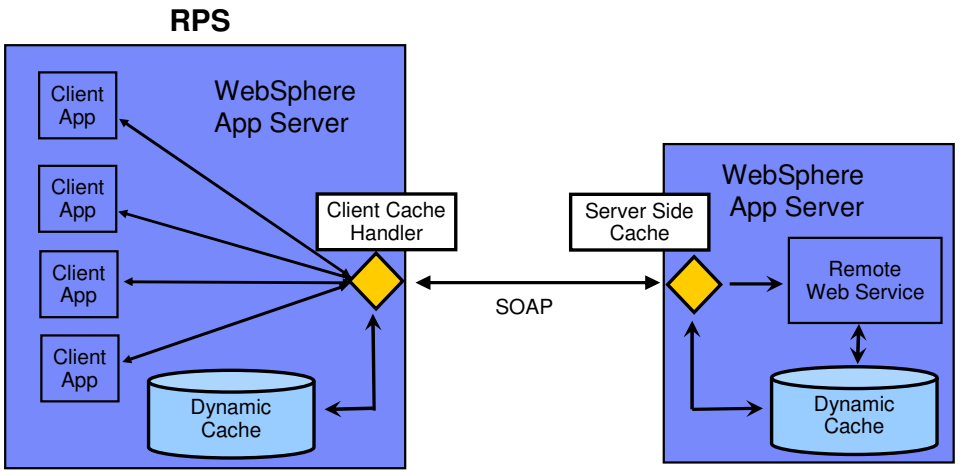
In this example we build upon the last example by add a client side cache. The cache will be checked before calling the remote service. This can result in significant performance increase, by preventing unnecessary calls to the service provider.



This type of caching is very beneficial in a Reverse Proxy scenario where a Web Services Gateway could respond to requests from cache instead of invoking backend services.

This shows the already provided provider side cache capability. Which helps improve performance by checking a cache on the target server. This will result in a performance increase, as objects may not need to be instantiated. Requests must sill pass through the RPS to the service provider however.

Split Tier Scenario with Cache

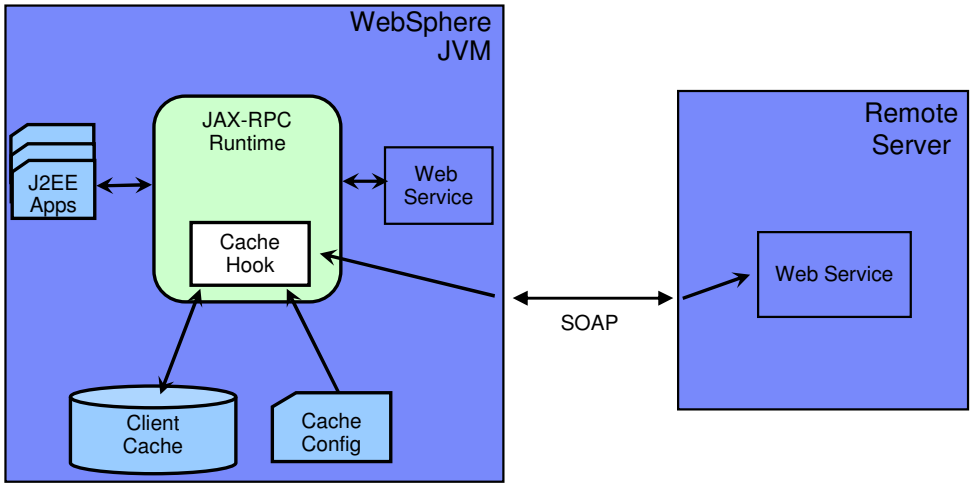


We can increase performance further by providing a second cache within the RPS. This cache will be checked before a call is made to the actual server provider allowing for the request to be fulfilled from cache. This type of cache can provide a greater performance increase for certain scenarios, and is compatible with a provider side cache as well.

