

NCache 3.6 Migration Guide

March 15, 2009

Table of Contents

1	Introduction	2
2	Legends	2
3	API Migration Instructions.....	2
3.1	ADD Operations.....	2
3.1.1	General Methods	2
3.1.2	AddBulk Methods.....	7
3.1.3	AddAsync Methods.....	8
3.2	Clear Operations.....	16
3.2.1	ClearAsync Methods.....	16
3.2.2	ClearClientCacheAsync Method	17
3.3	GET Operations	18
3.3.1	General Get Methods	18
3.3.2	GetBulk Methods	19
3.4	Insert Operations.....	20
3.4.1	General Insert Methods	20
3.4.2	InsertBulk Methods.....	26
3.4.3	InsertAsync Methods.....	29
3.5	Remove Operations.....	35
3.5.1	General Remove Methods	35
3.5.2	RemoveBulk Methods.....	36
3.5.3	RemoveAsync Methods.....	36
3.6	Unlock Operations.....	38
3.6.1	General UnLock Method.....	38

1 Introduction

NCache API has gone through major improvements in the recent release of NCache i.e. 'NCache 3.6' with the purpose of ease of use. The purpose of this document is to assist existing customers for a smooth migration from previous versions to NCache 3.6. This new API of NCache ensures availability of all existing features.

This document provides mapping information from old API to new API.

2 Legends

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

3 API Migration Instructions

3.1 ADD Operations

3.1.1 General Methods

Old API	<pre>public object Add(string key, CacheItem item, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
New API	<pre>public CacheItemVersion Add(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
Usage	Name of <code>DataSourceUpdateOpt</code> has been changed with <code>DSWriteOption</code>

Old API	<pre>public object Add(string key, CacheItem item, string group, string subGroup)</pre>
New API	<pre>public CacheItemVersion Add(string key, CacheItem item)</pre>
Usage	<p><code>CacheItem</code> has the properties 'Group' and 'SubGroup'. Instead of passing the group and sub-group as parameters, these properties of <code>CacheItem</code> can be set.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Group = "group"; cacheItem.SubGroup = "sub-group";</pre>

	<code>cache.Add("key", cacheItem);</code>
--	---

Old API	<pre>public object Add(string key, CacheItem item, string group, string subGroup, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
New API	<pre>public CacheItemVersion Add(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
Usage	<p><code>CacheItem</code> has the properties 'Group' and 'SubGroup'. Instead of passing the group and sub-group as parameters, these properties of <code>CacheItem</code> can be set.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Group = "group"; cacheItem.SubGroup = "sub-group"; cache.Add("key", cacheItem, DSWriteOption.None, null);</pre>

Old API	<pre>public object Add(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback)</pre>
New API	<pre>public CacheItemVersion Add(string key, CacheItem item)</pre>
Usage	<p><code>CacheItem</code> has the properties for the missing parameters that can be set before adding the <code>CacheItem</code> to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = null; cacheItem.SyncDependency = null; cacheItem.AbsoluteExpiration = DateTime.Now.AddMinutes(5); cacheItem.SlidingExpiration = Cache.NoSlidingExpiration; cacheItem.Priority = CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved);</pre>

	<code>cache.Add("key", cacheItem);</code>
--	---

Old API	<pre>public object Add(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback)</pre>
---------	---

New API	<pre>public CacheItemVersion Add(string key, CacheItem item)</pre>
---------	--

Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = null; cacheItem.SyncDependency = null; cacheItem.AbsoluteExpiration = DateTime.Now.AddMinutes(5); cacheItem.SlidingExpiration = Cache.NoSlidingExpiration; cacheItem.Priority = CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback (OnItemUpdated); cache.Add("key", cacheItem);</pre>
-------	--

Old API	<pre>public object Add(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, DataSourceUpdateOpt dsUpdateOpt, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback, DataSourceItemsAddedCallback onDataSourceItemAdded, bool isResyncExpiredItems, string group, string subGroup)</pre>
---------	--

New API	<pre>public CacheItemVersion Add(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded) </pre>
---------	---

Usage	<p><code>CacheItem</code> has the properties for the missing parameters that can be set before adding the <code>CacheItem</code> to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = null; cacheItem.SyncDependency = null; cacheItem.AbsoluteExpiration = DateTime.Now.AddMinutes(5); cacheItem.SlidingExpiration = Cache.NoSlidingExpiration; cacheItem.Priority = CacheItemPriority.Normal; cacheItem.IsResyncExpiredItems = false; cacheItem.Group = "group"; cacheItem.SubGroup = "sub-group"; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved); cacheItem.ItemUpdateCallback = new CacheItemRemovedCallback(OnItemUpdated); cache.Add("key", cacheItem, DSWriteOption.None, null);</pre>
-------	--

