

# Patcher Javascript API

3.2.1.35

Generated by Doxygen 1.7.4

Fri Mar 16 2012 16:35:14



# Contents

<b>1</b>	<b>Patcher Javascript API</b>	<b>1</b>
1.1	Overview . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	Patch Class Reference . . . . .	7
4.1.1	Detailed Description . . . . .	8
4.1.2	Member Function Documentation . . . . .	8
4.1.2.1	fail . . . . .	8
4.1.2.2	getBytesLeft . . . . .	8
4.1.2.3	getCurrentItemCompressedSize . . . . .	8
4.1.2.4	getCurrentItemDiskFileName . . . . .	8
4.1.2.5	getCurrentItemDiskOffset . . . . .	9
4.1.2.6	getFileName . . . . .	9
4.1.2.7	getOutputDirectory . . . . .	9
4.1.2.8	getRemainingTime . . . . .	9
4.1.2.9	getStalled . . . . .	9
4.1.2.10	getTotalBytes . . . . .	9
4.1.2.11	getWriteRates . . . . .	10
4.1.2.12	release . . . . .	10
4.1.2.13	resume . . . . .	10

4.1.2.14	<a href="#">setFileName</a>	10
4.1.2.15	<a href="#">setOutputDirectory</a>	10
4.1.2.16	<a href="#">skip</a>	10
4.1.2.17	<a href="#">start</a>	10
4.1.3	<a href="#">Member Data Documentation</a>	10
4.1.3.1	<a href="#">instanceId</a>	11
4.1.3.2	<a href="#">onComplete</a>	11
4.1.3.3	<a href="#">onCorruptDestFile</a>	11
4.1.3.4	<a href="#">onCorruptSourceFile</a>	11
4.1.3.5	<a href="#">onDidPatchFile</a>	11
4.1.3.6	<a href="#">onFileError</a>	12
4.1.3.7	<a href="#">onPrepareComplete</a>	12
4.1.3.8	<a href="#">onPreparePatchFile</a>	12
4.1.3.9	<a href="#">onProgress</a>	12
4.1.3.10	<a href="#">onStart</a>	12
4.1.3.11	<a href="#">onWillPatchFile</a>	12
4.2	<a href="#">PatchManager Class Reference</a>	13
4.2.1	<a href="#">Detailed Description</a>	14
4.2.2	<a href="#">Member Function Documentation</a>	14
4.2.2.1	<a href="#">addManifestAddedEventHandler</a>	14
4.2.2.2	<a href="#">addManifestCompleteEventHandler</a>	14
4.2.2.3	<a href="#">addManifestDependencyEventHandler</a>	14
4.2.2.4	<a href="#">addManifestRemovedEventHandler</a>	15
4.2.2.5	<a href="#">addManifestRequiredCompleteEventHandler</a>	15
4.2.2.6	<a href="#">addManifestStartEventHandler</a>	15
4.2.2.7	<a href="#">addManifestStateChangeEventHandler</a>	15
4.2.2.8	<a href="#">addManifestVerifyCompleteEventHandler</a>	15
4.2.2.9	<a href="#">addManifestVerifyProgressEventHandler</a>	16
4.2.2.10	<a href="#">addManifestVerifyStartEventHandler</a>	16
4.2.2.11	<a href="#">addPatchCompleteEventHandler</a>	16
4.2.2.12	<a href="#">addPatchCorruptDestFileEventHandler</a>	16
4.2.2.13	<a href="#">addPatchCorruptSourceFileEventHandler</a>	16
4.2.2.14	<a href="#">addPatchDidPatchFileEventHandler</a>	16
4.2.2.15	<a href="#">addPatchDownloadErrorEventHandler</a>	17

