

NCache Migration from 3.6 to 3.8 SP1

August 25, 2010

Table of Contents

1	Introduction.....	3
2	API Migration Tips.....	6
2.1	ADD Operations.....	6
2.2	Get Operations.....	6
2.3	Insert Operations.....	7
3	Read Thru Provider.....	7
3.1	Migration Tips.....	8
4	CacheLoader.....	9

1 Introduction

NCache API has gone through major improvements in the recent release of NCache i.e. 'NCache 3.8 SP1' for ease of use. The purpose of this document is to assist existing customers for a smooth migration from previous version NCache 3.6 to NCache 3.8 SP1. New API of NCache ensures availability of all existing features. The document provides mapping information from old API to new API.

The major changes between the 3.6 and 3.8 API are of refactoring other than new API methods provided in 3.8. Any new API method provided in NCache 3.8 SP1 can be found in NCache installed Help. Table 1 lists all the Classes/Interfaces/Enums which are moved from one assembly to other for the ease of use and to reduce ambiguity.

Class/Interface/Enum	Old Reference	New Reference
AggregateCacheDependency	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
CacheDependency	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
CacheItemPriority	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime
CmdParamsDbType	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
CmdParamsType	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
DBCachedependency	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
DBDependencyFactory	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
IReadThruProvider	Alachisoft.NCache.Caching.DatasourceProviders	Alachisoft.NCache.Runtime.DatasourceProviders
IWriteThruProvider	Alachisoft.NCache.Caching.DatasourceProviders	Alachisoft.NCache.Runtime.DatasourceProviders
ICacheLoader	Alachisoft.NCache.Caching.CacheLoader	Alachisoft.NCache.Runtime.CacheLoader
OracleCacheDependency	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
OracleCmdParams	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
OracleCmdParamsType	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
OracleCommandType	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
OracleParameterDirection	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
SqlCacheDependency	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
SqlCmdParams	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies

SqlCmpOptions	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
SqlCommandType	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
SqlDataRowVersion	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
SqlParameterDirection	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Dependencies
Tag	Alachisoft.NCache.Web.Caching	Alachisoft.NCache.Runtime.Caching

Table 1 - Refactored Classes/Enum

2 Legend

Color	Description
	Not available in New API. Alternate usage is suggested
	Option exist in Old API and an alternate options is available in New API
	Alternate substitute option in New API.

3 API Migration Tips

3.1 ADD Operations

Old API	<code>public CacheItemVersion Add(string key, object value, Tag[] tags)</code>
New API	<code>public CacheItemVersion Add(string key, object value, Tag[] tags)</code>
Change	Alachisoft.NCache.Web.Caching.Tag is moved to Alachisoft.NCache.Runtime.Caching.Tag

Old API	<code>public CacheItemVersion Add(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority)</code>
New API	<code>public CacheItemVersion Add(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority)</code>
Change	Alachisoft.NCache.Web.Caching.CacheDependency is moved to Alachisoft.NCache.Runtime.Dependencies.CacheDependency. Similarly, Alachisoft.NCache.Web.Caching.CacheItemPriority is moved to Alachisoft.NCache.Runtime.CacheItemPriority

Old API	<code>public virtual bool AddDependency(string key, CacheDependency dependency, bool isResyncRequired)</code>
New API	<code>public virtual bool AddDependency(string key, CacheDependency dependency, bool isResyncRequired)</code>
Change	Alachisoft.NCache.Web.Caching.CacheDependency is moved to Alachisoft.NCache.Runtime.Dependencies.CacheDependency

3.2 Get Operations

Old API	<code>public virtual Hashtable GetByAllTags(Tag[] tags)</code>
New API	<code>public virtual Hashtable GetByAllTags(Tag[] tags)</code>
Change	Alachisoft.NCache.Web.Caching.Tag is moved to Alachisoft.NCache.Runtime.Caching.Tag

Old API	<code>public virtual Hashtable GetByAnyTag(Tag[] tags)</code>
New API	<code>public virtual Hashtable GetByAnyTag(Tag[] tags)</code>

Change	Alachisoft.NCache.Web.Caching.Tag is moved to Alachisoft.NCache.Runtime.Caching.Tag
--------	---

Old API	<code>public virtual Hashtable GetByTag(Tag tag)</code>
New API	<code>public virtual Hashtable GetByTag(Tag tag)</code>
Change	Alachisoft.NCache.Web.Caching.Tag is moved to Alachisoft.NCache.Runtime.Caching.Tag