Section

Architecture / Big-Picture

Architecture: Big Picture



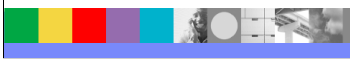
* A small performance penalty is paid to check the cache policy on each invocation

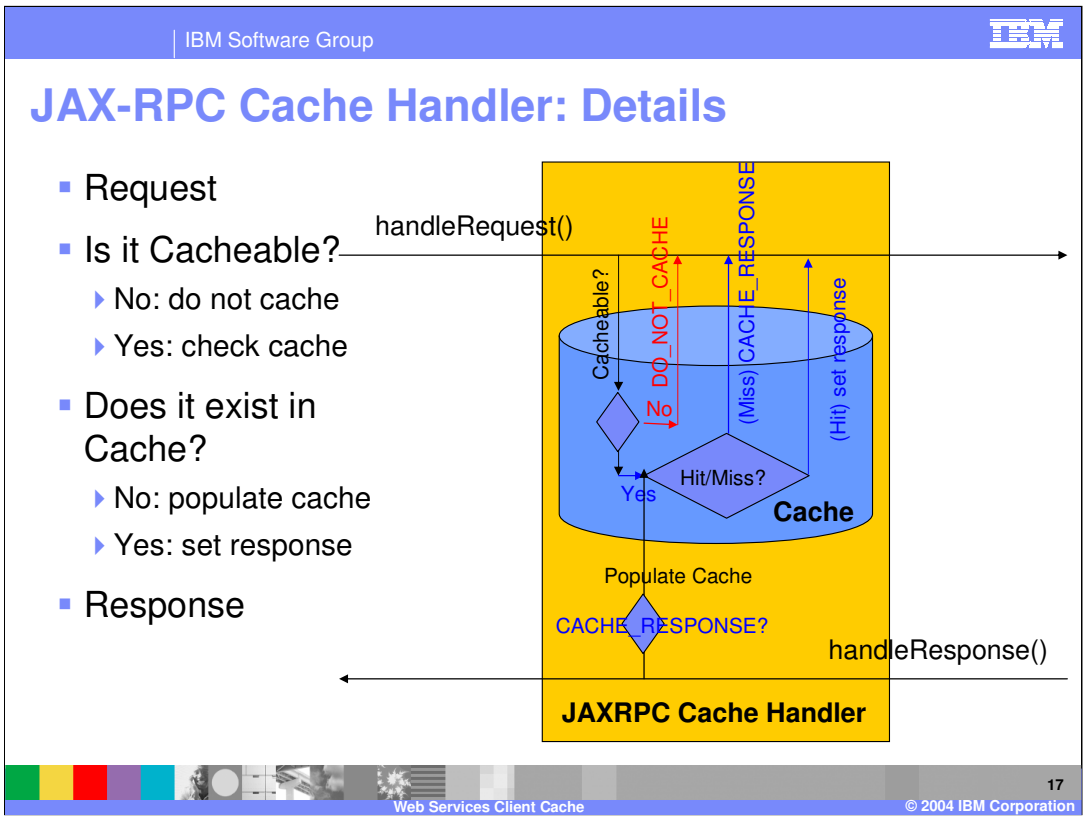
Within the JVM the JAX-RPC runtime has a hook to the caching service. When a client request comes into the RPC runtime, it is intercepted by the cache handler that checks the cache based on rules found in the cache config XML file. If it doesn't find the information in the Cache, then it will either call the web service within the same WebSphere server, or forward the call on to the target service located elsewhere. The Web Service can be local or remote to this server. The result would be placed in the cache before being returned to the client.

The JAX-RPC Cache handler needs to identify a cache policy based on the target web service. Once a policy is found, all the cache id rules specified in the policy will be evaluated one by one until a valid rule is detected.

