Sitecore CMS 6.6 or later

Cache Configuration Reference

Tips and Techniques for Administrators and Developers

Table of Contents

Chapter 1

Introduction

This document describes caching features of the Sitecore Web Content Management System (CMS). Developers and administrators should read this document to optimize caching, which can improve solution performance, reduce server load, and increase system capacity.

This document explains general caching concepts, including caching facilities provided by the Microsoft Internet Information Server (IIS) Web server, the ASP.NET application server, and different types of Web clients, including edge-caching devices. This document next describes how the Sitecore caches operate, and then provides instructions to monitor and tune the Sitecore caches.

This document contains the following chapters:

- **Chapter 1 — Introduction**
  This introduction to the manual.
- **Chapter 2 — Caching Overview**
  This chapter contains an overview of general caching concepts.
- **Chapter 3 — Sitecore Caching**
  This chapter contains overview of Sitecore caching facilities, and then describes how to configure the Sitecore caching features.
- **Chapter 4 — Monitor and Tune Sitecore Caches**
  This chapter describes how to monitor Sitecore cache utilization and tune cache sizes.

Chapter 2

Caching Overview

This chapter provides an overview of general caching concepts. This chapter then describes the caching features provided by the Microsoft Internet Information Server Web Server, the ASP.NET application server, and Web clients including browsers and edge-caching devices.

This chapter contains the following sections:

- Caching
- Platform Caches

2.1 Caching Concepts

Caching increases performance by storing copies of data accessed frequently from external systems in a high-performance subsystem. For example, solutions often cache data from relational databases in local Random Access Memory (RAM). Caching data in memory decreases load on the processor by increasing memory load.

Note
Not all caches store data in RAM. For example, file systems can cache large volumes of pre-processed data for long periods.

Most caches function as collections with keyed access. Each entry in the collection has a unique key, which is typically a sequence of characters. You can add an entry to the collection, remove an entry from the collection, or access the value associated with an entry in the collection using the entry's key.

Web solutions involve several types of data that you can cache at different layers of the application stack:

- Web servers typically store data that they expect to process again in RAM. Web clients and edge caching devices can cache any data in RAM, but can also use a file system to cache data.
- Web clients including edge-caching devices typically cache the content of static files, including CSS, JavaScript libraries, and images managed by developers.
- Web servers and edge caching devices typically cache dynamic markup managed by developers.
- Web servers and edge-caching devices typically cache dynamic textual content managed by CMS users.
- Web servers and edge-caching devices typically cache static and dynamic media managed by CMS users.
- Some applications involve data that you cannot cache.

Microsoft Windows limits the amount of memory available to each ASP.NET application pool. For information about techniques that you can use to control the amount of memory allocated to an application pool, see the documentation for your version of Microsoft Internet Information Server.

Important
Compared with 64-bit Microsoft Windows platforms, 32-bit Windows platforms present significant memory limitations. Use 64-bit Windows whenever possible.

2.1.1 Cache Eviction and Cache Clearing

Cache eviction is the removal of an entry from a cache.

Cache clearing refers to the removal of all entries from a cache.

2.1.2 Cache Size Limits

Cache size limits can prevent cache utilization from exceeding available memory resources. When the estimated memory consumed by the entries in a cache exceed the cache size limit, the system must evict one or more entries from the cache, or clear the entire cache, before it can add additional entries to the cache.

2.1.3 Cache Dependency

Cache dependency refers to caching layers that depend on data from other caching layers. Entries in the first cache depend on entries in the second cache. When the underlying data changes, the system must evict entries from both caches, or clear both caches entirely.

Sitecore automatically resolves cache dependencies by evicting the required entries from both caches or clearing both caches.

2.2 Platform Caches

You can cache resources in different layers in the Web application stack. For example, you can cache static resources on Web clients, such as Web browser and edge-caching devices, and you can cache relatively dynamic resources on the Web server.

2.2.1 ASP.NET Caches

Native ASP.NET caching facilities provide in-memory caching on the Web server. Sitecore does not use native ASP.NET caching facilities. This document does not describe native ASP.NET caching facilities.

Important
When you use Sitecore to publish content, Sitecore automatically evicts entries from the Sitecore caches, but does not evict entries from the native ASP.NET caches. If you use the native ASP.NET caches, then you must implement eviction and clearing strategies.

2.2.2 IIS Caches

You can configure IIS to cache resources including static files. This document does not describe IIS caching facilities.

For more information about IIS caching of static files, see http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/a0483502-c6da-486a-917a-586c463b7ed6.mspx.

2.2.3 Client Caches

Web clients, such as Web browsers and edge-caching devices, can cache resources requested from Web servers.

When a Web client requests a resource from a Web server, it can keep a copy of the response in a local cache. When a Web client processes a request for a resource that it has already requested from a Web server, the Web client can retrieve a local cached copy rather than requesting the resource from the Web server a second time. Web clients use HTTP headers and other criteria to determine caching criteria for each resource.

The default Sitecore configuration allows client caching of media, but disables client caching of pages. Sitecore has no effect on client caching of static resources such as files. To learn how to configure client caching of static files, see the section Content Expiration. To learn how to enable client caching of content, see the section *The DisableBrowserCaching Setting*.

ETags and Client Caches

An ETag (entity tag) is an HTTP header used to determine if the content associated with a URL has changed since a Web client last requested that URL. If an HTTP response includes an ETag HTTP header, and the Web client caches that response, then the Web client includes that ETag in HTTP requests for the same URL. If the Web server responds with HTTP 200 Success, the Web client renders the payload of that response. If the Web server responds with HTTP 304 Not Modified, the Web client renders the cached resource.

For more information about ETags, see http://en.wikipedia.org/wiki/HTTP_ETag.

Note
The Sitecore media library uses ETags.