3.3 Insert Operations

Old API	<code>public CacheItemVersion Insert(string key, object value, Tag[] tags)</code>
New API	<code>public CacheItemVersion Insert(string key, object value, Tag[] tags)</code>
Change	Alachisoft.NCache.Web.Caching.Tag is moved to Alachisoft.NCache.Runtime.Caching.Tag

Old API	<code>public CacheItemVersion Insert(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority)</code>
New API	<code>public CacheItemVersion Insert(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority)</code>
Change	Alachisoft.NCache.Web.Caching.CacheDependency is moved to Alachisoft.NCache.Runtime.Dependencies.CacheDependency Similarly, Alachisoft.NCache.Web.Caching.CacheItemPriority is moved to Alachisoft.NCache.Runtime.CacheItemPriority

4 Read Thru Provider

Previously in `IReadThruProvider` implementation `ExpirationHint` and `EvictionHint` can only be specified while adding value in cache. Because of that users were unable to provide the group, sub-group, CacheDependency and tags etc. Now using new flexible implementation you can provide most of the option as you can provide using Client API. `ProviderCacheItem` is introduced which is almost identical to `CacheItem`.

`ProviderCacheItem` contains the following parameters:

Type	Field	Description
<code>DateTime</code>	<code>AbsoluteExpiration</code>	Used to specify Absolute Expiration time.

<code>CacheDependency</code>	Dependency	Used if you want to specify any type of dependency.
<code>string</code>	Group	Used specify item group.
<code>string</code>	SubGroup	Used to specify item sub group.
<code>CacheItemPriority</code>	ItemPriority	Specify the priority level in case of eviction.
<code>bool</code>	ResyncItemOnExpiration	Set to true if you want toe resync item on expiration.
<code>string</code>	ResyncProviderName	Provider name to resync item when expired.
<code>TimeSpan</code>	SlidingExpiration	Specify if you want sliding expiration
<code>Tag[]</code>	Tags	Using this you can specify any number of tags with the object.
<code>object</code>	Value	Value you want to insert

Table 2 - ProviderCacheItem Description

4.1 Migration Tips

Old API	<code>public object LoadFromSource(string key, out ExpirationHint exh, out EvictionHint evh)</code>
New API	<code>public void LoadFromSource(string key, out ProviderCacheItem cacheItem)</code>
Change	<code>ProviderCacheItem</code> is introduced using which you can provide most of the option as you can specify on Client API while adding <code>CacheItem</code> .
Usage	<pre>public void LoadFromSource(string key, out ProviderCacheItem cacheItem) { cacheItem = new ProviderCacheItem("Dell Computer"); cacheItem.AbsoluteExpiration = DateTime.Now.AddSeconds(10); cacheItem.Group = "World"; cacheItem.SubGroup = "USA"; cacheItem.Tags = new Tag[] { new Tag("Notebook"), new Tag("Work Stations"), new Tag("Office") }; cacheItem.ResyncItemOnExpiration = true; }</pre>
Old API	<code>public Hashtable LoadFromSource(String[] keys, out ExpirationHint[] exh, out EvictionHint[] evh)</code>

New API	<pre>public Dictionary<string, ProviderCacheItem> LoadFromSource(String[] keys)</pre>
Change	<p><code>ProviderCacheItem</code> is introduced using which you can provide most of the option as you can specify on client API while adding <code>CacheItem</code>.</p> <p>The new implementation expects the Dictionary of <code>ProviderCacheItem</code> along with keys as this help in keeping the type check and reduces inconsistencies.</p>
Usage	<pre>public Dictionary<string, ProviderCacheItem> LoadFromSource(string[] keys) { Dictionary<string, ProviderCacheItem> providerCacheItems = new Dictionary<string, ProviderCacheItem>(); foreach (string key in keys) { ProviderCacheItem cacheItem = new ProviderCacheItem("Dell Computer"); cacheItem.AbsoluteExpiration = DateTime.Now.AddSeconds(10); cacheItem.Group = "World"; cacheItem.SubGroup = "USA"; cacheItem.Tags = new Tag[] { new Tag("Notebook"), new Tag("Work Stations"), new Tag("Office") }; cacheItem.ResyncItemOnExpiration = true; providerCacheItems.Add(key, item); } return providerCacheItems; }</pre>

5 CacheLoader

Now `ICacheLoader` implementation also expects `ProviderCacheItem` instead of `CacheItem`.