Old API	<pre>public object Add(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback)</pre>
---------	--

New API	<pre>public CacheItemVersion Add(string key, CacheItem item)</pre>
---------	--

Usage	<p><code>CacheItem</code> has the properties for the missing parameters that can be set before adding the <code>CacheItem</code> to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved); cache.Add("key", cacheItem);</pre>
-------	--

Old API	<pre>public object Add(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback)</pre>
---------	---

New API	<pre>public CacheItemVersion Add(string key, CacheItem item)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved); cacheItem.ItemUpdateCallback = new CacheItemRemovedCallback(OnItemUpdated); cache.Add("key", cacheItem);</pre>

Old API	<pre>public object Add(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback, bool isResyncExpiredItems, string group, string subGroup)</pre>
New API	<pre>public CacheItemVersion Add(string key, CacheItem item)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.IsResyncExpiredItems = false; cacheItem.Group = "group"; cacheItem.SubGroup = "sub-group"; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(OnItemUpdated); cache.Add("key", cacheItem);</pre>

3.1.2 AddBulk Methods

Old API	<pre>public IDictionary Add(string[] keys, CacheItem[] items)</pre>
New API	<pre>public virtual IDictionary AddBulk(string[] keys, CacheItem[] items, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemsAdded)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem[] items = new CacheItem[5]; string[] keys = new string[5]; for (int i = 0; i < 5; i++) { items[i] = new CacheItem(i); keys[i] = i.ToString(); } cache.AddBulk(keys, items, DSWriteOption.None, null);</pre>

Old API	<pre>public IDictionary Add(string[] keys, CacheItem[] items, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsAddedCallback onDataSourceItemsAdded)</pre>
New API	<pre>public virtual IDictionary AddBulk(string[] keys, CacheItem[] items, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemsAdded)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem[] items = new CacheItem[5]; string[] keys = new string[5]; for (int i = 0; i < 5; i++) { items[i] = new CacheItem(i); keys[i] = i.ToString(); } cache.AddBulk(keys, items, DSWriteOption.None, null);</pre>

Old API	<pre>public IDictionary Add(string[] keys, CacheItem[] items, string group, string subGroup)</pre>
New API	<pre>public virtual IDictionary AddBulk(string[] keys, CacheItem[] items, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemsAdded)</pre>

Usage	<p><code>CacheItem</code> has the properties for the missing parameters that can be set before adding the <code>CacheItem</code> to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem[] items = new CacheItem[5]; string[] keys = new string[5]; for (int i = 0; i < 5; i++) { items[i] = new CacheItem(i); items[i].Group = "group"; items[i].SubGroup = "sub-group"; keys[i] = i.ToString(); } cache.AddBulk(keys, items, DSWriteOption.None, null);</pre>
-------	--

Old API	<pre>public virtual IDictionary Add(string[] keys, CacheItem[] items, string group, string subGroup, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsAddedCallback onDataSourceItemsAdded)</pre>
New API	<pre>public virtual IDictionary AddBulk(string[] keys, CacheItem[] items, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemsAdded)</pre>
Usage	<p><code>CacheItem</code> has the properties for the missing parameters that can be set before adding the <code>CacheItem</code> to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem[] items = new CacheItem[5]; string[] keys = new string[5]; for (int i = 0; i < 5; i++) { items[i] = new CacheItem(i); items[i].Group = "group"; items[i].SubGroup = "sub-group"; keys[i] = i.ToString(); } cache.AddBulk(keys, items, DSWriteOption.None, null);</pre>

3.1.3 AddAsync Methods

Old API	<pre>public object AddAsync(string key, CacheItem item)</pre>
---------	---

New API	<pre>public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cache.AddAsync("key", cacheItem, DSWriteOption.None, null);</pre>

Old API	<pre>public object AddAsync(string key, CacheItem item, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
New API	<pre>public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cache.AddAsync("key", cacheItem, DSWriteOption.None, new DataSourceItemsAddedCallback(OnDSItemAdded));</pre>

Old API	<pre>public object AddAsync(string key, CacheItem item, string group, string subGroup)</pre>
New API	<pre>public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Group = "group"; cacheItem.SubGroup = "sub-group"; cache.AddAsync("key", cacheItem, DSWriteOption.None, new DataSourceItemsAddedCallback(OnDSItemAdded));</pre>