Content Expiration

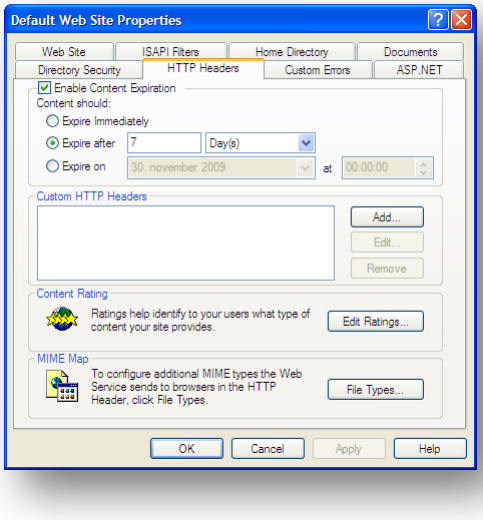
Content expiration instructs Web clients such as browsers to cache resources locally for some time.

You can enable content expiration for resources that change infrequently. If you add files to a directory, but do not change those files, you can configure content expiration for that directory. For example, you can configure content expiration for the `/sitecore` directory.

Note
With content expiration enabled, IIS defines a Cache-Control HTTP header.

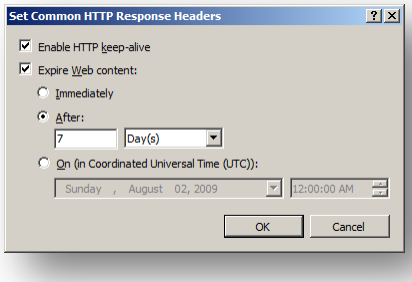
To enable content expiration on IIS6 or earlier:

1. In the **IIS Management Console**, right-click the directory or the Web site, and then click **Properties**. The **Default Web Site Properties** dialog appears.
2. In the **Default Web Site Properties** dialog box, click the **HTTP Headers** tab, and then select the **Enable Content Expiration** check box.
3. In the **Enable Content Expiration** section, click **Expire after**, and enter the number of days that Web clients should cache the files.



To enable content expiration on IIS7 or later:

1. In the **IIS Management Console**, select the directory or the Web site, and then double-click **HTTP Response Headers**.
2. Click **Set Common Headers...**. The **Set Common HTTP Response Headers** dialog box appears.
3. In the **Set Common HTTP Response Headers** dialog box, select the **Expire Web content** check box, click **After**, and enter the number of days that Web clients should cache the files.



Edge-Caching Devices

Firewalls, load-balancers, and edge-caching devices can provide caching facilities. Edge-caching devices can use the same HTTP caching headers used by other Web clients such as browsers, but can also use logic to determine what resources to cache. Because edge-caching devices support multiple Web clients, caching at the edge reduces demand on the Web server more significantly than caching on individual clients. Edge-caching devices typically respect ETags and content expiration controls in HTTP headers.

Chapter 3

Sitecore Caching

This chapter begins with an overview of Sitecore caching facilities, and then provides specific information that you can use to configure Sitecore caching features.

This chapter contains the following sections:

- Sitecore Caching Overview
- The DisableBrowserCaching Setting
- Sitecore Cache Size Limits
- Sitecore Caches
- Caching in the Executive Insight Dashboard

3.1 Sitecore Caching Overview

Sitecore caches consist of named ASP.NET collections containing various types of data in Random Access Memory (RAM) on the Web server.

Note
Sitecore caches media using one or more file systems rather than RAM. For more information about media caches, see the section *Managed Web Site Media Caches*.

Each cache serves as a key/value collection, associating each unique key with a cached value. Most caches use a sequence of characters as the key. Because each cache is a separate collection, each cache can contain different entries with a common cache key.

The name of each cache identifies its purpose. Where multiple caches serve the same purpose for different contexts, the cache name also identifies the context.

Sitecore provides a number of caching layers, where numerous caches may exist at a given layer. For example, Sitecore maintains one or more caches containing data retrieved from each content database. The layout engine uses this cached data to generate markup, with separate caches for the markup associated with each managed Web site.

In most Sitecore solutions, many pages include the same markup, such as headers and footers. Especially for large solutions, repeatedly caching the same block of markup can consume inordinate memory. Additionally, most Sitecore solutions include at least some dynamic content in each page. For these reasons, Sitecore by default disables client caching of entire pages, instead caching the individual components used by one or more pages on the Web server, while directing Web clients to cache embedded static resources such as CSS files and JavaScript libraries.

3.2 The DisableBrowserCaching Setting

The `DisableBrowserCaching` setting in the `web.config` file controls whether Sitecore instructs clients to cache entire pages.

When `DisableBrowserCaching` is `true`, Sitecore sets the following HTTP headers for each requested item (excluding media items):

```
Cache-Control: no-cache, no-store
Pragma: no-cache
```

These HTTP headers instruct Web clients, including edge-caching devices, not to cache the page. Web clients can cache CSS, media, and other resources used by the page.

Note
If `DisableBrowserCaching` is `true`, Sitecore does not apply ASP.NET `OutputCache` directives in layouts.

To instruct Web clients to cache entire pages, set `DisableBrowserCaching` to `false`, and use the ASP.NET `OutputCache` directive in layouts and sublayouts, or manipulate HTTP headers using the appropriate ASP.NET Application Programming Interfaces (APIs).

For more information about the ASP.NET `OutputCache` directive, see the Microsoft Knowledge Base article 308375 at http://support.microsoft.com/kb/308375/EN-US.

3.3 Sitecore Cache Size Limits

You can configure a size limit for each Sitecore memory cache. When a cache reaches its size limit, Sitecore evicts one or more entries from the cache at random before adding a new entry.

Note
Size limits do not apply to media caches.

Because objects can grow and shrink over time, Sitecore cannot determine the exact amount of memory consumed by each entry in each cache. Instead, Sitecore estimates the memory consumed by each cache. Sitecore uses different algorithms to determine the average size of entries in different caches.