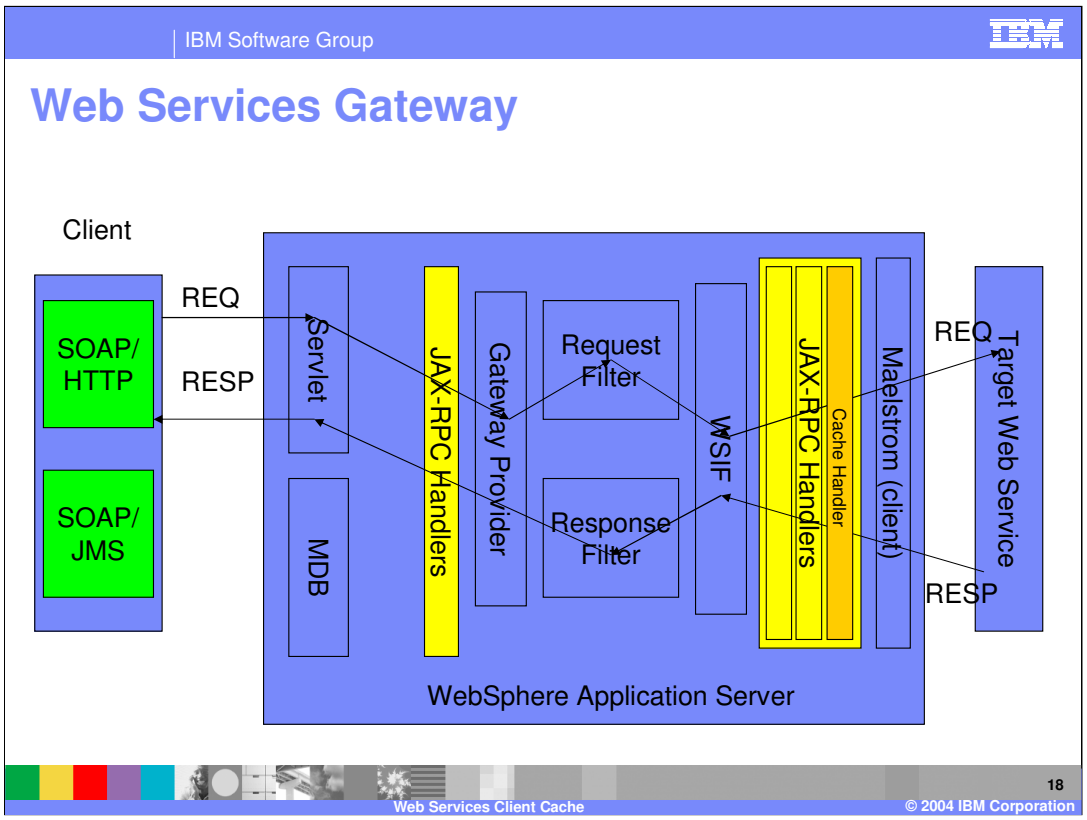
Section

Details





Web services client caching is provided as a JAX-RPC cache handler. In the handleRequest() method, cache config manager is searched for a cache policy based on the target endpoint address specified in the request. Request is not cacheable if a matching cache policy is not found. If a matching policy is found, all the cache id rules in that policy are executed one by one until a valid rule is identified. Result of the first valid cache rule will be the cache key for lookup. If this lookup ends in a cache miss, a property is added to the handler chain's message context to cache the response in handleResponse() method. If this lookup ends in a cache hit, the value from the cache is set as the response and the rest of the request handle chain is blocked. If a SOAP fault is returned, the response is not cached. Else it will be cached in handleResponse() method using the cache key specified in the message context.



This shows how the Cache Handler fits within the Web Services Gateway architecture. With the gateway functioning as an RPS. Cache handlers operate like any other JAX-RPC handler installed into the Gateway.

Enabling JAX-RPC Cache

- JAX-RPC caching is enabled if Dynamic cache Service is enabled.
- Configure caching policy in cachespec.xml
  - New type of config entry “JAXRPCClient”
  - Supporting new types “part” and “operation”
- The cachespec.xml file is found inside the WEB-INF directory of a Web module.
  - Can place a global cachespec.xml file in the application server properties directory
  - Or place the cache configuration file with the deployment module - Recommended

Section

Example

Cachespec.xml and Cache Ids

- Cache IDs are used to reference entries in the cache
- Cache id from SOAP header entries
  - Best performance
- Cache id from SOAP envelope
  - Hash code
- Cache id from SOAP Body
  - Operation and Part
  - Allows highest level of granularity

There are several ways to specify the way information gets cached. Particularly in the creation of cache ID's which are used by the WebSphere server to store and identify the values stored in cache. WebSphere can create the cache Id a number of ways. From the SOAP header, this is best from a performance perspective, as the body of the SOAP message doesn't need to be parsed in order to determine if the message is in the cache, but this is fairly course grained. The other options allows finer granularity and control over what is cached.

Sample WSDL

```
<definitions targetNamespace=http://TradeSample.com/…>
    <message name="getQuoteRequest">
        <part name="symbol" type="xsd:string"/>
    </message>
    .....
    .....
    <binding name="SoapBinding" type="tns:GetQuote">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="getQuote">
                <soap:operation soapAction=""/>
                <input name="getQuoteRequest">
                    <soap:body namespace="http://TradeSample.com/" use="encoded"
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                ......
                </operation>
    </binding>
    <service name="GetQuoteService">
        <port binding="tns:SoapBinding" name="SoapPort">
            <soap:address location="http://TradeSample.com:9080/service/getquote"/>
        </port>
    </service>
</definitions>
```


