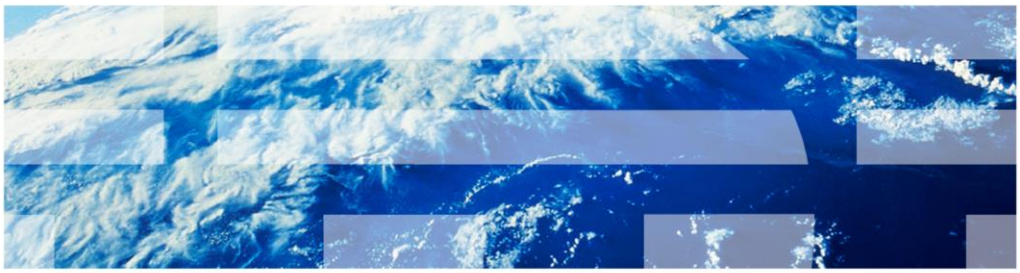
IBM

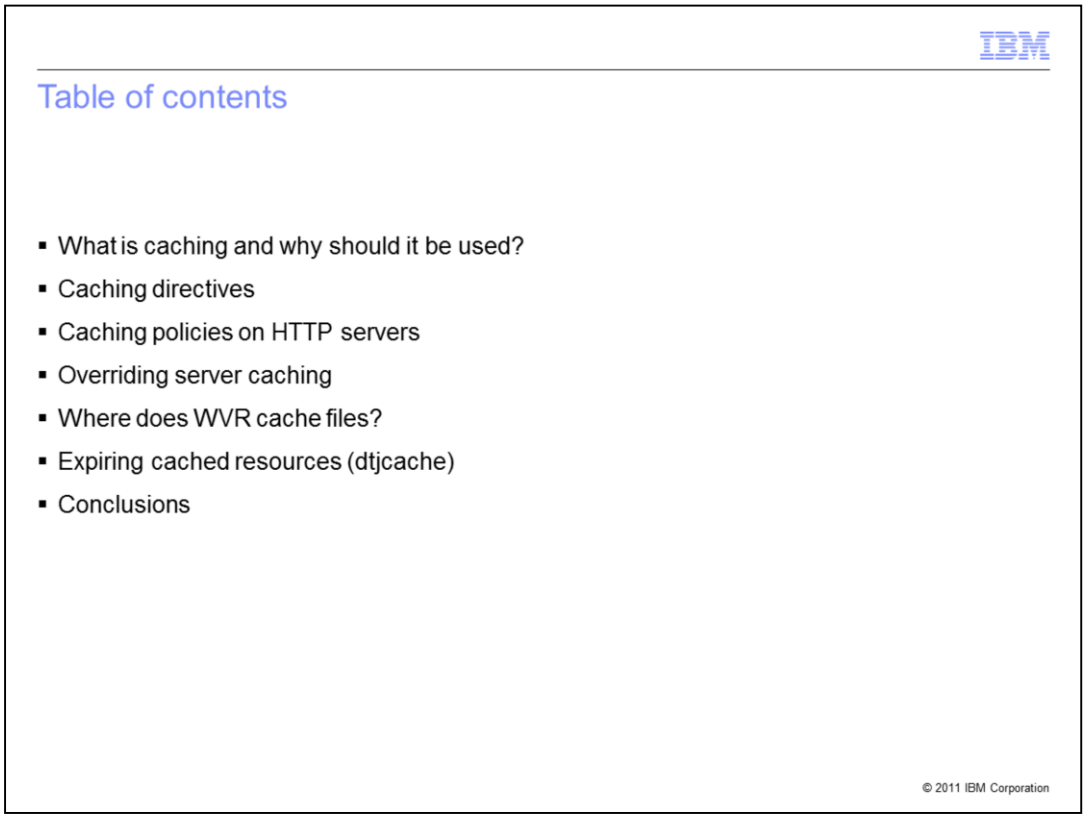
WebSphere Voice Response for AIX

Understanding caching in WebSphere Voice Response

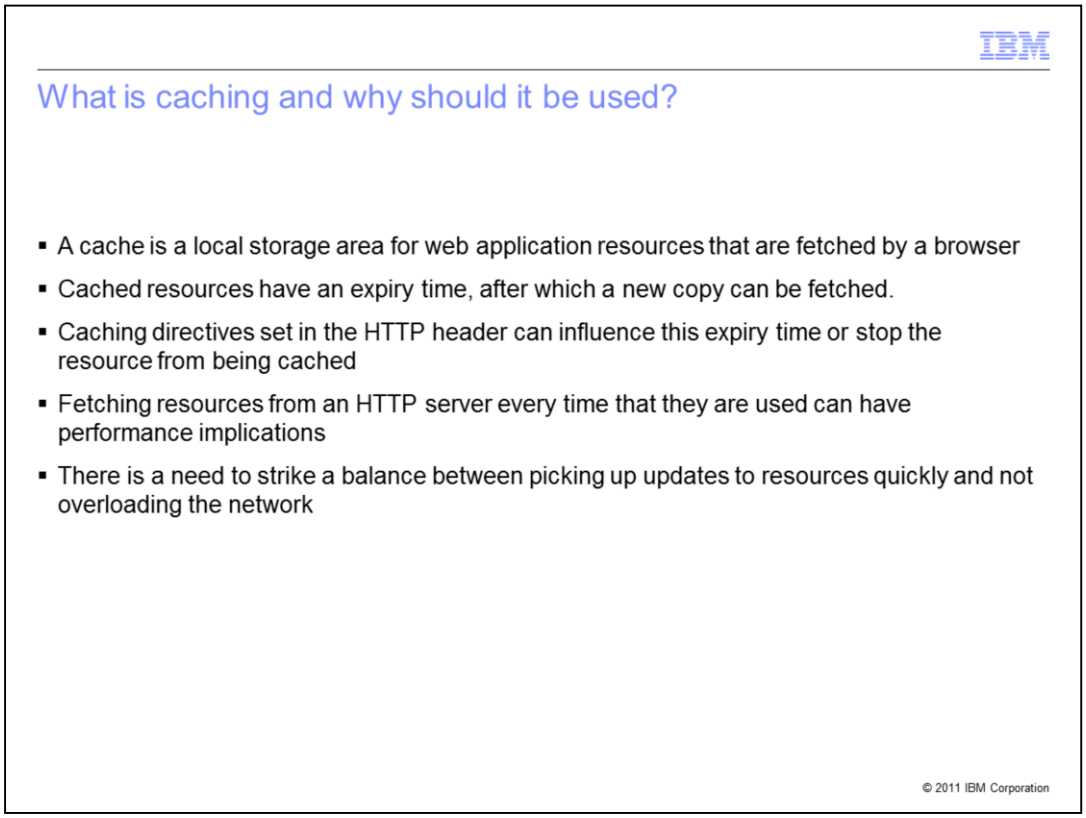


This presentation will describe how caching directives can be set up and used for WebSphere Voice Response applications to achieve the best performance possible.

WebSphere Voice Response is referred to as WVR throughout these slides.



In this presentation, we will explain the concept of caching, what the different caching directives mean, how to set those directives and where WVR caches the files that it fetches. We will then go on to discuss the dtjcache tool introduced into WVR with Fix Pack 2.