4.2.2.16	<a href="#">addPatchFileErrorEventHandler</a>	17
4.2.2.17	<a href="#">addPatchManifest</a>	17
4.2.2.18	<a href="#">addPatchPrepareCompleteEventHandler</a>	17
4.2.2.19	<a href="#">addPatchPreparePatchFileEventHandler</a>	17
4.2.2.20	<a href="#">addPatchProgressEventHandler</a>	18
4.2.2.21	<a href="#">addPatchStartEventHandler</a>	18
4.2.2.22	<a href="#">addPatchWillPatchFileEventHandler</a>	18
4.2.2.23	<a href="#">manifestByld</a>	18
4.2.2.24	<a href="#">release</a>	18
4.2.2.25	<a href="#">removeManifestAddedEventHandler</a>	18
4.2.2.26	<a href="#">removeManifestCompleteEventHandler</a>	19
4.2.2.27	<a href="#">removeManifestDependencyEventHandler</a>	19
4.2.2.28	<a href="#">removeManifestRemovedEventHandler</a>	19
4.2.2.29	<a href="#">removeManifestRequiredCompleteEventHandler</a>	19
4.2.2.30	<a href="#">removeManifestStartEventHandler</a>	19
4.2.2.31	<a href="#">removeManifestStateChangeEventHandler</a>	19
4.2.2.32	<a href="#">removeManifestVerifyCompleteEventHandler</a>	20
4.2.2.33	<a href="#">removeManifestVerifyProgressEventHandler</a>	20
4.2.2.34	<a href="#">removeManifestVerifyStartEventHandler</a>	20
4.2.2.35	<a href="#">removePatchCompleteEventHandler</a>	20
4.2.2.36	<a href="#">removePatchCorruptDestFileEventHandler</a>	20
4.2.2.37	<a href="#">removePatchCorruptSourceFileEventHandler</a>	20
4.2.2.38	<a href="#">removePatchDidPatchFileEventHandler</a>	21
4.2.2.39	<a href="#">removePatchDownloadErrorEventHandler</a>	21
4.2.2.40	<a href="#">removePatchFileErrorEventHandler</a>	21
4.2.2.41	<a href="#">removePatchManifest</a>	21
4.2.2.42	<a href="#">removePatchPrepareCompleteEventHandler</a>	21
4.2.2.43	<a href="#">removePatchPreparePatchFileEventHandler</a>	22
4.2.2.44	<a href="#">removePatchProgressEventHandler</a>	22
4.2.2.45	<a href="#">removePatchStartEventHandler</a>	22
4.2.2.46	<a href="#">removePatchWillPatchFileEventHandler</a>	22
4.2.2.47	<a href="#">repairPatchManifest</a>	22
4.2.2.48	<a href="#">rootManifestByld</a>	23
4.2.2.49	<a href="#">rootManifestIdByld</a>	23

4.2.2.50	<a href="#">verifyManifestByld</a>	23
4.3	<a href="#">PatchManifest Class Reference</a>	23
4.3.1	<a href="#">Detailed Description</a>	24
4.3.2	<a href="#">Member Function Documentation</a>	24
4.3.2.1	<a href="#">dependencyFail</a>	24
4.3.2.2	<a href="#">expandString</a>	25
4.3.2.3	<a href="#">fail</a>	25
4.3.2.4	<a href="#">getCurrentReleaseld</a>	25
4.3.2.5	<a href="#">getExtraDataValue</a>	25
4.3.2.6	<a href="#">getLastError</a>	25
4.3.2.7	<a href="#">getLocal</a>	26
4.3.2.8	<a href="#">getMaintenance</a>	26
4.3.2.9	<a href="#">getName</a>	26
4.3.2.10	<a href="#">getOutput</a>	26
4.3.2.11	<a href="#">getReleaseName</a>	26
4.3.2.12	<a href="#">getRequiredReleaseld</a>	27
4.3.2.13	<a href="#">getRequiresElevation</a>	27
4.3.2.14	<a href="#">getRetryInterval</a>	27
4.3.2.15	<a href="#">getSourceReleaseld</a>	27
4.3.2.16	<a href="#">getSourceReleaseName</a>	27
4.3.2.17	<a href="#">getSourceReleaseSHA1</a>	27
4.3.2.18	<a href="#">getTargetDirectory</a>	28
4.3.2.19	<a href="#">getTargetReleaseld</a>	28
4.3.2.20	<a href="#">getTargetReleaseName</a>	28
4.3.2.21	<a href="#">getTargetReleaseSHA1</a>	28
4.3.2.22	<a href="#">getUpcomingReleaseld</a>	28
4.3.2.23	<a href="#">getUrl</a>	28
4.3.2.24	<a href="#">release</a>	29
4.3.2.25	<a href="#">resume</a>	29
4.3.2.26	<a href="#">setLocal</a>	29
4.3.2.27	<a href="#">setOutput</a>	29
4.3.2.28	<a href="#">setRetryInterval</a>	29
4.3.2.29	<a href="#">setSourceReleaseld</a>	29
4.3.2.30	<a href="#">setSourceReleaseName</a>	29