Important
When the memory consumed by all application pools exceeds available memory on the system, Windows swaps memory to disk and can raise out-of-memory and other errors. Paging can adversely affect performance. Tune cache size limits to prevent paging and out-of-memory conditions.

3.3.1 The Load Factor

The load factor represents memory load on the system relative to other possible loads.

The `Caching.LoadFactor` setting in the `web.config` file specifies the default load factor.

The `Caching.LoadFactor` setting affects the memory size estimate for each entry in each cache. Increasing the load factor increases the estimate of the size of each entry in each cache, effectively reducing the number of entries that will fit in each cache.

To reduce the amount of memory allocated to all caches by a constant proportion, increase the load factor.

3.4 Sitecore Caches

This section describes the different types of Sitecore caches.

3.4.1 Sitecore Database Caches

For each database, several database caches store different types of information.

You can control the size of certain database caches by configuring the values of the `<data>`, `<items>`, `<paths>`, `<itempaths>`, and `<standardValues>` elements within the appropriate `/configuration/sitecore/databases/database/cacheSizes` element in the `web.config` file. For example, to configure database cache sizes for the Core database, edit the following values:

```
<databases>
  <database id="core" singleInstance="true"
    type="Sitecore.Data.Database, Sitecore.Kernel">

    <cacheSizes hint="setting">
      <data>20MB</data>
      <items>10MB</items>
      <paths>500KB</paths>
      <itempaths>10MB</itempaths>
      <standardValues>500KB</standardValues>
    </cacheSizes>
```

When a database item cache does not contain an entry for an item, Sitecore retrieves the corresponding entry from the database data cache, converts it to a different type, and stores that converted data as an entry in the database item cache. When the database data cache does not contain an entry for an item, Sitecore retrieves the corresponding entry from the database prefetch cache, converts it to a different type, and stores that converted data as an entry in the database data cache. When an entry does not exist for an item in a database prefetch cache, Sitecore retrieves that item from the data provider for that database, converts it to a different type, and stores that converted data as an entry in the database prefetch cache.

For information about configuring the size of database prefetch caches, see the section *Sitecore Database Prefetch Caches*.

Sitecore Database Prefetch Caches

Database prefetch caches contain items that Sitecore accesses during and immediately after initialization, and items with children that Sitecore often accesses as a group.

Sitecore populates database prefetch caches at application initialization, and maintains those caches over the life of the application. Each database prefetch cache stores data from a different database or data source. Not all data providers implement prefetch caches.

Each database prefetch cache entry represents an item in a database. Database prefetch cache entries include all field values for all versions of that item, and information about the parent and children of the item.

Populating the prefetch cache results in smoother user experiences immediately after application restarts. Excessive use of prefetch caches can affect the time required for application initialization.

Each of the `.config` files in the `/App_Config/Prefetch` directory with a name that matches a database name controls the prefetch cache for that database. For example, the file `/App_Config/Prefetch/Master.config` controls the prefetch cache for the Master database. Sitecore merges the information in this file with that in `/App_Config/Prefetch/Common.config`, which specifies data to store in all database prefetch caches.

The following sample demonstrates the XML structure for configuring prefetch caches.

```
<configuration>
  <cacheSize>20MB</cacheSize>
```

```
        <childLimit>100</childLimit>
        <template desc="reference">{EF295CD8-19D4-4E02-9438-94C926EF5284}</template>
        <item desc="home">{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}</item>
        <children desc="field types">{76E6D8C7-1F93-4712-872B-DA3C96B808F2}</children>
      </configuration>
```

The `<cacheSize>` element controls the maximum size of the prefetch cache.

The `<childLimit>` element defines a limit to the number of children to represent in the prefetch cache. If an item has more than this number of children, Sitecore does not cache information about the children of that item in the prefetch cache.

The `<configuration>` element can contain any number of `<item>`, `<template>`, and `<children>` elements. For `<item>` elements, Sitecore loads the specified item into the prefetch cache. For `<template>` elements, Sitecore loads all of the items based on the specified data template into the prefetch cache. For `<children>` elements, Sitecore loads all of the children of the specified item into the prefetch cache.

Sitecore Database Data Caches

Database data caches store data from all of the data providers defined for the database. Database data caches are dependent on database prefetch caches, which operate at a lower level. Like database prefetch caches, each entry in a database data cache represents a single item in a database, including parent/child relationships and field values for all versions in all languages of that item. Sitecore does not pre-populate database data caches.

Note
Database data caches are especially important for custom data providers that do not implement database prefetch caches.

The `Caching.DefaultDataCacheSize` setting in the `web.config` file specifies the default size for database data caches.

To specify the size of the data cache for an individual database, set the value of the appropriate `/configuration/sitecore/databases/database/cacheSizes/data` element in the `web.config` file for the `<database>` element with a value for the `id` attribute that matches the database name.

Sitecore Database Item Caches

Database item caches store items. Database item caches are dependent on database data caches, which operate at a lower level. Each entry in a database item cache represents a single version of an item in a single language. Sitecore does not pre-populate database item caches.

Database item caches contain objects of type `Sitecore.Data.Items.Item`. Except when implementing data providers, Sitecore developers access this API and hence this caching level.

The `Caching.DefaultItemCacheSize` setting in the `web.config` file specifies the default size for database item caches.

To specify the size of the item cache for an individual database, set the value of the `/configuration/sitecore/databases/database/cacheSizes/item` element in the `web.config` file for the `<database>` element with a value for the `id` attribute that matches the database name.

The `Caching.ItemCachingEnabled` setting in the `web.config` file enables or disables all database item caches.

The `Caching.AverageItemSize` setting in the `web.config` is an estimate of the average size of each entry in the item cache. Sitecore estimates the memory consumed by each item cache by multiplying the number of entries in the cache by `Caching.AverageItemSize`, and then multiplying that product by the load factor. The limit to the number of entries in the cache is the configured cache size divided by the product of `Caching.AverageItemSize` and the load factor.