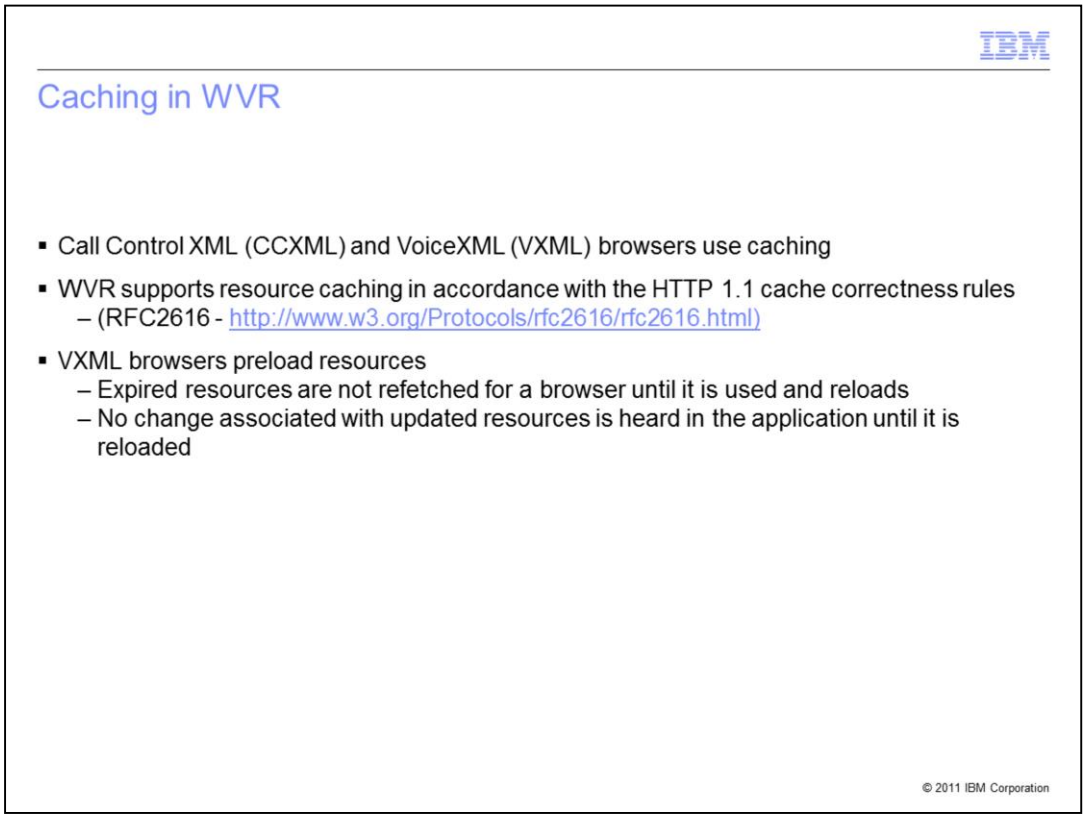


A cache in the context of a web application is a local storage area for resources that are fetched from either a local file system or an HTTP server.

The cache can either be in memory, written to disc or both. Cached resources may eventually expire, after which a new copy of the resource should be fetched.

If a browser fetches all required resources for an application every time that an application is started, then the browser can take a long time to load. This can be problematic for Voice browsers as any delay in fetching or downloading resources can result in the caller hearing an extended period of silence, which can be perceived as an error by the caller. This behavior can also impact network and HTTP server performance due to high numbers of large fetch requests.

Caching directives can be set to influence the time at which a resource will expire, so it is important to work out a balance between application performance and currency of resources. Resources that are unlikely to change very often should be allowed to be cached for a reasonably long period, whereas resources more liable to change should have a shorter cache expiry time.



CCXML and VXML can be used to develop powerful and complex applications for WVR.

As previously mentioned, Voice browsers have a particular need to cache items to ensure a decent level of application, network, and HTTP Server performance.

When a WVR application fetches a resource, WVR first checks the relevant cache to see if there is an entry for that resource. If there is an entry for that resource, then WVR checks to see if it has expired. If there is a cache entry and it has not expired, WVR can use that cached resource instead of making a fetch request. If the cache entry has expired, WVR performs a fetch-if-modified-since request. As a result, the web server can respond with a new copy of the resource or a response indicating that the cached version of the resource should still be used.

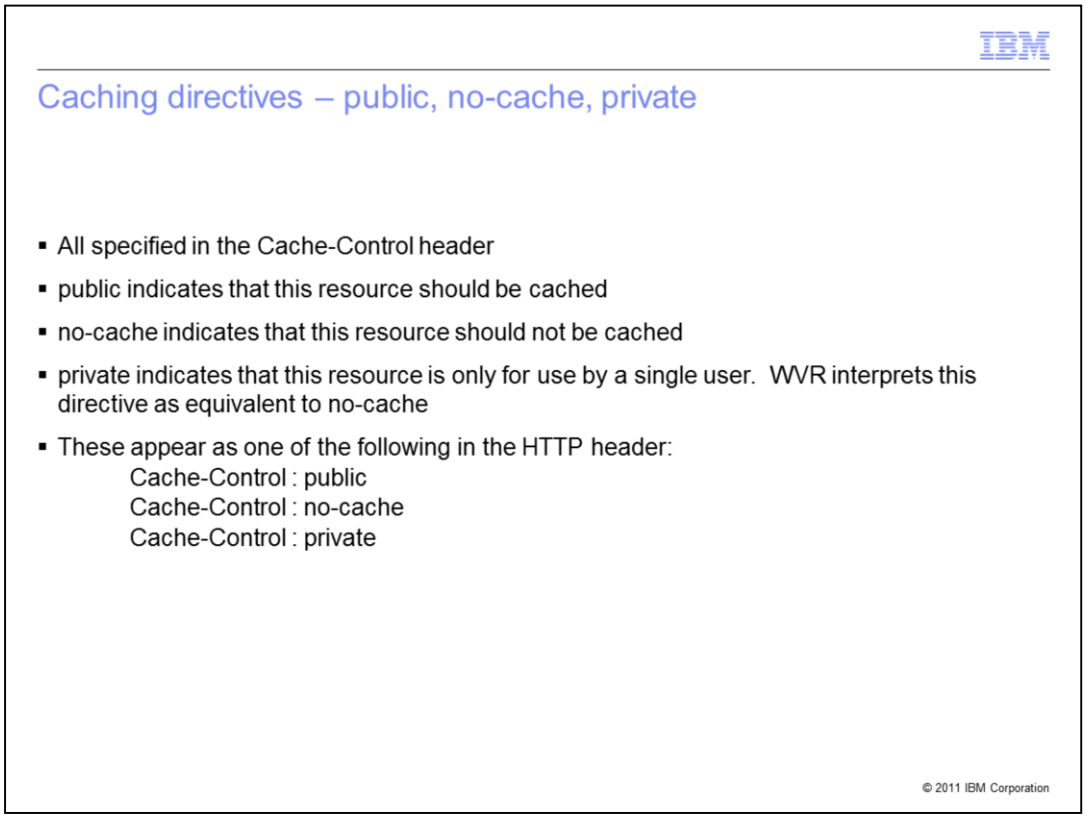
Resources can be directed to expire quickly if they are likely to change regularly in order to avoid using outdated cached versions of those resources.

VXML applications load their resources when the browser starts up and waits for a telephone call – this is called preloading. WVR does this in order to respond quickly when a call arrives. This can mean that an application uses expired resources for one call – when that call ends, all resources for that application are refetched. Application resources are only refetched when a call ends and the browser is reloaded, ready to wait for a new call.

A caching directive is a mechanism for an HTTP server to tell a browser what caching method to use for a resource. They are either set as an HTTP header or as a value of the Cache-Control HTTP header.

There are several directives built into HTTP to enable caching.

The next few slides will explore several of these caching directives and explain their meaning. Three of these fields are defined in the version 1.0 HTTP specification – these are Date, Last-Modified and Expires. The remaining six are values of the HTTP 1.1 Cache Control header: these are public, private, no-cache, must-revalidate, max-age and max-stale.



The public, no-cache and private cache directives are all specified in the Cache-Control HTTP header.

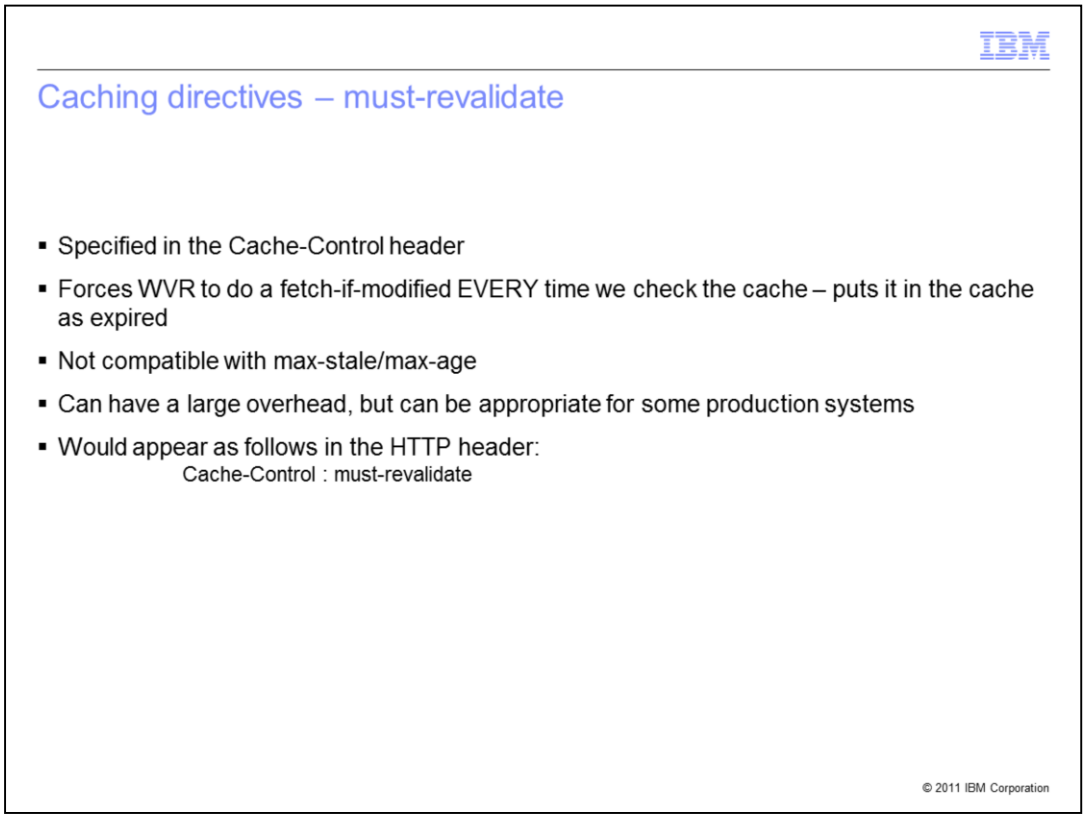
The public directive instructs a browser fetching a resource that it can be cached according to any other directives specified, such as max-age. If no other directives are specified, it is left to the browser to decide the expiry time of the resource.