4.3.2.31	setSourceReleaseSHA1	30
4.3.2.32	setTargetDirectory	30
4.3.2.33	setTargetReleaseId	30
4.3.2.34	setTargetReleaseName	30
4.3.2.35	setTargetReleaseSHA1	30
4.3.2.36	setUrl	31
4.3.2.37	start	31
4.3.3	Member Data Documentation	31
4.3.3.1	instanceId	31
4.3.3.2	onComplete	31
4.3.3.3	onDependency	31
4.3.3.4	onLoadComplete	31
4.3.3.5	onPatch	31
4.3.3.6	onStart	32
4.4	PatchManifestError Class Reference	32
4.4.1	Detailed Description	32
4.4.2	Member Function Documentation	32
4.4.2.1	nameFromId	32
4.4.3	Member Data Documentation	33
4.4.3.1	DEPENDENCYFAIL	33
4.4.3.2	DOWNLOADFAIL	33
4.4.3.3	INVALIDARGUMENT	33
4.4.3.4	MAINTENANCE	33
4.4.3.5	NONE	33
4.4.3.6	REQUIRESELEVATION	33
4.4.3.7	SOURCERELEASEUNDEFINED	33
4.4.3.8	TARGETRELEASEUNDEFINED	33
4.4.3.9	UNZIPFAILED	33
4.4.3.10	UPDATEPATHUNAVAILABLE	34
4.4.3.11	USERFAIL	34
4.4.3.12	VERSIONFAIL	34
4.4.3.13	XMLPARSEFAIL	34
4.5	PatchManifestState Class Reference	34
4.5.1	Detailed Description	34

4.5.2	Member Function Documentation	35
4.5.2.1	nameFromId	35
4.5.3	Member Data Documentation	35
4.5.3.1	COMPLETE	35
4.5.3.2	DEPENDENCY	35
4.5.3.3	DOWNLOAD	35
4.5.3.4	GETMANIFEST	35
4.5.3.5	PATCH	35
4.6	PatchVersion Class Reference	35
4.6.1	Detailed Description	36
4.6.2	Member Function Documentation	36
4.6.2.1	getFileName	36
4.6.2.2	getSha1	36
4.6.2.3	release	36
4.6.2.4	resume	36
4.6.2.5	setFileName	37
4.6.2.6	start	37
4.6.3	Member Data Documentation	37
4.6.3.1	instanceId	37
4.6.3.2	onComplete	37
4.6.3.3	onFile	37
4.6.3.4	onStart	37
4.7	PatchVersionVerify Class Reference	38
4.7.1	Detailed Description	38
4.7.2	Member Function Documentation	38
4.7.2.1	getFileName	38
4.7.2.2	getQuickScan	38
4.7.2.3	release	39
4.7.2.4	setFileName	39
4.7.2.5	setQuickScan	39
4.7.2.6	start	39
4.7.3	Member Data Documentation	39
4.7.3.1	instanceId	39
4.7.3.2	onComplete	39



4.7.3.3	onCorruptFile	39
4.7.3.4	onProgress	40
4.7.3.5	onStart	40
<b>5</b>	<b>File Documentation</b>	<b>41</b>
5.1	patch.js File Reference	41
5.1.1	Detailed Description	41
5.1.2	Function Documentation	41
5.1.2.1	createPatch	41
5.2	patchmanager.js File Reference	41
5.2.1	Detailed Description	42
5.2.2	Variable Documentation	42
5.2.2.1	patchManager	42
5.3	patchmanifest.js File Reference	42
5.3.1	Detailed Description	42
5.3.2	Function Documentation	42
5.3.2.1	createPatchManifest	42
5.4	patchmanifesterror.js File Reference	43
5.4.1	Detailed Description	43
5.4.2	Variable Documentation	43
5.4.2.1	patchManifestError	43
5.5	patchmanifeststate.js File Reference	43
5.5.1	Detailed Description	43
5.5.2	Variable Documentation	44
5.5.2.1	patchManifestState	44
5.6	patchversion.js File Reference	44
5.6.1	Detailed Description	44
5.6.2	Function Documentation	44
5.6.2.1	createPatchVersion	44
5.7	patchversionverify.js File Reference	44
5.7.1	Detailed Description	45
5.7.2	Function Documentation	45
5.7.2.1	createPatchVersionVerify	45



## Chapter 1

# Patcher Javascript API

### 1.1 Overview

The patcher javascript API allows users to integrate with objects implemented using the patcher API.