Old API	<pre>public object AddAsync(string key,</pre>
---------	--

	<pre>CacheItem item, string group, string subGroup, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsAddedCallback onDataSourceItemAdded) </pre>
New API	<pre>public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded) </pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Group = "group"; cacheItem.Group = "sub-group"; cache.AddAsync("key", cacheItem, DSWriteOption.None, new DataSourceItemsAddedCallback(OnDSItemAdded)); </pre>

Old API	<pre>public object AddAsync(string key, object value) </pre>
New API	<pre>public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded) </pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cache.AddAsync("key", cacheItem, DSWriteOption.None, null); </pre>

Old API	<pre>public object AddAsync(string key, object value, string group, string subGroup) </pre>
New API	<pre>public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded) </pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); </pre>

```

cacheItem.Group = "group";
cacheItem.Group = "sub-group";

cache.AddAsync("key", cacheItem, DSWriteOption.None, new
DataSourceItemsAddedCallback(OnDSItemAdded));

```

Old API	<pre> public object AddAsync(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback) </pre>
---------	---

New API	<pre> public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded) </pre>
---------	--

Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre> Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.SyncDependency = null; cacheItem.AbsoluteExpiration = Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration = TimeSpan.FromMinutes(10); cacheItem.Priority = CacheItemPriority.Normal; cacheItem.Group = "group"; cacheItem.Group = "sub-group"; cache.AddAsync("key", cacheItem, DSWriteOption.None, null); </pre>
-------	--

Old API	<pre> public object AddAsync(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback) </pre>
---------	--

New API	<pre> public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded) </pre>
---------	--

Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre> Cache cache = NCache.InitializeCache("mycache"); </pre>
-------	--

```

CacheItem cacheItem = new CacheItem("data");
cacheItem.Dependency = new CacheDependency("file-path");
cacheItem.SyncDependency = null;
cacheItem.AbsoluteExpiration = Cache.NoAbsoluteExpiration;
cacheItem.SlidingExpiration = TimeSpan.FromMinutes(10);
cacheItem.Priority = CacheItemPriority.Normal;
cacheItem.ItemRemoveCallback = new
CacheItemRemovedCallback(OnItemRemoved);
cacheItem.ItemUpdateCallback = new
CacheItemUpdatedCallback(OnItemUpdated);

cache.AddAsync("key", cacheItem, DSWriteOption.None, null);

```

Old API	<pre> public object AddAsync(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, DataSourceUpdateOpt dsUpdateOpt, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback, AsyncItemAddedCallback onAsyncItemAddCallback, DataSourceItemsAddedCallback onDataSourceItemAdded, string group, string subGroup) </pre>
---------	---

New API	<pre> public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded) </pre>
---------	--

Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre> Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.SyncDependency = null; cacheItem.AbsoluteExpiration = Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration = TimeSpan.FromMinutes(10); cacheItem.Priority = CacheItemPriority.Normal; cacheItem.Group = "group"; cacheItem.Group = "sub-group"; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(OnItemUpdated); cacheItem.AsyncItemAddCallback = new AsyncItemAddedCallback(OnAsyncItemAdded); cache.AddAsync("key", cacheItem, DSWriteOption.None, null); </pre>
-------	--

Old API	<pre>public object AddAsync(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback)</pre>
New API	<pre>public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.AbsoluteExpiration = Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration = TimeSpan.FromMinutes(10); cacheItem.Priority = CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved); cache.AddAsync("key", cacheItem, DSWriteOption.None, null);</pre>

Old API	<pre>public object AddAsync(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback)</pre>
New API	<pre>public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.AbsoluteExpiration = Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration = TimeSpan.FromMinutes(10); cacheItem.Priority = CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved); cacheItem.ItemUpdateCallback = new</pre>

```
CacheItemUpdatedCallback(OnItemUpdated);

cache.AddAsync("key", cacheItem, DSWriteOption.None, null);
```

Old API	<pre>public object AddAsync(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback, AsyncItemAddedCallback onAsyncItemAddCallback)</pre>
---------	---

New API	<pre>public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
---------	--

Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.AbsoluteExpiration = Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration = TimeSpan.FromMinutes(10); cacheItem.Priority = CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(OnItemUpdated); cacheItem.AsyncItemAddCallback = new AsyncItemAddedCallback(OnAsyncItemAdded); cache.AddAsync("key", cacheItem, DSWriteOption.None, null);</pre>
-------	---

Old API	<pre>public object AddAsync(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback, AsyncItemAddedCallback onAsyncItemAddCallback, string group, string subGroup)</pre>
---------	--

New API	<pre>public object AddAsync(string key, CacheItem item, DSWriteOption dsWriteOption,</pre>
---------	---