The no-cache directive instructs a browser fetching a resource that it should not be stored in the cache. As the resource is not stored in the cache, a local copy cannot be reused and so a new copy of the resource is fetched every time WVR needs to use it.

The private directive instructs a browser that the resource is intended for use by a single user, and that other users should not use the cached resource. WVR interprets this directive to have the same effect as using no-cache.

The example lines show how these directives appear in an HTTP header. Only one Cache-Control header should be defined. If more than one Cache-Control header is defined, the last one is used.

Using the no-cache or private directives has performance implications. It is strongly advised that you only do this for resources that are liable to regularly change or that otherwise would not be appropriate for caching.



The must-revalidate directive is specified in the Cache-Control HTTP header.

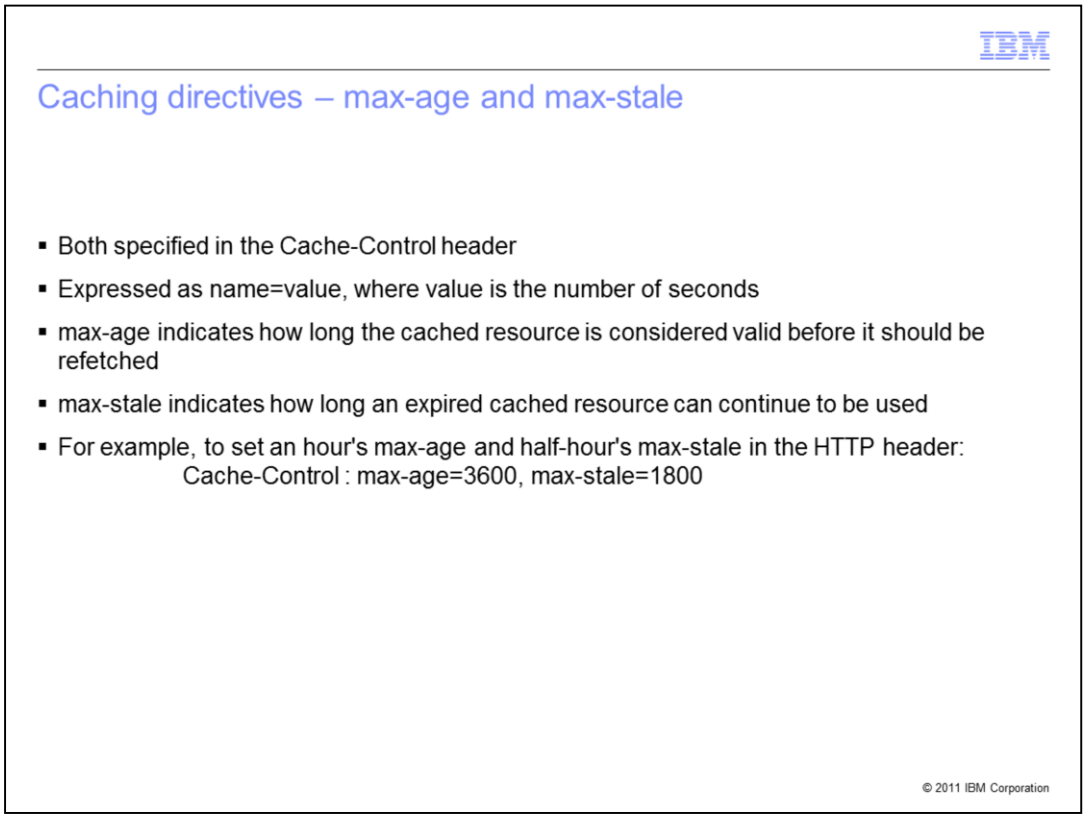
The must-revalidate directive instructs a browser fetching a resource that it can be cached but that the cache must check if the resource has changed every time that the resource is accessed. This is equivalent to adding it to the cache but making it expired immediately.

WVR performs a fetch-if-modified-since action to get the resource. The web server can respond with a new copy of the resource or a response indicating that the cached version of the resource should still be used.

Due to max-stale and max-age influencing the expiry time of a resource, they are not compatible with must-revalidate.

Using this directive adds a performance overhead as an HTTP-Fetch is performed every time that a resource is accessed. However, this may not be prohibitive as a large percentage of those fetches will only contain the response to continue using the cached version, rather than containing the entire resource. Thus must-revalidate may be appropriate for some production systems where the network is sufficiently fast.

The example line shows how this directive appears in an HTTP header



Max-age and max-stale are both specified in the Cache-Control HTTP header.

These directives allow an HTTP server control over how old a resource is before it expires and WVR must refetch it.

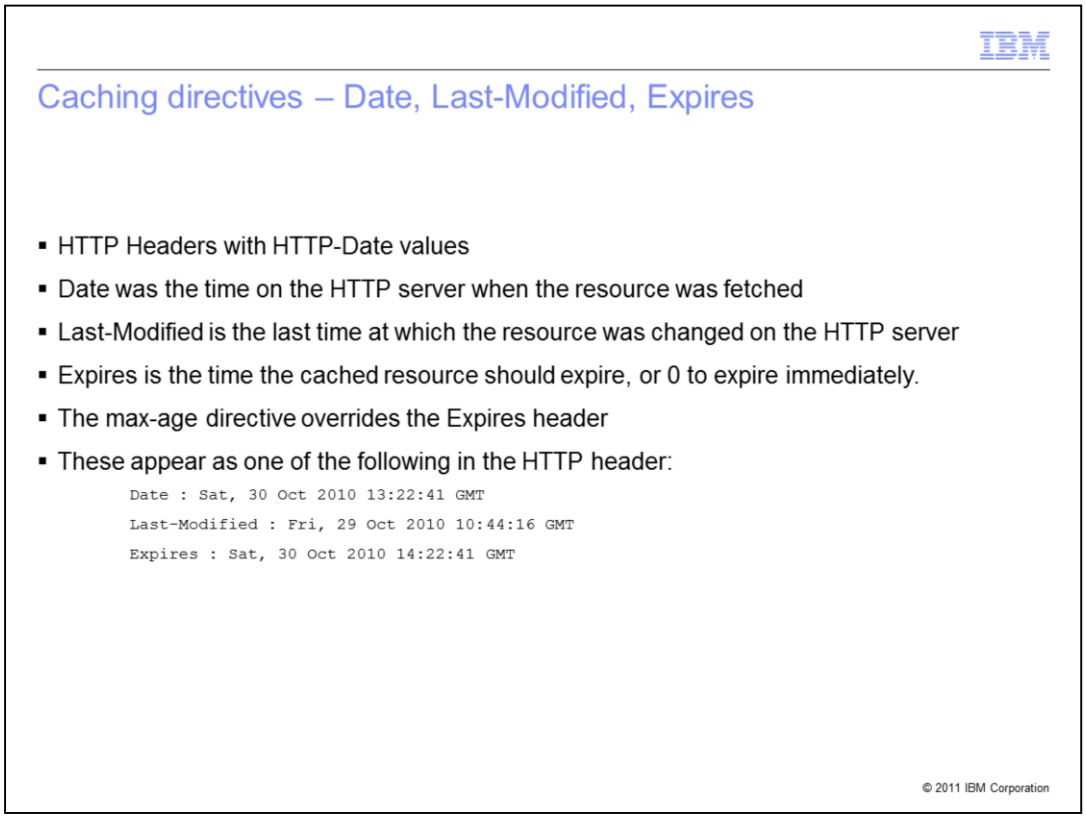
The max-age value indicates how long, in seconds, it should be before you consider a resource expired and should refetch the resource.

The max-stale value indicates how long expired resources can continue to be used before they must be refetched.

An hour's max-age and a half-hour's max-stale are given as an example. Note that these should be combined onto a single line instead of having two Cache-Control lines in the HTTP header as there can only be one Cache-Control directive in an HTTP header.

Note that other Cache-Control headers can be combined onto a single line – it is not only max-age and max-stale.

In practice, WVR treats max-stale as an extension to the length of max-age, so the example effectively expires after an hour and a half.