Note
Sitecore uses different algorithms to estimate other cache sizes.

Sitecore Database Path Caches

Database path caches map URL paths to items. Sitecore does not pre-populate database path caches.

In the `web.config` file, the `Caching.DefaultPathCacheSize` setting specifies the default size of the database path caches.

To specify the size of the path cache for an individual database, set the value of the `/configuration/sitecore/databases/database/cacheSizes/path` element in the `web.config` file for the `<database>` element with a value for the `id` attribute that matches the database name.

Sitecore Database Item Path Caches

Database item paths caches supplement the existing path caches. Item paths caches map item IDs and ItemPathTypes — Name, Key, DisplayName, or ItemID — to item paths, instead of generating these item paths over and over again. Sitecore does not pre-populate the managed web site caches.

In the `web.config` file, the `Caching.DefaultItemPathsCacheSize` setting specifies the default size of the item paths caches.

To specify the size of the item path cache for an individual database, set the value of the `/configuration/sitecore/databases/database/cacheSizes/item path` element in the `web.config` file for the `<database>` element with a value for the `id` attribute that matches the database name.

Here are some examples of how item paths are stored in the item path caches:

When ItemPathType is set to Key:

- `"/sitecore/content/home"`

When ItemPathType is set to ItemID:

- `"{11111111-1111-1111-1111-111111111111}/{0DE95AE4-41AB-4D01-9EB0-67441B7C2450}"`

Sitecore Database Standard Values Caches

Database standard values caches contain standard values for data templates in the database. Sitecore does not pre-populate database standard values caches. Database standard values caches do not depend on any other caches.

The `Caching.StandardValues.DefaultCacheSize` setting in the `web.config` file specifies the default size for database standard values caches.

To specify the size of the standard values cache for an individual database, set the value of the `/configuration/sitecore/databases/database/cacheSizes/standardValues` element in the `web.config` file for the `<database>` element with a value for the `id` attribute that matches the database name.

Sitecore uses the `Caching.StandardValues.AverageValueSize` setting in the `web.config` file to estimate the amount of memory consumed by the database standard values cache.

3.4.2 Managed Web Site Caches

Sitecore manages several caches for each of the managed Web sites. Sitecore does not pre-populate the managed Web site caches.

The first defined value from the following locations determines the size of each managed Web site cache:

- The `/configuration/sitecore/sites/site` element in the `web.config` file that corresponds to the context site.
- The `/configuration/sitecore/cacheSizes/sites/*` element in the `web.config` file that corresponds to the context site.
- A setting in the `web.config` file.

For example, the default `/configuration/sitecore/sites/site` element called `website` in the `web.config` file defines attributes that control managed Web site caches for that site.

```
<site name="website" ...
  cacheHtml="true" htmlCacheSize="10MB" registryCacheSize="0"
  viewStateCacheSize="0" xslCacheSize="5MB" filteredItemsCacheSize="2MB" ... />
```

If a managed Web site definition does not specify cache size attributes, the values within the `/configuration/sitecore/cacheSizes/sites/*` element named after the managed Web site in the `web.config` file apply. For example, to specify cache sizes for the default managed Web site named `website`:

```
<cacheSizes>
  <sites>
    <website>
      <html>10MB</html>
      <registry>0</registry>
      <viewState>0</viewState>
      <filteredItems>2MB</filteredItems>
      <xsl>5MB</xsl>
    </website>
...
```

Site HTML (Output) Caches

The HTML cache (also known as the output cache) associated with each managed Web site contains the output generated by individual renderings under different conditions. Each site HTML (output) cache is dependent on the database item cache.

You can configure caching options for each rendering to populate site HTML (output) caches. Each rendering can generate different output under different conditions, which can result in multiple entries in a site HTML (output) cache for a single rendering.

For more information about caching the output of renderings, see the manual *Presentation Component Reference*.

Important
While a solution should perform well without caching, performance often depends on optimized caching. In a high-volume solution, appropriate output caching can significantly increase capacity by reducing resource requirements. Before you monitor cache utilization, configure cache settings for each rendering. When you monitor cache utilization, pay special attention to HTML (output) caches.

The `htmlCacheSize` attribute of each `/configuration/sitecore/sites/site` element in the `web.config` file specifies the size of the HTML (output) cache for that managed Web site.

The `Caching.DefaultHtmlCacheSize` setting in the `web.config` file specifies the default size for site HTML (output) caches.

The `cacheHtml` attribute of each `/configuration/sitecore/sites/site` element in the `web.config` file enables or disables HTML (output) caching for that managed Web site.

Note
When you enable rendering information in the **Sitecore Debugger**, Sitecore does not cache rendered HTML (output). To disable rendering information in the **Sitecore Debugger**, in the **Rendering** group, clear the **Information** check box.

For more information about the Sitecore Debugger, see the *Presentation Component Troubleshooting Guide*.

Site XSL Caches

The XSL cache associated with each managed Web site contains XSL transformation objects corresponding to XSL renderings.

The `xslCacheSize` attribute of each `/configuration/sitecore/sites/site` element in the `web.config` file specifies the size of the XSL cache for that managed Web site.

The `Caching.DefaultXslCacheSize` setting in the `web.config` file specifies the default size for site XSL caches.

Site Filtered Items Caches

The filtered items cache associated with each managed Web site contains information about versions of items relevant to different users.

The `filteredItemsCacheSize` attribute of each `/configuration/sitecore/sites/site` element in the `web.config` file specifies the size of the filtered items cache for that managed Web site.

The `Caching.DefaultFilteredItemsCacheSize` setting in the `web.config` file specifies the default size for site filtered items caches.

Site Registry Caches

The registry cache associated with each managed Web site contains data used primarily by the Sitecore user interfaces.

The `registryCacheSize` attribute of each `/configuration/sitecore/sites/site` element in the `web.config` file specifies the size of the registry cache for that managed Web site.