The usage of this API is restricted to entities which have signed a license agreement with Solid State Networks, Inc.

Licenses are valid for only one PRODUCT usage. Please see the license agreement for further details.



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Patch</a> (Extract a patch ) . . . . .	7
<a href="#">PatchManager</a> (Manages patches and provides an integrated event structure )	13
<a href="#">PatchManifest</a> (Download/process a patch manifest ) . . . . .	23
<a href="#">PatchManifestError</a> ( <a href="#">Patch</a> manifest error constants and helper functions ) . . .	32
<a href="#">PatchManifestState</a> ( <a href="#">Patch</a> manifest state constants and helper functions ) . .	34
<a href="#">PatchVersion</a> (Enumerate a version file ) . . . . .	35
<a href="#">PatchVersionVerify</a> (Verify a version file ) . . . . .	38



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">patch.js</a> (File containing <a href="#">Patch</a> class and creation function ) . . . . .	41
<a href="#">patchmanager.js</a> (File containing patch manager ) . . . . .	41
<a href="#">patchmanifest.js</a> (File containing <a href="#">PatchManifest</a> class and creation function ) .	42
<a href="#">patchmanifesterror.js</a> (File containing patch manifest error constants and helper functions ) . . . . .	43
<a href="#">patchmanifeststate.js</a> (File containing patch manifest state constants and helper functions ) . . . . .	43
<a href="#">patchversion.js</a> (File containing <a href="#">PatchVersion</a> class and creation function ) . .	44
<a href="#">patchversionverify.js</a> (File containing <a href="#">PatchVersionVerify</a> class and creation function ) . . . . .	44





## Chapter 4

# Class Documentation

### 4.1 Patch Class Reference

Extract a patch.

#### Public Member Functions

- void [fail](#) ()
- string [getBytesLeft](#) ()
- string [getCurrentItemCompressedSize](#) ()
- string [getCurrentItemDiskFileName](#) ()
- string [getCurrentItemDiskOffset](#) ()
- string [getFileName](#) ()
- string [getOutputDirectory](#) ()
- double [getRemainingTime](#) ()
- bool [getStalled](#) ()
- string [getTotalBytes](#) ()
- string [getWriteRates](#) ()
- void [release](#) ()
- void [resume](#) ()
- void [setFileName](#) (string value)
- void [setOutputDirectory](#) (string value)
- void [skip](#) ()
- void [start](#) ()

#### Public Attributes

- string [instanceId](#)
- event [onComplete](#)
- event [onCorruptDestFile](#)
- event [onCorruptSourceFile](#)

- event [onDidPatchFile](#)
- event [onFileError](#)
- event [onPrepareComplete](#)
- event [onPreparePatchFile](#)
- event [onProgress](#)
- event [onStart](#)
- event [onWillPatchFile](#)

#### 4.1.1 Detailed Description

Extract a patch.

#### 4.1.2 Member Function Documentation

##### 4.1.2.1 void Patch.fail ( )

Fail the patch after an action has stalled

##### 4.1.2.2 string Patch.getBytesLeft ( )

gets the bytes left in the patch (64-bit number)

##### Returns

bytes left in the patch (64-bit number)