Here we see portions of an example WSDL for a stock quote service. It contains a method for getQuote, which requires a parameter ‘symbol’ which would be the stock name like IBM. The various bolded areas are information you would need for cache ID’s

Sample SOAP Request

```
POST /wsgwsoap1/soaprpcrouther HTTP/1.1
….
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope …>
<soapenv:Header>
  <getQuote soapenv:actor="com.ibm.websphere.cache">
    IBM
  </getQuote>
</soapenv:Header>
<soapenv:Body … >
  <getQuote xmlns="urn:ibmwsgw#GetQuoteSample">
    <symbol xsi:type="xsd:string">IBM</symbol>
  </getQuote>
</soapenv:Body>
</soapenv:Envelope>
```


This is a simplified SOAP request, for the example stock quote service. Showing the SOAP envelope and header as well as the parameters for the service. With the actor for the value stored in the cache being IBM. And the SOAP body with the getQuoteSample information and the string attribute IBM.

Cache ID from SOAP Header

```
<cache>
  <cache-entry>
    <class>JAXRPCClient</class>
    <name>http://TradeSample.com:9080/service/getquote</name>
    <cache-id>
      <component id="getQuote" type="SOAPHeaderEntry"/>
    </cache-id>
  </cache-entry>
</cache>
```

- Cache ID is **http://TradeSample.com:9080/service/getWQuote:getQuote=IBM**

With this example of a cache entry using the SOAP header to create the cache id, we see the cache entry class is JAXRPCClient. The name is the tradesample service getquote binding. The cache id generated from this is shown on the bottom. So from this cache entry example you would cache a response to the getQuote method for IBM.

Cache ID from SOAP Envelope

```
<cache>
  <cache-entry>
    <class>JAXRPCClient</class>
    <name>http://TradeSample.com:9080/service/getquote</name>
    <cache-id>
      <component id="hash" type="SOAPEnvelope"/>
      <timeout>60</timeout>
    </cache-id>
  </cache-entry>
</cache>
```

Cache ID is
http://TradeSample.com:9080/service/getquote:Hash=<*xxxHashSoapEnvelope*>

This is an example of getting the information from the SOAP envelope. This performs slightly worse then the SOAP header example, as it requires some parsing of the SOAP message to retrieve this information. The name is the SOAP port coming in, and we are saying we want a hash on the SOAP envelope. As we see in our id value that is created containing the hash for the soap envelope.

Cache ID from SOAP Body

```
<cache>
  <cache-entry>
    <class>JAXRPCClient</class>
    <name>http://TradeSample.com:9080/service/getquote</name>
    <cache-id>
      <component id="" type="operation">
        <value>http://TradeSample.com/:getQuote</value>
      </component>
      <component id="symbol" type="part"/>
    </cache-id>
  </cache-entry>
</cache>
```

- Cache ID is
  http://TradeSample.com:9080/service/getquote:operation=http://TradeSample.com/:getQuote/symbol=IBM

This is an example showing how to create a cache id from the SOAP body. This method allows the greatest control in selecting what is cached, but also requires the largest performance penalty as the entire XML message must be parsed by the cache to retrieve this information. This would allow you to cache certain portions of an XML message that will be common across multiple service requests.

Section

Summary

Summary

- Introduced enhancements to Dynacache in V5.1.1
- Discussed
  - Performance benefits
  - Architecture
  - Example of caching a Web Service

Section

References

References

- WebSphere V5.1.1 Information Center
- "Caching In" – WebSphere Journal
  - http://sys-con.com/story/?storyid=44291&DE=1
- **IBM WebSphere V5.0 Performance, Scalability, and High Availability**
  - http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg246198.html?Open
- Developing and Deploying Command Caching with WebSphere Studio V5
  - http://www-106.ibm.com/developerworks/websphere/registered/tutorials/0306_mcguinnes/mcguinnes.html

Section

Appendix

Trademarks and Disclaimers

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

IBM	iSeries	OS/400	Informix	WebSphere
IBM(logo)	pSeries	AIX	Cloudscape	MQSeries
e(logo)business	xSeries	CICS	DB2 Universal Database	DB2
Tivoli	zSeries	OS/390	IMS	Lotus

Java and all Java-based trademarks 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. Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are
trademarks of Intel 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. Linux is a registered trademark of Linus Torvalds. Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectually property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. 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 in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

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

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

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 and IBM Corp.