	<pre>DataSourceItemsAddedCallback onDataSourceItemAdded)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.AbsoluteExpiration = Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration = TimeSpan.FromMinutes(10); cacheItem.Priority = CacheItemPriority.Normal; cacheItem.Group = "group"; cacheItem.SubGroup = "sub-group"; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(OnItemRemoved); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(OnItemUpdated); cacheItem.AsyncItemAddCallback = new AsyncItemAddedCallback(OnAsyncItemAdded); cache.AddAsync("key", cacheItem, DSWriteOption.None, null);</pre>

3.2 Clear Operations

3.2.1 ClearAsync Methods

Old API	<code>public virtual void ClearAsync()</code>
New API	<code>public virtual void ClearAsync(DSWriteOption updateOpt, AsyncCacheClearedCallback onAsyncCacheClearCallback, DataSourceClearedCallback dataSourceClearedCallback)</code>
Usage	<code>Cache cache = NCache.InitializeCache("mycache"); cache.ClearAsync(DSWriteOption.None, null, null);</code>

Old API	<code>public virtual void ClearAsync(AsyncCacheClearedCallback onAsyncCacheClearCallback)</code>
New API	<code>public virtual void ClearAsync(DSWriteOption updateOpt, AsyncCacheClearedCallback onAsyncCacheClearCallback, DataSourceClearedCallback dataSourceClearedCallback)</code>
Usage	<code>Cache cache = NCache.InitializeCache("mycache"); cache.ClearAsync(DSWriteOption.None, new AsyncCacheClearedCallback(OnAsyncCacheCleared), null);</code>

Old API	<code>public virtual void ClearAsync(DataSourceUpdateOpt updateOpt, AsyncCacheClearedCallback onAsyncCacheClearCallback, DataSourceClearedCallback dataSourceClearedCallback)</code>
New API	<code>public virtual void ClearAsync(DSWriteOption updateOpt, AsyncCacheClearedCallback onAsyncCacheClearCallback, DataSourceClearedCallback dataSourceClearedCallback)</code>
Usage	<code>Cache cache = NCache.InitializeCache("mycache"); cache.ClearAsync(DSWriteOption.None, new AsyncCacheClearedCallback(OnAsyncCacheCleared), new DataSourceClearedCallback(OnDSCleared));</code>

Old API	<code>public virtual void ClearAsync(DataSourceUpdateOpt updateOpt, DataSourceClearedCallback dataSourceClearedCallback)</code>
New API	<code>public virtual void ClearAsync(DSWriteOption updateOpt, AsyncCacheClearedCallback onAsyncCacheClearCallback, DataSourceClearedCallback dataSourceClearedCallback)</code>
Usage	<code>Cache cache = NCache.InitializeCache("mycache");</code>

	<code>cache.ClearAsync(DSWriteOption.None, null, new DataSourceClearedCallback(OnDSCleared));</code>
--	--

3.2.2 ClearClientCacheAsync Method

Old API	<code>public virtual void ClearClientCacheAsync()</code>
New API	<code>public virtual void ClearClientCacheAsync(AsyncCacheClearedCallback onAsyncCacheClearCallback)</code>
Usage	<code>Cache cache = NCache.InitializeCache("mycache"); cache.ClearClientCacheAsync(null);</code>

3.3 GET Operations

3.3.1 General Get Methods

Old API	<pre>public virtual object Get(string key, DataSourceReadOpt dsReadOpt)</pre>
New API	<pre>public virtual object Get(string key, DSReadOption dsReadOption)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); Object value = cache.Get("key", DSReadOption.ReadThru);</pre>

Old API	<pre>public object Get(string key, string group, string subGroup)</pre>
New API	<pre>public virtual object Get(string key, string group, string subGroup, DSReadOption dsReadOption)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); Object value = cache.Get("key", "group", "sub-group" DSReadOption.None);</pre>

Old API	<pre>public virtual object Get(string key, string group, string subGroup, DataSourceReadOpt dsReadOpt)</pre>
New API	<pre>public virtual object Get(string key, string group, string subGroup, DSReadOption dsReadOption)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); Object value = cache.Get("key", "group", "sub-group" DSReadOption.ReadThru);</pre>

Old API	<pre>public virtual object Get(string key, ref object lockId, ref DateTime lockDate, bool acquireLock)</pre>
---------	--

New API	<pre>public virtual object Get(string key, TimeSpan lockTimeout, ref LockHandle lockHandle, bool acquireLock)</pre>
Usage	<p>Lock timeout is a new feature added in NCache 3.6 whereas Lock-Id and Lock-date are combined in <code>LockHandle</code>.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); LockHandle lockHandle = new LockHandle(); object value = Cache.Get("key", Cache.NoLockingExpiration, ref lockHandle, true);</pre>