##### 4.1.2.3 string Patch.getCurrentItemCompressedSize ( )

gets the compressed size of the file in the zip

##### Returns

compressed size of the file in the zip

##### 4.1.2.4 string Patch.getCurrentItemDiskFileName ( )

gets the the zip file currently being used to patched

##### Returns

the zip file currently being used to patched

#### 4.1.2.5 string Patch.getCurrentItemDiskOffset ( )

gets the offset in the zip where the patch file exists

##### Returns

offset in the zip where the patch file exists

#### 4.1.2.6 string Patch.GetFileName ( )

gets the file name of path

##### Returns

file name of path

#### 4.1.2.7 string Patch.getOutputDirectory ( )

gets the location where patch is applied

##### Returns

location where patch is applied

#### 4.1.2.8 double Patch.getRemainingTime ( )

gets the estimated time remaining in a patch

##### Returns

estimated time remaining in a patch

#### 4.1.2.9 bool Patch.getStalled ( )

gets the patch stalled state

##### Returns

patch stalled state

#### 4.1.2.10 string Patch.getTotalBytes ( )

gets the size of the patch (64-bit number)

##### Returns

size of the patch (64-bit number)

#### 4.1.2.11 string Patch.getWriteRates ( )

gets the json object containing the rate of the current/average/max throughput for patch writes

##### Returns

json object containing the rate of the current/average/max throughput for patch writes

#### 4.1.2.12 void Patch.release ( )

releases the object

#### 4.1.2.13 void Patch.resume ( )

Resume the patch file after an action has stalled

#### 4.1.2.14 void Patch.setFileName ( string *value* )

sets the file name of path

##### Parameters

<i>value</i>	file name of path
--------------	-------------------

#### 4.1.2.15 void Patch.setOutputDirectory ( string *value* )

sets the location where patch is applied

##### Parameters

<i>value</i>	location where patch is applied
--------------	---------------------------------

#### 4.1.2.16 void Patch.skip ( )

Skip the patch file after an action has stalled

#### 4.1.2.17 void Patch.start ( )

Starts the patch manifest

### 4.1.3 Member Data Documentation

#### 4.1.3.1 string **Patch.instanceId**

Instance id used to bind proxy object to native object

#### 4.1.3.2 event **Patch.onComplete**

Triggered when the patch is complete

##### Parameters

<i>successful</i>	true if the patch was successful, false otherwise
-------------------	---

#### 4.1.3.3 event **Patch.onCorruptDestFile**

Triggered when the patch has detected a corrupt dest file

##### Parameters

<i>name</i>	Name of the corrupt file
<i>expected-SHA1</i>	SHA1 the dest file is supposed to be
<i>SHA1</i>	SHA1 of the dest file

#### 4.1.3.4 event **Patch.onCorruptSourceFile**

Triggered when the patch has detected a corrupt source file

##### Parameters

<i>name</i>	Name of the corrupt file
<i>expected-SHA1</i>	SHA1 the source file is supposed to be
<i>SHA1</i>	SHA1 of the source file

#### 4.1.3.5 event **Patch.onDidPatchFile**

Triggered after a file is patched

##### Parameters

<i>name</i>	Name of the file to patched
<i>successful</i>	was the patch successfully applied

#### 4.1.3.6 event **Patch.onFileError**

Triggered when the patch encounters a file error

##### Parameters

<i>name</i>	File where the error was generated
<i>error</i>	Number value representing file error

#### 4.1.3.7 event **Patch.onPrepareComplete**

Triggered when the patch preparation is complete

##### Parameters

<i>successful</i>	true if the patch preparation was successful, false otherwise
-------------------	---

#### 4.1.3.8 event **Patch.onPreparePatchFile**

Triggered before patch starts on each file in the patch

##### Parameters

<i>name</i>	Name of the file to be patched
-------------	--------------------------------

#### 4.1.3.9 event **Patch.onProgress**

Triggered when the patch has made progress

##### Parameters

<i>percent</i>	(-1.0: Still calculating) (0.0 to 1.0: Percent completed)
----------------	---

#### 4.1.3.10 event **Patch.onStart**

Triggered when patch is started

#### 4.1.3.11 event **Patch.onWillPatchFile**

Triggered before a file is patched

##### Parameters

<i>name</i>	Name of the file to be patched
-------------	--------------------------------

### Returns

bool: true to patch file, false to skip

The documentation for this class was generated from the following file:

- [patch.js](#)

## 4.2 PatchManager Class Reference

Manages patches and provides an integrated event structure.