Expires, Date and Last-Modified are their own HTTP headers, and each can have an HTTP-date value associated with them.

Date indicates the date-time on the HTTP server when the resource was fetched. It's primary use is to work out the time difference between the two systems and determine the actual age at which an Expires header applies.

Last-Modified indicates the last time that this resource was altered, and is used by the HTTP server on a fetch-if-modified-since request to determine if a new version of the resource is sent or a response to indicate that the cached version should be used

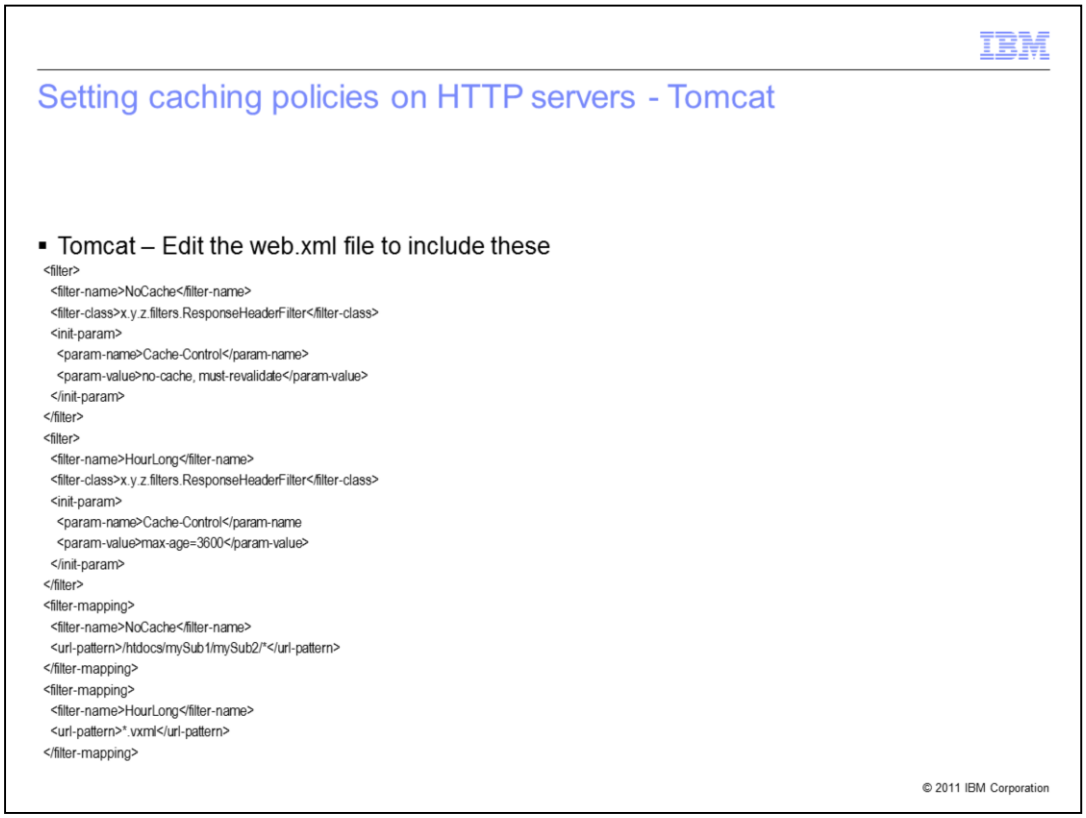
Expires indicates the absolute time at which this resource should expire. If supplied without the Date header, the browser has to assume that the HTTP server that the browser has fetched this resource from is operating on the same clock time as the browser. In effect, the browser works out the time between the fetch time and the expires time and uses that as the max-age value.

A special case is made for Expired content values that do not conform to date standards - they are assumed to indicate an immediately expired resource. This is conventionally expressed as Expires=0, but this will have the same effect as Expires=20 or Expires=Never. It is recommended to use Expires=0 for this purpose.

Note that if the Expires and max-age directives are both specified, max-age overrides Expires. Thus it is recommended to use the max-age or no-cache directives rather than the expires directive in the majority of cases.

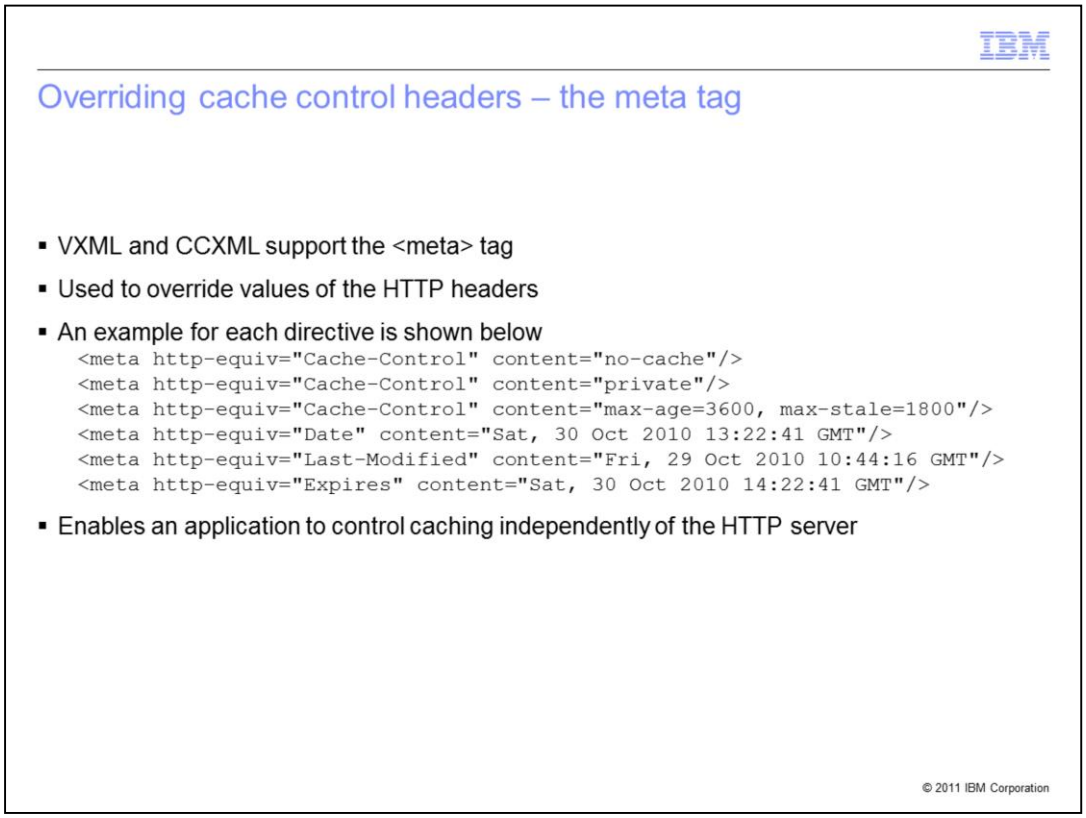
The previous slides have shown the various types of caching policy that can be set in HTTP headers. Consult your HTTP server's documentation for how to set these values. It is possible to set differing caching policies for directories on the server, and set general policies based on the file extension.

Here is an example for the IBM HTTP Server – to use it, edit the httpd dot conf configuration file and restart the IBM HTTP Server. This particular example sets up an hour's maxage for all dot vxml files and no-cache for the sub-directory htdocs slash mySub1 slash mySub2.



Here is an example for configuring a Tomcat HTTP server – to use it, edit web dot xml to add the example XML and restart Tomcat. This particular example sets up an hour's maxage for all dot vxml files and no-cache for the sub-directory htdocs slash mySub1 slash mySub2.

Note that you will need to alter the example to match the relevant Java classes first – consult your HTTP server's documentation for more details.