The `Caching.DefaultRegistryCacheSize` setting in the `web.config` file specifies the default size for site registry caches.

Site ViewState Cache(s)

The site ViewState caches store information about the state of ASP.NET controls.

Note
Not all managed Web sites use the ViewState cache. The default size of the ViewState cache for some managed Web sites is zero bytes.

The `viewStateCacheSize` attribute of each `/configuration/sitecore/sites/site` element in the `web.config` file specifies the size of the ViewState cache for that managed Web site.

The `Caching.DefaultViewStateCacheSize` setting in the `web.config` file specifies the default size for site ViewState caches.

The `Caching.CacheViewState` setting in the `web.config` file enables or disables ViewState caching.

Managed Web Site Media Caches

The managed Web site media caches store media. Sitecore manages separate caches for the media associated with each managed Web site.

For more information about media items, see the Media Facilities article on the SDN.

Most Sitecore caches use Random Access Memory (RAM). RAM is limited and the system can swap or reclaim it at any time. Site media caches use file systems, which have greater capacity and persist between system restarts and other events. Events that clear other Sitecore caches do not clear the site media caches.

A scheduled agent configured in the `web.config` file evicts entries from the site media caches.

Important
If you change the directory configuration for any site media cache, be sure to update the `/configuration/sitecore/scheduling/agent` element with `type Sitecore.Tasks.CleanupAgent` in the `web.config` file.

Sitecore caches the binary data associated with media items after performing any required transformations, such as resizing images. The site media caches can contain multiple entries for a single media item, such as entries for an image both at full scale and at a thumbnail size. Sitecore transmits data from entries in the site media caches without performing any additional transformation or modification of the data. For each cached media item, Sitecore maintains an additional file in the media cache that contains related metadata, such as the MIME type of the file.

There is no limit to the size of a site media cache. Sitecore does not pre-populate site media caches.

Sitecore manages site media caches in a subdirectory named after each managed Web site. The `Media.CacheFolder` setting in the `web.config` file specifies the parent directory for media caches. For example, by default, Sitecore manages the media cache for the managed Web site named `website` in the `/App_Data/MediaCache/website` directory beneath the document root of the Web site.

The `mediaCachePath` attribute of each `/configuration/sitecore/sites/site` element in the `web.config` file specifies the location of the media cache for the managed Web site. If this attribute is absent, then the `Media.CacheFolder` setting applies as described previously.

The `cacheMedia` attribute of each `/configuration/sitecore/sites/site` element in the `web.config` file enables or disables media caching for each managed Web site.

3.4.3 Client Data Store Cache

The client data store cache stores information about each authenticated user, such as the username or other user properties.

The `Caching.DefaultClientDataCacheSize` setting in the `web.config` file specifies the size of the client data store cache.

The `disableClientData` attribute of each `/configuration/sitecore/sites/site` element in the `web.config` file enables or disables client data caching for that managed Web site.

3.4.4 Security Caches

Sitecore maintains a number of caches containing security information.

The IsUserInRole Security Cache

Sitecore caches information about the roles associated with each user in the IsUserInRole cache.

The `Caching.IsUserInRoleCacheSize` setting in the `web.config` file specifies the size of the IsUserInRole cache.

The UserProfile Security Cache

Sitecore caches information about user properties such as email address in the UserProfile cache.

The `Caching.UserProfileCacheSize` setting in the `web.config` file specifies the size of the UserProfile cache.

Note
Sitecore evicts entries for a user from the UserProfile cache when a user logs in or logs out.

The AccessResult Security Cache

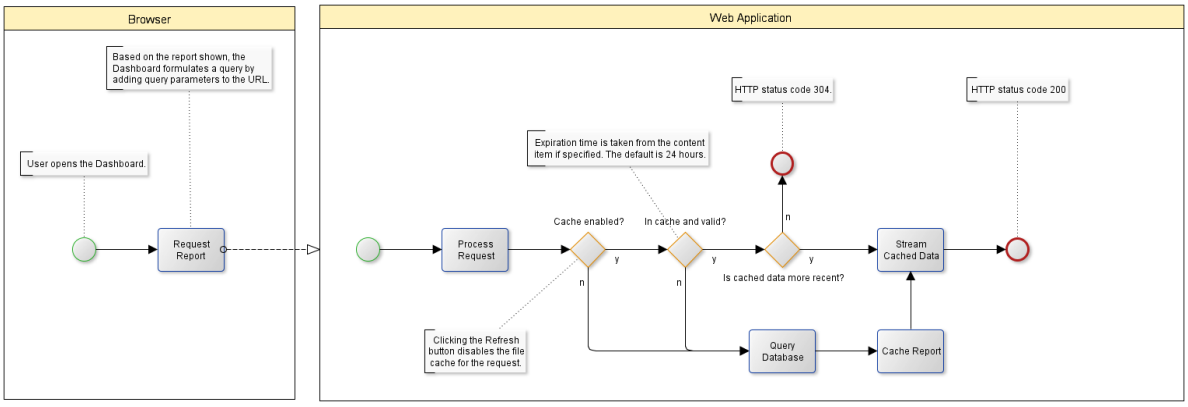
The AccessResult cache contains information about whether access rights allow users to access items.

The `Caching.AccessResultCacheSize` setting in the `web.config` file specifies the size of the AccessResult cache.

3.5 Caching in the Executive Insight Dashboard

This section describes how data is cached for the charts in the Executive Insight Dashboard. The information in this section applies to Sitecore 6.5.1 or later.

The following diagram illustrates how a data request is processed for the Executive Insight Dashboard:



3.5.1 Cache Tables and Views

The charts in the Executive Insight Dashboard are based on data that is fetched from four views:

- `TrafficByDay`
- `ValueBySource`
- `VisitEvents`
- `VisitorsByLocation`

The dashboard also uses some other views that are all based on these four views. These views aggregate large amounts of data from physical tables such as the `Visitors`, `Visits`, `Pages`, and `PageEvents` tables.