3.3.2 GetBulk Methods

Old API	<pre>public IDictionary Get(string[] keys)</pre>
New API	<pre>public virtual IDictionary GetBulk(string[] keys, DSReadOption dsReadOption)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); string[] keys = new string[]{"key-1", "key-2"}; IDictionary items = cache.Get(keys, DSReadOption.None);</pre>

Old API	<pre>public virtual IDictionary Get(string[] keys, DataSourceReadOpt dsReadOpt)</pre>
New API	<pre>public virtual IDictionary GetBulk(string[] keys, DSReadOption dsReadOption)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); string[] keys = new string[]{"key-1", "key-2"}; IDictionary items = cache.Get(keys, DSReadOption.ReadThru);</pre>

3.4 Insert Operations

3.4.1 General Insert Methods

Old API	<pre>public void Insert(string key, CacheItem item, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
New API	<pre>public CacheItemVersion Insert(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cache.Insert("key", cacheItem, DSWriteOption.WriteBehind, new DataSourceItemsUpdatedCallback(onDSItemUpdated));</pre>

Old API	<pre>public void Insert(string key, CacheItem item, object lockId, bool releaseLock)</pre>
New API	<pre>public CacheItemVersion Insert(string key, CacheItem item, LockHandle lockHandle, bool releaseLock)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); //acquire lock... LockHandle lockHandle = new LockHandle(); object value = Cache.Get("key", Cache.NoLockingExpiration, ref lockHandle, true); CacheItem cacheItem = new CacheItem("data"); //update the new item and release lock... cache.Insert("key", cacheItem, lockHandle, true);</pre>

Old API	<pre>public void Insert(string key, CacheItem item, string group, string subGroup)</pre>
New API	<pre>public CacheItemVersion Insert(string key,</pre>

	<pre>CacheItem item)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Group = "group"; cacheItem.SubGroup = "sub-group"; cache.Insert("key", cacheItem);</pre>

Old API	<pre>public void Insert(string key, CacheItem item, string group, string subGroup, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
New API	<pre>public CacheItemVersion Insert(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Group = "group"; cacheItem.SubGroup = "sub-group"; cache.Insert("key", cacheItem, DSWriteOption.WriteBehind, new DataSourceItemsUpdatedCallback(onDSItemUpdated));</pre>

Old API	<pre>public void Insert(string key, object value, CacheDependency dependency)</pre>
New API	<pre>public CacheItemVersion Insert(string key, CacheItem item)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path");</pre>

	<code>cache.Insert("key", cacheItem);</code>
--	--

Old API	<pre>public void Insert (string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency)</pre>
New API	<pre>public CacheItemVersion Insert(string key, CacheItem item)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.SyncDependency= new CacheSyncDependency("remoteCacheId", "key"); cache.Insert("key", cacheItem);</pre>

Old API	<pre>public void Insert(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration)</pre>
New API	<pre>public CacheItemVersion Insert(string key, CacheItem item)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.SyncDependency= new CacheSyncDependency("remoteCacheId", "key"); cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cache.Insert("key", cacheItem);</pre>

Old API	<pre>public object Insert(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency,</pre>
---------	--

	<pre> DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback) </pre>
New API	<pre> public CacheItemVersion Insert(string key, CacheItem item) </pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre> Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.SyncDependency= new CacheSyncDependency("remoteCacheId", "key"); cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cache.Insert("key", cacheItem); </pre>

Old API	<pre> public object Insert(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback) </pre>
New API	<pre> public CacheItemVersion Insert(string key, CacheItem item) </pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre> Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file name"); cacheItem.SyncDependency= new CacheSyncDependency("remoteCacheId", "key"); cacheItem.AbsoluteExpiration= DateTime.Now.AddMinutes(5); cacheItem.SlidingExpiration= Cache.NoSlidingExpiration; cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(UpdatedCallback); cache.Insert("key", cacheItem); </pre>

Old API	<pre>public object Insert(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback, bool isResyncExpiredItems, string group, string subGroup)</pre>
New API	<pre>public CacheItemVersion Insert(string key, CacheItem item)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file name"); cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(UpdatedCallback); cacheItem.IsResyncExpiredItems= false; cacheItem.Group = "group"; cacheItem.SubGroup = "sub-group"; cache.Insert("key", cacheItem);</pre>

Old API	<pre>public void Insert(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration)</pre>
New API	<pre>public CacheItemVersion Insert(string key, CacheItem item)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file name"); cacheItem.AbsoluteExpiration= DateTime.Now.AddMinutes(5); cacheItem.SlidingExpiration= Cache.NoSlidingExpiration;</pre>