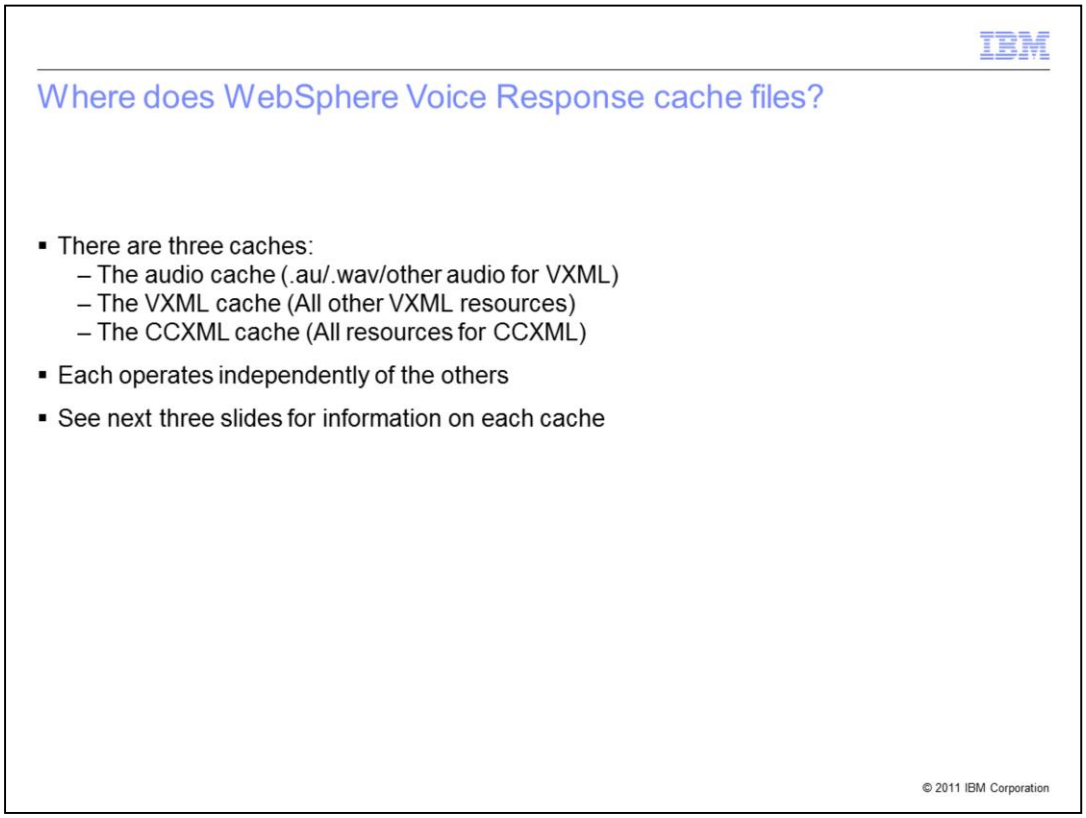


Sometimes application developers will want to override the normal caching policies on an HTTP server for a particular resource. While developers can update the configuration resources for that server to make an exemption for one specific resource, it is easier to use the VXML or CCXML meta tag to alter the caching behavior for just this resource.

This will override any matching directives that the HTTP server sent in the response header – if the HTTP server sets the Expires and Cache-Control headers and the meta tag only overrides Expires, then the value of Cache-Control remains unchanged.

The example shows how to use the meta tag in your VXML or CCXML documents - in each case, you should add the element as a child of the root element. This is the vxml element in a VXML document and the ccxml element in a CCXML document.

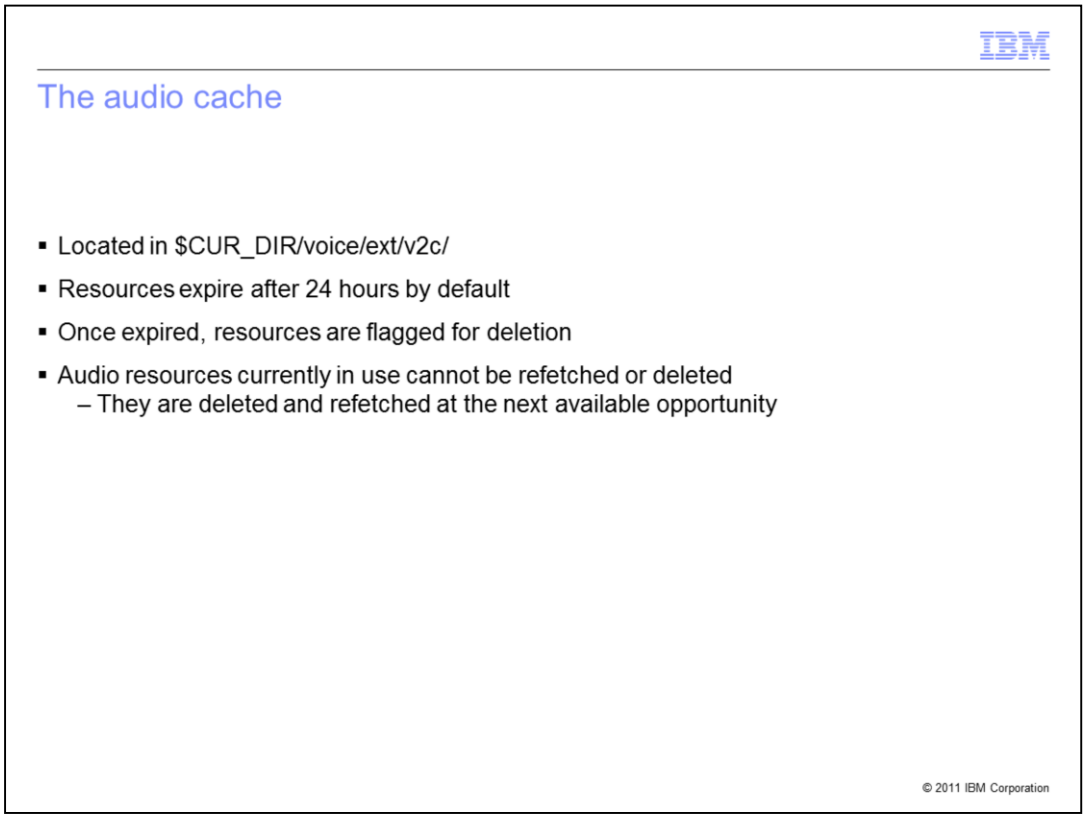
This is particularly useful when developing an application - specifying no-cache will ensure that each test runs against the latest version of the application's resources. You should remember to review any meta tag values before deploying an application, to ensure caching behaviour meets your production requirements.



There are three independent WebSphere Voice Response caches, which are used based on the type of resource fetched.

These are the audio cache, which stores all audio resources including dot au and dot wav ; the VXML cache which stores the VXML applications and all non-audio resources referenced in the VXML applications; and the CCXML cache, which stores the CCXML applications and all resources referenced in the CCXML applications excluding VXML documents

Each of these caches behaves differently, and operates independently of the other caches. The following slides will explain each cache in detail.



The audio cache contains all audio resource fetched by VXML applications.

These resources are separated from the other resources in the VXML applications due to the larger size of the cached items.

Audio resources are stored in their original format and can be converted by WVR into different formats depending on the characteristics of the channel that the call is handled on.

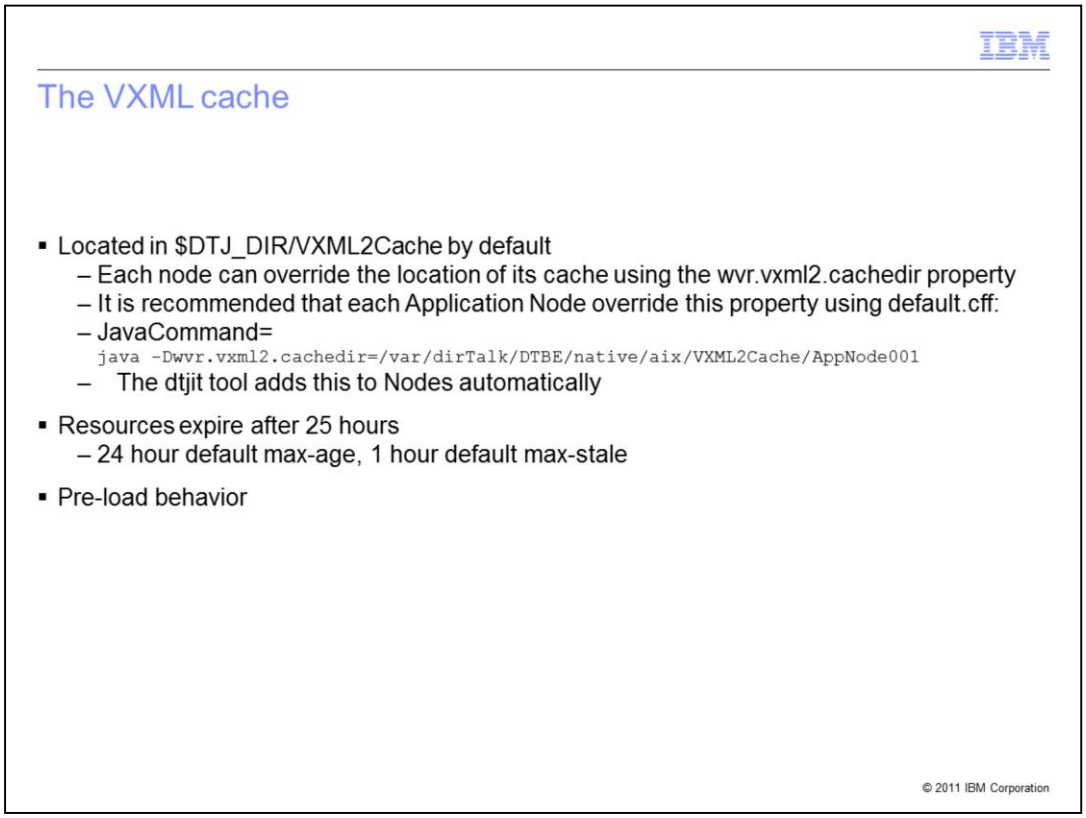
Audio resources have a default expiry time of 24 hours.