Computing the aggregated values that are displayed in the dashboards is a time consuming process. To improve the processing time, we have created four cache tables. The cache tables have the same name as the views prefixed with the word Cache followed by an underscore, for example, `Cache_PageEvents`. A daily job executes some stored procedures that populate the cache tables. Because the cache tables do not contain the newest up-to-date data, the four views contain the newest data from the base tables — the data that has been generated on the website since the cache tables were last updated.

Performance

If the cache tables are not updated frequently, the performance of the reporting views degrades as the amount of *live* data that needs to be computed increases.

3.5.2 File Cache

When the Executive Insight Dashboard client requests data for a chart, the result of every database query is stored in a cache file. These cache files are stored in the `Data\Dashboard Reports\` directory and the folder system is created automatically. If another request is made for a query that has already been processed, the response is generated from the cache file instead of querying the

database again. The cache files expire after 24 hours by default. The default expiration time can be overwritten on the individual content items that define the queries.

The items are located in:

- `\master\sitecore\system\Settings\Analytics\Dashboard Reports`

In the Executive Insight Dashboard, when you click **Refresh** in any of the charts, the `pragma: no cache` HTTP header is added to the web request. This header instructs the server to bypass the file cache and retrieve the data directly from the reporting views. The results of these queries are inserted into the file cache.

Clearing the File Cache

The `web.config` file contain the settings that control when the file cache is cleared.

The settings are:

```
<agent type="Sitecore.Tasks.CleanupAgent" method="Run" interval="06:00:00">
```

This setting specifies that the cleanup agent should run every six hours and clear the cache.

```
<remove folder="$(dataFolder)/Dashboard reports" pattern="*.*" maxAge="2.00:00:00"
recursive="true" />
```

This setting specifies that cache files should be removed every two days.

If you install a new build, you should remove these data folders or the old data will not be removed for at least two days.

Furthermore, the structure of the data could change in the future.

3.5.3 Browser Cache

The Executive Insight Dashboard also uses the browser cache. When the server sends data to the client, it includes the `Last-Modified` HTTP header in the data. This header contains information about the date and time when the report was generated. The timestamp is the last modified date attribute of the file in the server cache.

The next time the exact same URL is requested the browser includes the `If-Modified-Since` HTTP header in the request. The server uses this value to determine whether the cached report – if it is still available — is more recent than the one in the browser cache.

- If the data in the browser cache is up-to-date, the server responds with HTTP status code 304 — there is no information to send back, and the client should stay in the same document view.
- If the data in the browser cache is not up-to-date, the server sends the data from the server cache along with HTTP status code 200 — the request was fulfilled.

Browser Considerations

Firefox and IE only update the timestamp of the file in the browser cache when the browser is restarted. The browser may therefore send an outdated timestamp value even if the data in the browser cache is up-to-date.

3.6 Sitecore Cache Eviction and Cache Clearing

Sitecore evict entries from caches and clears caches due to numerous causes.

3.6.1 Data Changes Evict Cache Entries

Sitecore automatically evicts entries from the database caches when data changes. For example, when you add, update, move, rename, or delete an item, Sitecore evicts all entries relevant to that item from the appropriate database caches, and if required, recreates those entries using the new data.

3.6.2 Publishing Clears Caches

Sitecore clears one or more caches after each publishing operation.

In environments that consist of a single Sitecore instance providing both content management and content delivery, Sitecore clears the site HTML (output) caches after each publishing operation. In environments that separate a content management instance from some number of content delivery instances, Sitecore clears the site HTML (output) caches after each publishing operation, while all other caches are updated.

For more information about scaling in Sitecore, see the *Sitecore Scaling Guide* on the Sitecore Developer Network.

Note
The default Sitecore configuration assumes a single instance providing both content management and content delivery.

3.6.3 Cache Administration Page Clears Caches

You can use the **Cache Administration** page to clear Sitecore caches manually. For more information about the Cache Administration page, see the section *The Cache Administration Page*.

3.6.4 Reaching Cache Size Limit Evicts Cache Entries

When a cache reaches its size limit, Sitecore evicts one or more entries from the cache at random before adding a new entry.

3.6.5 API Calls Evict Cache Entries and Clear Caches

Sitecore exposes Application Programming Interfaces (APIs) that allow developers to evict one or more items from a cache and to clear one or more caches.

3.6.6 ASP.NET Restart Clears Caches

An ASP.NET application server restart effectively evicts all entries from all caches. You can configure IIS to restart application pools automatically.

Note
ASP.NET restarts if you update the `web.config` file, any file in the `/bin` subdirectory, or one of various other files.

Important
If the combined size of all of the caches exceeds available memory, then you can receive out-of-memory and other errors, and ASP.NET can restart.

3.6.7 Scheduled Task Clears Caches

You can configure a Sitecore scheduled task to clear the site HTML (output) caches for all of the managed Web sites.

Note
The default Sitecore configuration disables the scheduled task that clears the HTML (output) caches for each of the managed Web sites.

Note
You can implement a custom scheduled task to clear additional caches.

To configure the scheduled task to clear the site HTML (output) caches for all managed Web sites:

1. Set the `interval` attribute of the `/configuration/sitecore/scheduling/agent` element in the `web.config` file with `type` `Sitecore.Tasks.HtmlCacheClearAgent` to an interval in `HH:mm:ss` format.
2. Set the value of the `/configuration/sitecore/scheduling/frequency` element in the `web.config` file to an interval in `HH:mm:ss` format equal to or less than the value of the `interval` attribute of the `/configuration/sitecore/scheduling/agent` element with type `Sitecore.Tasks.HtmlCacheClearAgent`.