	<code>cache.Insert("key", cacheItem);</code>
--	--

Old API	<pre>public void Insert(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback)</pre>
New API	<pre>public CacheItemVersion Insert(string key, CacheItem item)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file name"); cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cache.Insert("key", cacheItem);</pre>

Old API	<pre>public void Insert(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback)</pre>
New API	<pre>public CacheItemVersion Insert(string key, CacheItem item)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file name"); cacheItem.AbsoluteExpiration= DateTime.Now.AddMinutes(5); cacheItem.SlidingExpiration= Cache.NoSlidingExpiration; cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback);</pre>

	<pre>cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(UpdatedCallback); cache.Insert("key", cacheItem);</pre>
--	---

Old API	<pre>public void Insert(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, DataSourceUpdateOptions dsUpdateOpts, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback, bool isResyncExpiredItems, string group, string subGroup)</pre>
---------	--

New API	<pre>public CacheItemVersion Insert(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCALLback)</pre>
---------	--

Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file name"); cacheItem.SyncDependency= new CacheSyncDependency("remoteCacheId", "key"); cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(UpdatedCallback); cacheItem.IsResyncExpiredItems= false; cacheItem.Group = "group"; cacheItem.SubGroup = "sub-group"; cache.Insert("key", cacheItem, DSWriteOption.WriteBehind, new DataSourceItemsUpdatedCallback(onDSItemUpdated));</pre>
-------	---

3.4.2 InsertBulk Methods

Old API	<pre>public virtual IDictionary Insert(string[] keys, CacheItem[] items)</pre>
---------	--

New API	<pre>public virtual IDictionary InsertBulk(string[] keys, CacheItem[] items, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem[] items = new CacheItem[5]; string[] keys = new string[5]; for (int i = 0; i < 5; i++) { items[i] = new CacheItem(i); keys[i] = i.ToString(); } cache.Insert(keys, items, DSWriteOption.None, null);</pre>

Old API	<pre>public virtual IDictionary Insert(string[] keys, CacheItem[] items, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
New API	<pre>public virtual IDictionary InsertBulk(string[] keys, CacheItem[] items, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem[] items = new CacheItem[5]; string[] keys = new string[5]; for (int i = 0; i < 5; i++) { items[i] = new CacheItem(i); keys[i] = i.ToString(); } cache.Insert(keys, items, DSWriteOption.WriteThru, new DataSourceItemsUpdatedCallback(onDSItemsUpdated));</pre>

Old API	<pre>public virtual IDictionary Insert(string[] keys, CacheItem[] items, string group, string subGroup)</pre>
New API	<pre>public virtual IDictionary InsertBulk(string[] keys, CacheItem[] items, DSWriteOption dsWriteOption,</pre>

	<pre> DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback) </pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre> Cache cache = NCache.InitializeCache("mycache"); CacheItem[] items = new CacheItem[5]; string[] keys = new string[5]; for (int i = 0; i < 5; i++) { items[i] = new CacheItem(i); items[i].Group="group"; items[i].SubGroup="sub-group"; keys[i] = i.ToString(); } cache.Insert(keys, items, DSWriteOption.None, null); </pre>

Old API	<pre> public virtual IDictionary Insert(string[] keys, CacheItem[] items, string group, string subGroup, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback) </pre>
New API	<pre> public virtual IDictionary InsertBulk(string[] keys, CacheItem[] items, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback) </pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre> Cache cache = NCache.InitializeCache("mycache"); CacheItem[] items = new CacheItem[5]; string[] keys = new string[5]; for (int i = 0; i < 5; i++) { items[i] = new CacheItem(i); items[i].Group="group"; items[i].SubGroup="sub-group"; keys[i] = i.ToString(); } cache.Insert(keys, items, DSWriteOption.WriteThru, new DataSourceItemsUpdatedCallback(onDSItemsUpdated)); </pre>

3.4.3 InsertAsync Methods

Old API	<pre>public void InsertAsync(string key, CacheItem item)</pre>
New API	<pre>public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cache.InsertAsync("key", cacheItem, DSWriteOption.None, null)</pre>

Old API	<pre>public void InsertAsync(string key, CacheItem item, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
New API	<pre>public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cache.InsertAsync("key", cacheItem, DSWriteOption.WriteThru, new DataSourceItemsUpdatedCallback(onDSUpdate))</pre>

Old API	<pre>public void InsertAsync(string key, CacheItem item, string group, string subGroup)</pre>
New API	<pre>public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data");</pre>

```

cacheItem.Group="group";
cacheItem.SubGroup="sub-group";

cache.InsertAsync("key",cacheItem,DSWriteOption.None, null)