### Public Member Functions

- void [addManifestAddedEventHandler](#) (event handler)
- void [addManifestCompleteEventHandler](#) (event handler)
- void [addManifestDependencyEventHandler](#) (event handler)
- void [addManifestRemovedEventHandler](#) (event handler)
- void [addManifestRequiredCompleteEventHandler](#) (event handler)
- void [addManifestStartEventHandler](#) (event handler)
- void [addManifestStateChangeEventHandler](#) (event handler)
- void [addManifestVerifyCompleteEventHandler](#) (event handler)
- void [addManifestVerifyProgressEventHandler](#) (event handler)
- void [addManifestVerifyStartEventHandler](#) (event handler)
- void [addPatchCompleteEventHandler](#) (event handler)
- void [addPatchCorruptDestFileEventHandler](#) (event handler)
- void [addPatchCorruptSourceFileEventHandler](#) (event handler)
- void [addPatchDidPatchFileEventHandler](#) (event handler)
- void [addPatchDownloadErrorEventHandler](#) (event handler)
- void [addPatchFileErrorEventHandler](#) (event handler)
- patchManifestItem [addPatchManifest](#) (string manifestUrl)
- void [addPatchPrepareCompleteEventHandler](#) (event handler)
- void [addPatchPreparePatchFileEventHandler](#) (event handler)
- void [addPatchProgressEventHandler](#) (event handler)
- void [addPatchStartEventHandler](#) (event handler)
- void [addPatchWillPatchFileEventHandler](#) (event handler)
- void [manifestById](#) (string id)
- void [release](#) ()
- void [removeManifestAddedEventHandler](#) (event handler)
- void [removeManifestCompleteEventHandler](#) (event handler)
- void [removeManifestDependencyEventHandler](#) (event handler)
- void [removeManifestRemovedEventHandler](#) (event handler)
- void [removeManifestRequiredCompleteEventHandler](#) (event handler)
- void [removeManifestStartEventHandler](#) (event handler)
- void [removeManifestStateChangeEventHandler](#) (event handler)

- void [removeManifestVerifyCompleteEventHandler](#) (event handler)
- void [removeManifestVerifyProgressEventHandler](#) (event handler)
- void [removeManifestVerifyStartEventHandler](#) (event handler)
- void [removePatchCompleteEventHandler](#) (event handler)
- void [removePatchCorruptDestFileEventHandler](#) (event handler)
- void [removePatchCorruptSourceFileEventHandler](#) (event handler)
- void [removePatchDidPatchFileEventHandler](#) (event handler)
- void [removePatchDownloadErrorEventHandler](#) (event handler)
- void [removePatchFileErrorEventHandler](#) (event handler)
- bool [removePatchManifest](#) (string id)
- void [removePatchPrepareCompleteEventHandler](#) (event handler)
- void [removePatchPreparePatchFileEventHandler](#) (event handler)
- void [removePatchProgressEventHandler](#) (event handler)
- void [removePatchStartEventHandler](#) (event handler)
- void [removePatchWillPatchFileEventHandler](#) (event handler)
- patchManifestItem [repairPatchManifest](#) (string manifestUrl, array maskedFiles)
- void [rootManifestById](#) (string id)
- void [rootManifestIdById](#) (string id)
- void [verifyManifestById](#) (string manifestId, bool quickScan)

#### 4.2.1 Detailed Description

Manages patches and provides an integrated event structure.

#### 4.2.2 Member Function Documentation

##### 4.2.2.1 void PatchManager.addManifestAddedEventHandler ( event handler )

subscribe to the ManifestAdded event

###### Parameters

<i>handler</i>	callback function for ManifestAdded event
----------------	---

##### 4.2.2.2 void PatchManager.addManifestCompleteEventHandler ( event handler )

subscribe to the ManifestComplete event

###### Parameters

<i>handler</i>	callback function for ManifestComplete event
----------------	--

##### 4.2.2.3 void PatchManager.addManifestDependencyEventHandler ( event handler )

subscribe to the ManifestDependency event



**Parameters**

<i>handler</i>	callback function for ManifestDependency event
----------------	--

**4.2.2.4 void PatchManager.addManifestRemovedEventHandler ( event *handler* )**

subscribe to the ManifestRemoved event

**Parameters**

<i>handler</i>	callback function for ManifestRemoved event
----------------	---

**4.2.2.5 void PatchManager.addManifestRequiredCompleteEventHandler ( event *handler* )**

subscribe to the ManifestRequiredComplete event

**Parameters**

<i>handler</i>	callback function for ManifestRequiredComplete event
----------------	--