When a resource in the audio cache expires, it is flagged for deletion. Once deleted, the next request for that resource will fetch a new copy of the audio resource from the HTTP server and place it in the cache.

However, WVR cannot do this for items currently in use by VXML applications. Thus it may take some time for an expired audio resource to be deleted and refetched. In a particularly busy system this may take several hours.

Note that WVR does not perform a fetch-if-modified-since request for audio resources because the resource is physically deleted and so WVR has no cached resource to reference.

The audio cache is persistent over WVR restarts.



The VXML cache stores the non-audio resources fetched by VXML applications, including the VXML, grammar and script resources.

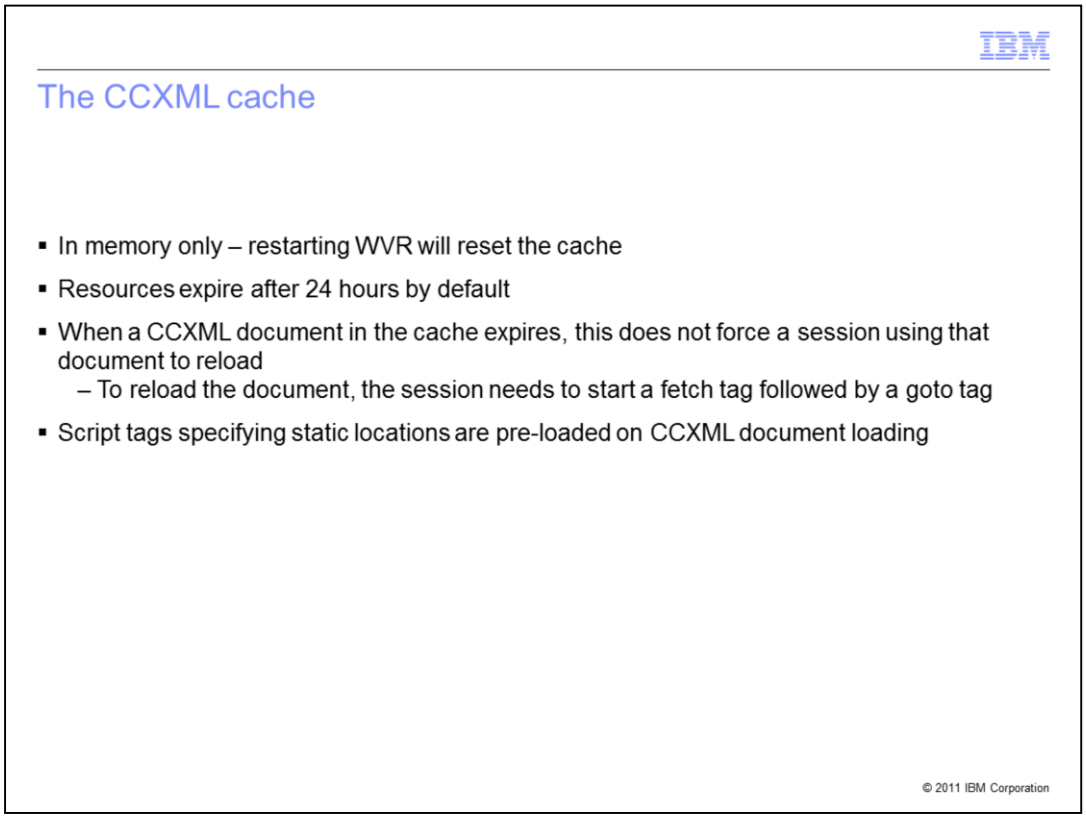
By default the VXML cache for each application node is set to dollar DTJ underscore DIR slash VXML2Cache, but can be overridden using the wvr dot vxml2 dot cachedir java property. This means that by default all application nodes use the same directory for their VXML cache.

It is recommended that on a production system each application node uses its own VXML cache directory to reduce the risk of file lock deadlocks between VXML browsers. This property can be set using the JavaCommand line in default dot cff as shown. The dtjit configuration tool will set the JavaCommand line automatically.

Resources in the VXML cache have a 24 hour max age and a 1 hour max stale by default. This gives an effective default expiry time of 25 hours. If different cache directories have been set for each application node, then each node's cache is independent and can expire at different times.

As previously mentioned, VXML browsers preload their content when they first load or reload after a call, so some cycling of browsers is necessary before changes to cached resources are universally reflected. This does not apply to dynamic URI VXML browsers used by CCXML.

The VXML cache will persist after a WVR restart.



CCXML does not maintain a persistent cache on the file system, only an in-memory cache. Thus, stopping and starting WVR will always clear the CCXML cache.

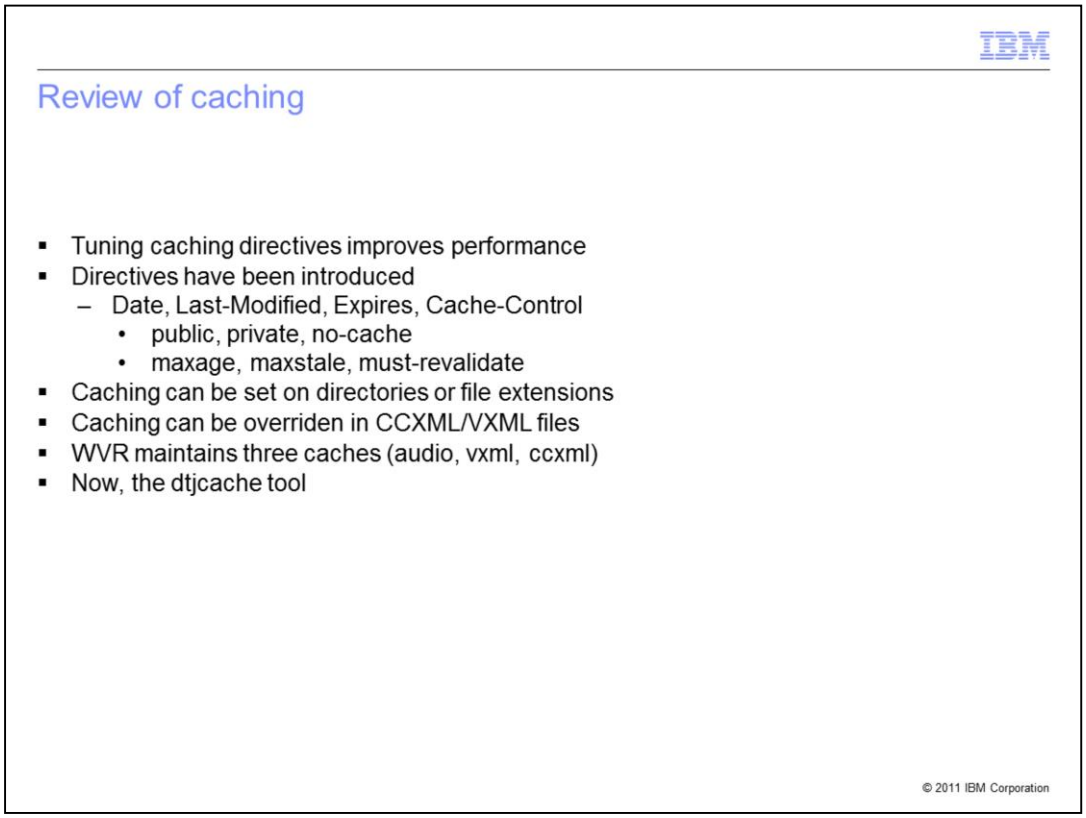
Resources in the CCXML cache expire after a 24 hour period by default.

Expired CCXML resources are refetched when they are requested by a script, fetch or createccxml tag – but resources that have already been loaded into a CCXML application is not reloaded by that application since they have expired in the cache.

When a CCXML application has expired, it is the responsibility of the CCXML application to refetch its main document by using a <fetch> element, and then using a <goto> element on the fetch.done transition.

You can use the CCXML HTTP server to trigger a transition manually, which may be a custom named transition.

Scripts that are fetched statically (rather than using a fetch element) will be loaded when the CCXML document is loaded.



This concludes the first part of this presentation, which has discussed how tuning the caching directives that apply to resources in an application can improve its performance.