```

Old API	<pre> public void InsertAsync(string key, CacheItem item, string group, string subGroup, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback) </pre>
---------	---

New API	<pre> public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback) </pre>
---------	---

Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre> Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Group="group"; cacheItem.SubGroup="sub-group"; cache.InsertAsync("key",cacheItem,DSWriteOption.WriteThru, new DataSourceItemsUpdatedCallback(onDSUpdate)) </pre>
-------	---

Old API	<pre> public void InsertAsync(string key, object value) </pre>
---------	--

New API	<pre> public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback) </pre>
---------	---

Usage	<pre> Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cache.InsertAsync("key",cacheItem,DSWriteOption.None, null) </pre>
-------	---

Old API	<pre> public void InsertAsync(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration,) </pre>
---------	--

	<pre>CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback) </pre>
New API	<pre>public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback) </pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.SyncDependency= null; cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cache.InsertAsync("key", cacheItem, DSWriteOption.None, null) </pre>

Old API	<pre>public object InsertAsync(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback) </pre>
New API	<pre>public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback) </pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.SyncDependency= null; cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(UpdatedCallback); </pre>

	<code>cache.InsertAsync("key", cacheItem, DSWriteOption.None, null)</code>
--	--

Old API	<pre>public object InsertAsync(string key, object value, CacheDependency dependency, CacheSyncDependency syncDependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, DataSourceUpdateOpt dsUpdateOpts, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback, AsyncItemUpdatedCallback onAsyncItemUpdateCallback, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
---------	---

New API	<pre>public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
---------	---

Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.SyncDependency= null; cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(UpdatedCallback); cacheItem.AsyncItemUpdatedCallback = new AsyncItemUpdatedCallback(UpdatedCallBack); DataSourceItemsUpdatedCallback onDSUpdate= new DataSourceItemsUpdatedCallback(onUpdate); cache.InsertAsync("key", cacheItem, DSWriteOption.WriteThru, onDSUpdate)</pre>
-------	--

Old API	<pre>public object InsertAsync(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback</pre>
---------	--

)
New API	<pre>public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.AbsoluteExpiration= Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration= TimeSpan.FromSeconds(20); cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cache.InsertAsync("key",cacheItem, DSWriteOption.None, null)</pre>

Old API	<pre>public object InsertAsync(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback)</pre>
New API	<pre>public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.AbsoluteExpiration = Cache.NoAbsoluteExpiration; cacheItem.SlidingExpiration = TimeSpan.FromSeconds(20); cacheItem.Priority = CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(UpdatedCallback); cache.InsertAsync("key",cacheItem, DSWriteOption.None, null)</pre>

Old API	<pre>public object InsertAsync(string key, object value, CacheDependency dependency, DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority priority, CacheItemRemovedCallback onRemoveCallback, CacheItemUpdatedCallback onUpdateCallback, AsyncItemUpdatedCallback onAsyncItemUpdateCallback,)</pre>
New API	<pre>public void InsertAsync(string key, CacheItem item, DSWriteOption dsWriteOption, DataSourceItemsUpdatedCallback onDataSourceItemUpdatedCallback)</pre>
Usage	<p>CacheItem has the properties for the missing parameters that can be set before adding the CacheItem to the cache.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); CacheItem cacheItem = new CacheItem("data"); cacheItem.Dependency = new CacheDependency("file-path"); cacheItem.AbsoluteExpiration= DateTime.Now.AddMinutes(5); cacheItem.SlidingExpiration= Cache.NoSlidingExpiration; cacheItem.Priority= CacheItemPriority.Normal; cacheItem.ItemRemoveCallback = new CacheItemRemovedCallback(RemovedCallback); cacheItem.ItemUpdateCallback = new CacheItemUpdatedCallback(UpdatedCallback); cacheItem.AsyncItemAddCallback = new AsyncItemAddedCallback(AddedCallBack); cache.InsertAsync("key", cacheItem, DSWriteOption.None, null)</pre>

Old API	<pre>public void InsertAsync(string key, object value, string group, string subGroup)</pre>
New API	<pre>public object InsertAsync(string key, object value, AsyncItemAddedCallback onAsyncItemAddCallback, string group, string subGroup)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); Object cacheObject = new object("Data"); cache.InsertAsync("Key", cacheObject, null, "group-name", "subGroup-name");</pre>