For example, given the following configuration, Sitecore clears the HTML (output) caches for all of the Web sites caches every 24 hours:

```
        <scheduling>
...
          <frequency>12:00:00</frequency>
...
          <agent type="Sitecore.Tasks.HtmlCacheClearAgent" method="Run"
            interval="24:00:00" />
```

Note
Use an alternate technique to cause Sitecore to clear caches at a specific time.

Note
Excluding prefetch caches, Sitecore does not allocate memory for caches in advance. When the number of items in the cache reaches the estimated limit based on the cache size, estimated average entry size, and load factor, Sitecore evicts a random entry from the cache.

3.6.8 Memory Monitor Clears Caches

You can configure the memory monitor to clear caches periodically if ASP.NET consumes more than an allowed amount of memory. For more information about the memory monitor, see the section *The Memory Monitor*.

3.6.9 Timeout Evicts Cache Entries

The `Caching.HtmlLifetime` setting in the web.config file specifies the duration of validity for each entry in each site output cache. By default, each entry in the output cache is valid indefinitely; publishing clears caches. You can configure `Caching.HtmlLifetime` to cause Sitecore to expire each entry in each output cache after the specified amount of time.

3.7 The Memory Monitor

Sitecore includes a memory monitor that can clear all the Sitecore caches, increase the load factor, and invoke garbage collection when ASP.NET consumes more than an allowed amount of memory.

Note
The default Sitecore configuration disables the memory monitor.

If you set the memory threshold too low, Sitecore will clear caches too frequently, adversely affecting performance. You can identify this condition from frequent messages such as the following in the Sitecore log.

WARN Memory usage exceeded the MemoryMonitor threshold.

The memory monitor can increase the load factor by 0.2 each time ASP.NET consumes more than the allowed amount of memory, but does not increase the load factor beyond the value specified by the `Caching.MaxLoadFactor` setting in the `web.config` file.

To configure the memory monitor to clear the Sitecore caches, to invoke the .NET garbage collector, or to increase the load factor when the system has consumed more than an allowed amount of memory, edit the `web.config` file. Within the `/configuration/sitecore/hooks/hook` element with `type Sitecore.Diagnostics.MemoryMonitorHook`:

- Set the value of the first `<param>` element to the memory threshold.
- Set the value of the second `<param>` element to the frequency at which the monitor should run in `HH:mm:ss` format.
- Set the value of the `<AdjustLoadFactor>` element to `true` to enable adjustment of the load factor.
- Set the value of the `<ClearCaches>` element to `true` to enable cache clearing.
- Set the value of the `<GarbageCollect>` element to `true` to enable garbage collection.

3.8 The Memory Threshold Monitor

The memory threshold monitor logs requests during which ASP.NET consumes more than a configurable amount of memory.

The memory threshold monitor logs messages such as:

WARN Memory threshold exceeded for web page

Important
If the memory threshold monitor generates repeated warnings about a specific item or type of item, then investigate requests for that item or type of item using tools for diagnosing ASP.NET memory usage.

The memory threshold monitor calculates the difference between the amount of memory used by the ASP.NET worker process when request processing begins and when request processing ends. Other activity within the same worker process, including scheduled tasks and concurrent requests, can consume memory during that period. Under load, such processing can skew the results of the memory threshold monitor.

You can use the memory threshold monitor to identify issues with memory usage in development and test environments. You can disable the memory threshold monitor or increase the threshold in production environments to reduce memory threshold logging.

To change the threshold used by the memory threshold monitor, in the `web.config` file, set the value of the `/configuration/sitecore/pipelines/httpRequestEnd/MemoryThreshold` element.

To disable the memory threshold monitor, in the `web.config` file, set the value of the `/configuration/sitecore/pipelines/httpRequestEnd/MemoryThreshold` element to `0`.

3.9 Caching Application Programmer Interfaces (APIs)

This section describes Application Programmer Interfaces (APIs) relevant to caching.

3.9.1 The Sitecore.Caching.Cache Class

The `Sitecore.Caching.Cache` class provides access to each Sitecore cache. The `Sitecore.Caching.Cache` class exposes `Add()`, `Clear()`, `Count`, `Name`, `MaxSize`, and other methods and properties that provide access to the cache.

3.9.2 The Sitecore.Caching.CacheManager Class

The `Sitecore.Caching.CacheManager` class provides access to the Sitecore caches. The `Sitecore.Caching.CacheManager` class exposes `FindCacheByName()`, `GetUserProfileCache()`, and other methods that provide access to the caches.

The `Sitecore.Caching.CacheManager.GetAllCaches()` method returns a list of all of the Sitecore caches.

The `Sitecore.Caching.CacheManager.ClearAllCaches()` method clears all Sitecore caches.

Chapter 4

Monitor and Tune Sitecore Caches

This chapter provides instructions to monitor Sitecore cache utilization and to tune cache sizes.

This chapter contains the following sections:

- Monitor Cache Utilization
- Tune Cache Sizes

4.1 Monitor Cache Utilization

You can use the tools described in this section to monitor cache utilization before tuning cache sizes. For most accurate results, monitor cache utilization at peak usage periods after the solution has been under load for several days.