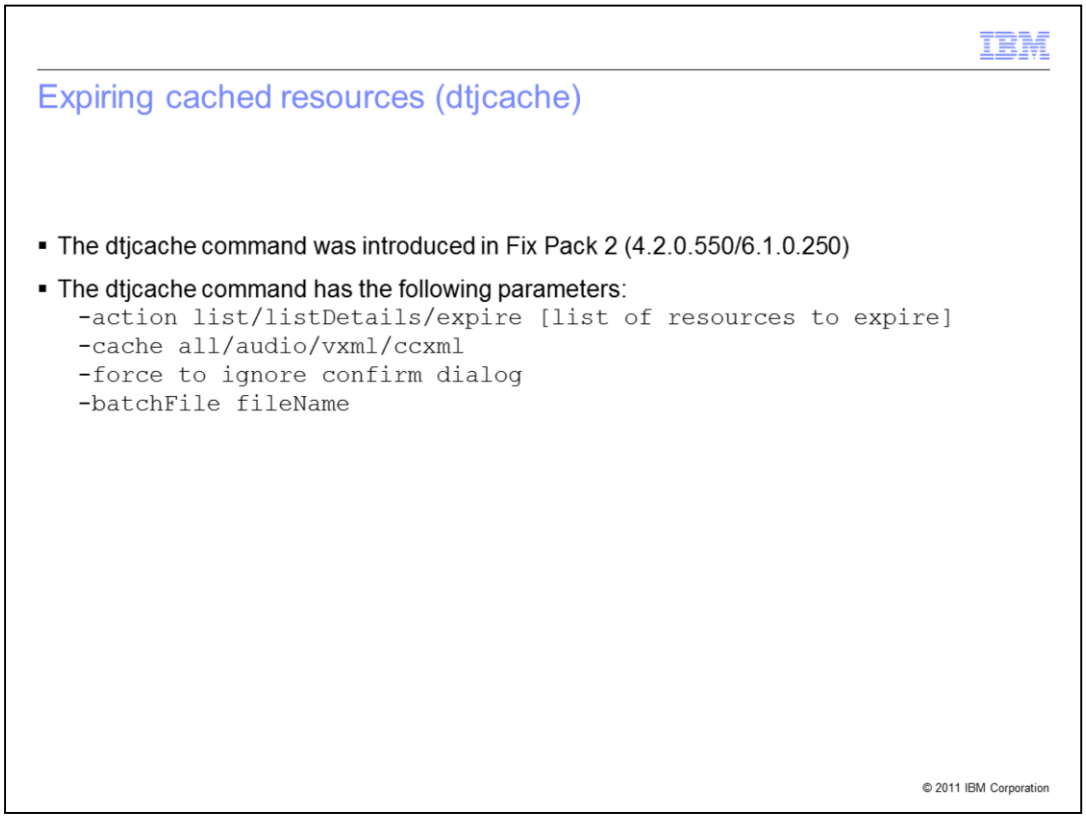
Several HTTP headers have been explained in how they pertain to caching resources. In particular, the values of the HTTP 1.1 Cache Control header have been explained.

Some examples were shown of directory and file extension based caching rules for common web servers, which will enable intelligent use of caching for applications.

The CCXML and VXML meta tag was also explained, which allows individual files to override the caching directives in their HTTP header. This tag is particularly useful in application development.

It was explained that WVR maintains three separate caches, each with slightly different rules. These are the audio, VXML and CCXML caches.

We will now go on to explain the dtjcache tool, which allows you to view and expire resources in the WVR caches.



D T J cache is a new tool that was introduced in WVR FixPack2 – level 550 on WVR 4 point 2 and level 250 on WVR 6 point 1

The command should be started as the WVR user which is dtuser by default, and has four possible parameters.

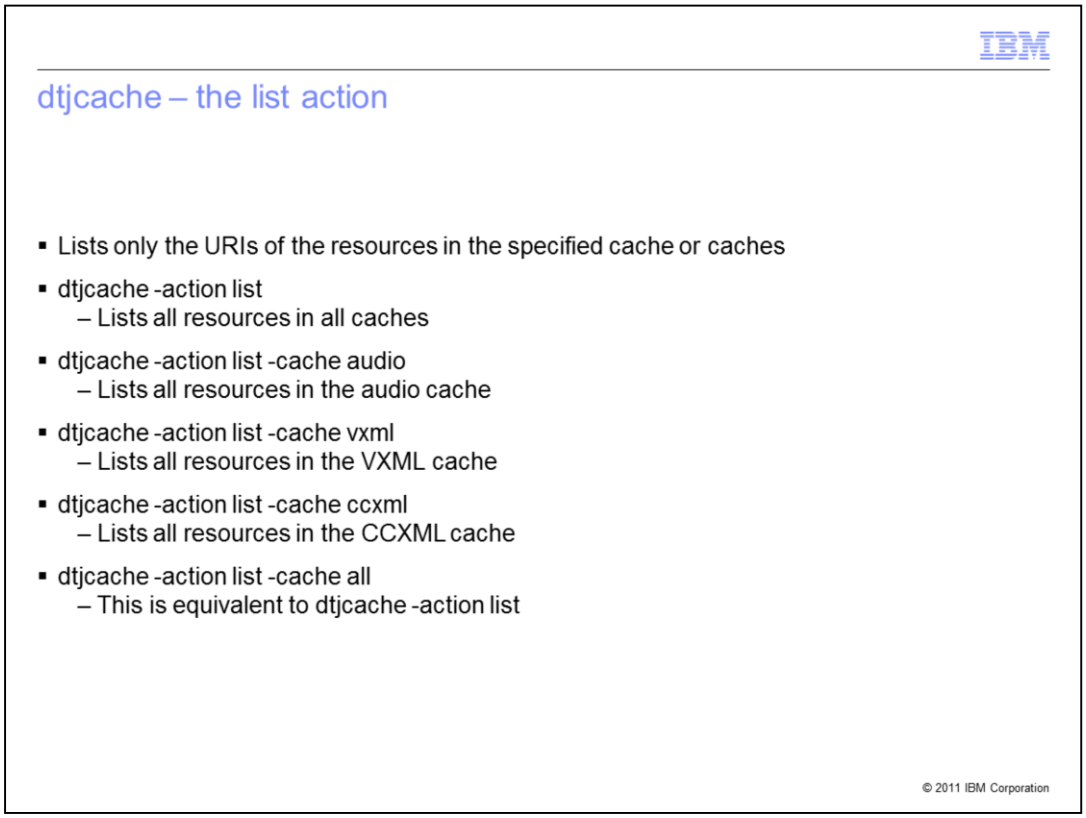
The Action parameter must be one of these: list, listDetails or expire. They list the various resources in the caches, list them with extra details, or expire resources in the caches.

The Cache parameter specifies the cache to operate on. If the parameter is not supplied then WVR assumes that the dtjcache tool should operate on all caches.

If you do not provide specific URIs to expire with the expire action, all items in the cache or caches specified are expired. A confirmation dialog will be shown in this case - you can override the presentation of this dialog by using the force parameter. This can be useful for automated scripting that uses the dtjcache tool.

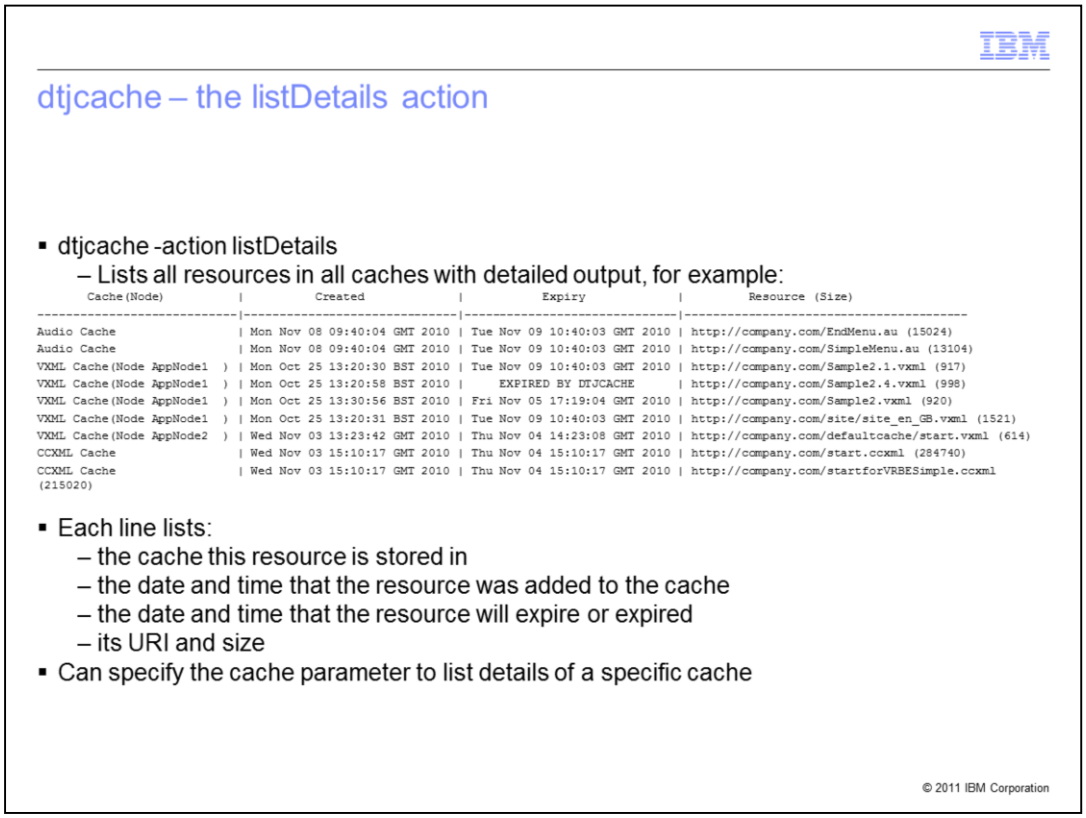
Instead of specifying URIs as a whitespace separated command line list, you can specify the batchFile parameter to use a file which should have one resource to expire per line. This will operate as if you had entered the contents of the batch file on the command line as a whitespace separated list.