3.5 Remove Operations

3.5.1 General Remove Methods

Old API	<pre>public virtual object Remove(string key, DataSourceUpdateOpt dsUpdateOpt, DataSourceItemsRemovedCallback onDataSourceItemRemovedCallback)</pre>
New API	<pre>public virtual object Remove(string key, DSWriteOption dsWriteOption, DataSourceItemsRemovedCallback onDataSourceItemRemovedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); cache.Remove("key", DSWriteOption.WriteBehind, new DataSourceItemsRemovedCallback(onDSRemoved));</pre>

Old API	<pre>public virtual object Remove(string key, object lockId)</pre>
New API	<pre>public virtual object Remove(string key, LockHandle lockHandle)</pre>
Usage	<p>Lock-Id and Lock-date are combined in LockHandle.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); LockHandle lockHandle = new LockHandle(); cache.Remove("key", lockHandle);</pre>

3.5.2 RemoveBulk Methods

Old API	<pre>public IDictionary Remove(string[] keys)</pre>
New API	<pre>public virtual IDictionary RemoveBulk(string[] keys, DSWriteOption dsWriteOption, DataSourceItemsRemovedCallback onDataSourceItemsRemovedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); string[] keys = new string[]{"key-1", "key-2"}; cache.RemoveBulk(keys, DSWriteOption.None, null);</pre>

Old API	<pre>public virtual IDictionary Remove(string[] keys, DataSourceUpdateOpt dsUpdateOpts, DataSourceItemsRemovedCallback onDataSourceItemsRemovedCallback)</pre>
New API	<pre>public virtual IDictionary RemoveBulk(string[] keys, DSWriteOption dsWriteOption, DataSourceItemsRemovedCallback onDataSourceItemsRemovedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); string[] keys = new string[]{"key-1", "key-2"}; cache.RemoveBulk(keys, DSWriteOption.WriteThru, new DataSourceItemsRemovedCallback(onDSRemoved));</pre>

3.5.3 RemoveAsync Methods

Old API	<pre>public virtual void RemoveAsync(string key)</pre>
New API	<pre>public virtual void RemoveAsync(string key, AsyncItemRemovedCallback onAsyncItemRemoveCallback, DSWriteOption dsWriteOption, DataSourceItemsRemovedCallback onDataSourceItemRemovedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); cache.RemoveAsync("key", null, DSWriteOption.None, null);</pre>

Old API	<pre>public virtual void RemoveAsync(string key, AsyncItemRemovedCallback onAsyncItemRemoveCallback)</pre>
---------	--

New API	<pre>public virtual void RemoveAsync(string key, AsyncItemRemovedCallback onAsyncItemRemoveCallback, DSWriteOption dsWriteOption, DataSourceItemsRemovedCallback onDataSourceItemRemovedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); cache.RemoveAsync("key", new AsyncItemRemovedCallback(onRemove), DSWriteOption.None, null);</pre>

Old API	<pre>public virtual void RemoveAsync(string key, DataSourceUpdateOpt dsUpdateOpt, AsyncItemRemovedCallback onAsyncItemRemoveCallback, DataSourceItemsRemovedCallback onDataSourceItemRemovedCallback)</pre>
New API	<pre>public virtual void RemoveAsync(string key, AsyncItemRemovedCallback onAsyncItemRemoveCallback, DSWriteOption dsWriteOption, DataSourceItemsRemovedCallback onDataSourceItemRemovedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); cache.RemoveAsync("key", new AsyncItemRemovedCallback(onRemove), DSWriteOption.WriteThru, new DataSourceItemsRemovedCallback(onDSRemoved));</pre>

Old API	<pre>public virtual void RemoveAsync(string key, DataSourceUpdateOpt dsUpdateOpt, DataSourceItemsRemovedCallback onDataSourceItemRemovedCallback)</pre>
New API	<pre>public virtual void RemoveAsync(string key, AsyncItemRemovedCallback onAsyncItemRemoveCallback, DSWriteOption dsWriteOption, DataSourceItemsRemovedCallback onDataSourceItemRemovedCallback)</pre>
Usage	<pre>Cache cache = NCache.InitializeCache("mycache"); cache.RemoveAsync("key", null, DSWriteOption.WriteThru, new DataSourceItemsRemovedCallback(onDSRemoved));</pre>

3.6 Unlock Operations

3.6.1 General UnLock Method

Old API	<pre>public virtual void Unlock(string key, object lockId)</pre>
New API	<pre>public virtual void Unlock(string key, LockHandle lockHandle)</pre>
Usage	<p>Lock-Id and Lock-date are combined in <code>LockHandle</code>.</p> <pre>Cache cache = NCache.InitializeCache("mycache"); LockHandle lockHandle = new LockHandle(); cache.Unlock("key",lockHandle);</pre>