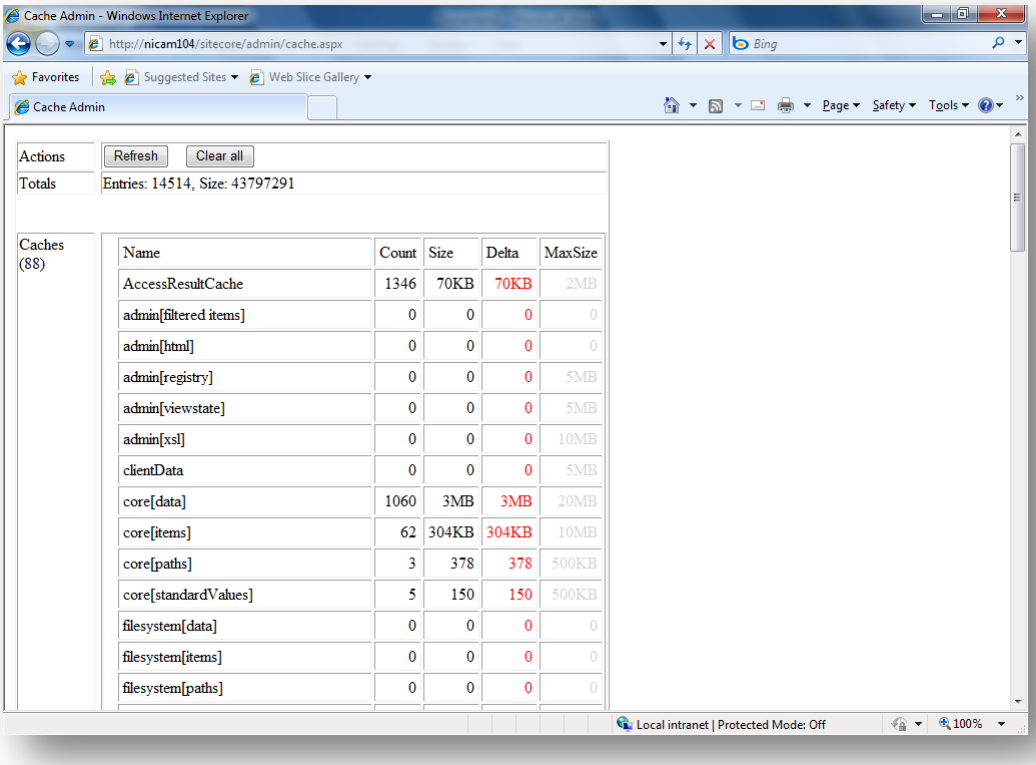
4.1.1 The Cache Administration Page

The **Cache Administration** page provides information about each of the Sitecore caches. You can use the **Cache Administration** page to monitor Sitecore cache utilization and to clear the Sitecore caches.

You can access the **Cache Administration** page at the following URL:

```
http://<hostname>/sitecore/admin/cache.aspx
```

Replace `<hostname>` with the hostname of your CMS server.



Note
If the **Cache Administration** page appears blank, in the **Actions** row at the top of the page, click **Refresh**.

Note
By default, Sitecore configures the IIS Web site to block anonymous access to pages in the `/sitecore/admin` directory. You should secure access to these pages using an alternate technique.

In the **Cache Administration** page, identifiers for caches associated with each of the managed Web sites begin with the site name, followed by the name of the cache in square brackets ("`[]`"). For example, the cache identifier `website[html]` identifies the HTML (output) cache for the managed Web site named `website`. The row labeled Totals lists the total number of items in all Sitecore

caches, and the estimated total memory consumed by those entries in bytes. The number in parentheses ("()") under the label **Caches** indicates the total number of Sitecore caches.

The **Cache Administration** page reports the following details for each cache:

- **Name** — A unique string that identifies each cache.
- **Count** — The number of items in the cache.
- **Size** — The approximate current size of the cache.
- **Delta** — The approximate change in the size of the cache since the last refresh of the cache administration page.
- **MaxSize** — The approximate limit to the size of the cache.

To update the data in the **Cache Administration** page, in the **Actions** row, click **Refresh**.

Important
Use the **Refresh** button in the **Cache Administration** page rather than the refresh feature of the browser.

To clear all of the caches listed in the Sitecore **Cache Administration** page, in the **Actions** row, click **Clear All**.

4.1.2 The Rendering Statistics Page

The **Rendering Statistics** page provides information about each of the renderings for which an entry exists in each site HTML (output) cache. You can use the rendering statistics page to identify underperforming renderings and renderings for which you can improve cache configuration.

You can access the **Rendering Statistics** page at the following URL:

```
http://<hostname>/sitecore/admin/stats.aspx
```

Replace `<hostname>` with the hostname of your CMS server.

For more information about the **Rendering Statistics** page, see the *Presentation Component Troubleshooting Guide*.

4.2 Tune Cache Sizes

This section provides instructions to set appropriate initial cache sizes and to tune cache sizes to maximize cache hits.

4.2.1 Set Initial Cache Sizes

For an environment that provides only content delivery, set the size of the prefetch and data caches for the publishing target (Web) database to at least 150MB. Set the size of the items cache for the publishing target database to at least 125MB.

For an environment that provides both content management and content delivery, set the size of the prefetch and data caches for the Master database to at least 100MB. Set the size of the item cache for the Master database and the prefetch and data caches for the publishing target database to at least 75MB. Set the size of the items cache for the publishing target database to at least 50MB.

To configure cache sizes for the worst-case scenario:

1. Configure cache sizes of at least 100MB for the most important caches, which include site HTML (output) caches and the database prefetch, data, and item caches.
2. While monitoring cache utilization, repeatedly invoke a process that accesses each field value of each version of each item in each language of each database. Increase cache sizes until Sitecore can cache all available data.
3. Based on factors including expected growth and available memory, calculate an appropriate limit for each cache based on current utilization. If memory is constrained, allocate a fraction of current utilization, such as 80%. If memory is unconstrained and you expect rapid growth, then allocate more than 100% of current utilization to each cache. When balancing available memory, give priority to data that is expensive to retrieve, including data retrieved from databases.

4.2.2 Adjust Cache Sizes

In the **Cache Administration** page, if the **Delta** for a cache fluctuates constantly, or if the size of a cache is consistently above 80% of the size limit for that cache, then increase the size of the cache by at least 50%.

Database access can use significant processing resources. You can reduce database access by tuning the database prefetch, data, and item caches. Database cache tuning is especially important in solutions with large numbers of items. When possible, provide sufficient RAM to cache the entire database.

To reduce rendering load and database access, optimize cache settings for renderings and tune the site HTML (output) caches.

Important
The amount of data to cache increases as you add items, languages, and versions. Continue to monitor and tune cache sizes over the life of the solution.