Examples of dtjcache tool usage appear in the next three slides.



The list action lists the URIs of resources in the cache or caches specified, including any expired resources.

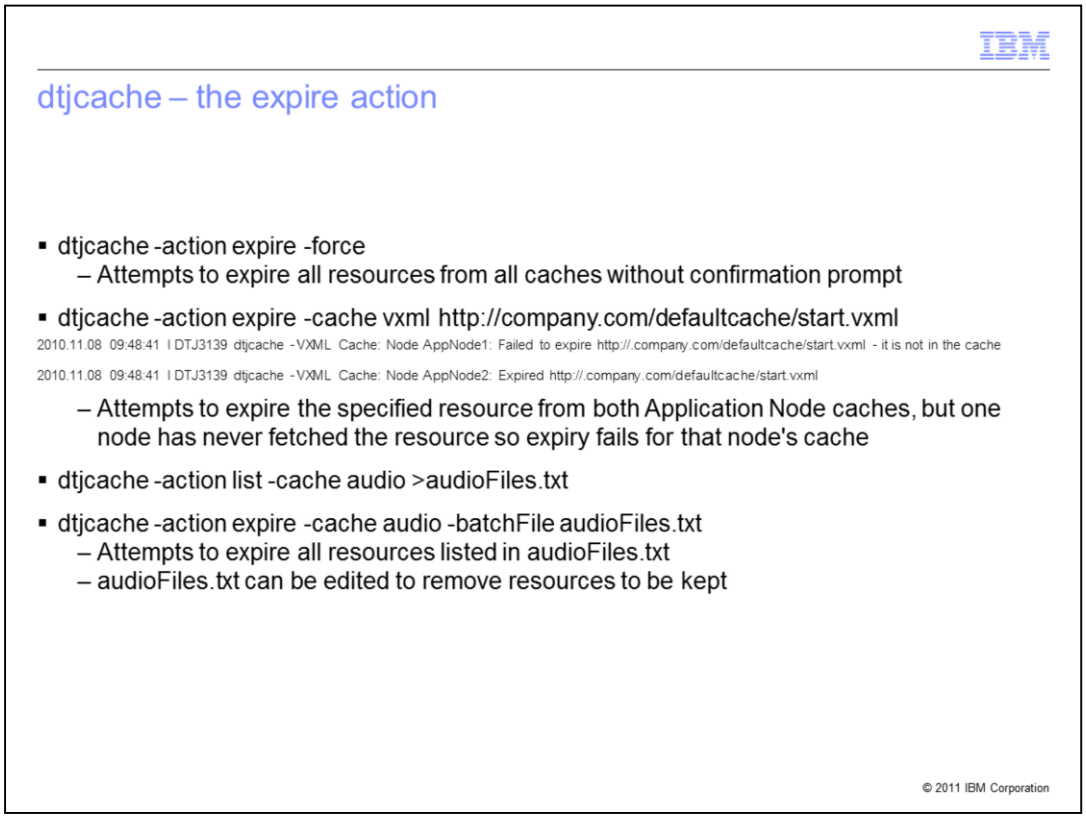
It can be used to get an idea of what resources currently exist in a cache, and can be used to generate a batch file for use with the expire action by redirecting the command line output to a file.



Here is output for the detailed listing action. As you can see, it lists the cache the item belongs in, when it was created, when it expires or expired, and the name of the resource with the size in bytes noted in brackets after it.

The expiry time is calculated from the various cache control directives and is the time after which WVR will send a fetch request as described in the cache description slides.

The cache parameter can be specified to list only resources in that cache



In its simplest form, the expire action attempts to expire everything from all caches. To refine the resources that are expired, you can specify only the cache, in which case all resources in that cache are expired, or the cache and a list of URIs on the command line or the cache and a batch file containing the URIs to expire.

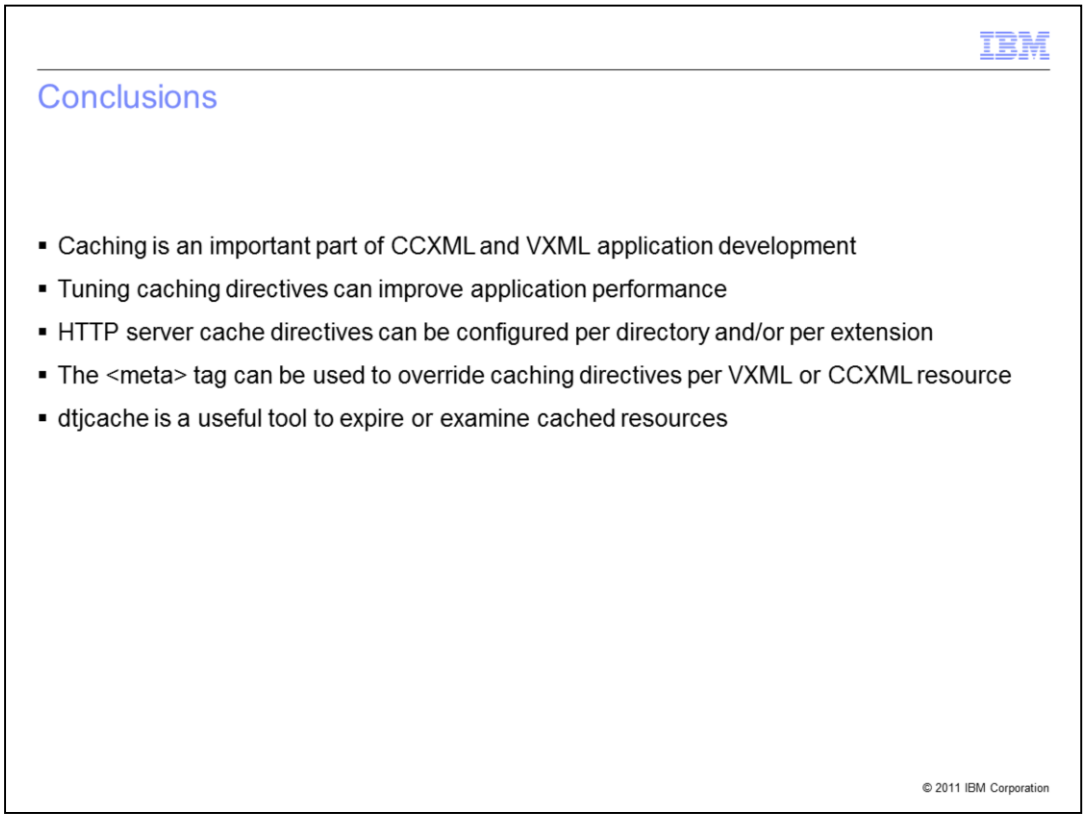
You may use the force parameter to attempt to expire everything without showing the confirmation dialog. If you are expiring using a list of resources or a batch file then no confirmation dialog is issued in any case, so the force parameter does nothing.

Each VXML application node cache is treated separately so that the tool can work where there are different cache directories for each application node.

Note that when expiring a resource, a browser will not refetch a resource that it is already using. It is when the browser next references a resource that it is refetched and the cached version is updated.

So, if your VXML application uses the subdialog tag to call another VXML file, and does so at two different points in its execution, then the second subdialog call may use a different version of the target VXML file if it expired between the first and second subdialog call.

An example is shown for using the batchFile parameter – if you want to specify all but a few resources in a cache for expiry, then generating a list of all the resources in the cache, modifying that list to remove the resources that you want to keep and then using the list as a batch file for that cache will accomplice this.



In this presentation, the importance of caching has been shown. It has also been shown how and why you can want to modify the caching directives for resources in a web application. It has been established what the performance changing properties caching can have, and some examples of how to set up common web servers have been given.

It has been shown how the meta tag works and its use has been discussed, particularly during application development, and finally the dtjcache tool introduced into WVR with Fix Pack 2 has been explained.

Trademarks, disclaimer, and copyright information