**4.2.2.6 void PatchManager.addManifestStartEventHandler ( event *handler* )**

subscribe to the ManifestStart event

**Parameters**

<i>handler</i>	callback function for ManifestStart event
----------------	---

**4.2.2.7 void PatchManager.addManifestStateChangeEventHandler ( event *handler* )**

subscribe to the ManifestStateChange event

**Parameters**

<i>handler</i>	callback function for ManifestStateChange event
----------------	---

**4.2.2.8 void PatchManager.addManifestVerifyCompleteEventHandler ( event *handler* )**

subscribe to the ManifestVerifyComplete event

**Parameters**

<i>handler</i>	callback function for ManifestVerifyComplete event
----------------	--

**4.2.2.9 void PatchManager.addManifestVerifyProgressEventHandler ( event *handler* )**

subscribe to the ManifestVerifyProgress event

**Parameters**

<i>handler</i>	callback function for ManifestVerifyProgress event
----------------	--

**4.2.2.10 void PatchManager.addManifestVerifyStartEventHandler ( event *handler* )**

subscribe to the ManifestVerifyStart event

**Parameters**

<i>handler</i>	callback function for ManifestVerifyStart event
----------------	---

**4.2.2.11 void PatchManager.addPatchCompleteEventHandler ( event *handler* )**

subscribe to the PatchComplete event

**Parameters**

<i>handler</i>	callback function for PatchComplete event
----------------	---

**4.2.2.12 void PatchManager.addPatchCorruptDestFileEventHandler ( event *handler* )**

subscribe to the PatchCorruptDestFile event

**Parameters**

<i>handler</i>	callback function for PatchCorruptDestFile event
----------------	--

**4.2.2.13 void PatchManager.addPatchCorruptSourceFileEventHandler ( event *handler* )**

subscribe to the PatchCorruptSourceFile event

**Parameters**

<i>handler</i>	callback function for PatchCorruptSourceFile event
----------------	--

**4.2.2.14 void PatchManager.addPatchDidPatchFileEventHandler ( event *handler* )**

subscribe to the PatchDidPatchFile event

**Parameters**

<i>handler</i>	callback function for PatchDidPatchFile event
----------------	---

**4.2.2.15 void PatchManager.addPatchDownloadErrorEventHandler ( event *handler* )**

subscribe to the PatchDownloadError event

**Parameters**

<i>handler</i>	callback function for PatchDownloadError event
----------------	--

**4.2.2.16 void PatchManager.addPatchFileErrorEventHandler ( event *handler* )**

subscribe to the PatchFileError event

**Parameters**

<i>handler</i>	callback function for PatchFileError event
----------------	--

**4.2.2.17 patchManifestItem PatchManager.addPatchManifest ( string *manifestUrl* )**

adds patch to the manager

**Parameters**

<i>manifestUrl</i>	url to the manifest file
--------------------	--------------------------

**Returns**

patchManifestItem containing the id and patch manifest

**4.2.2.18 void PatchManager.addPatchPrepareCompleteEventHandler ( event *handler* )**

subscribe to the PatchPrepareComplete event

**Parameters**

<i>handler</i>	callback function for PatchPrepareComplete event
----------------	--

**4.2.2.19 void PatchManager.addPatchPreparePatchFileEventHandler ( event *handler* )**

subscribe to the PatchPreparePatchFile event

**Parameters**

<i>handler</i>	callback function for PatchPreparePatchFile event
----------------	---

**4.2.2.20 void PatchManager.addPatchProgressEventHandler ( event *handler* )**

subscribe to the PatchProgress event

**Parameters**

<i>handler</i>	callback function for PatchProgress event
----------------	---

**4.2.2.21 void PatchManager.addPatchStartEventHandler ( event *handler* )**

subscribe to the PatchStart event

**Parameters**

<i>handler</i>	callback function for PatchStart event
----------------	--

**4.2.2.22 void PatchManager.addPatchWillPatchFileEventHandler ( event *handler* )**

subscribe to the PatchWillPatchFile event

**Parameters**

<i>handler</i>	callback function for PatchWillPatchFile event
----------------	--

**4.2.2.23 void PatchManager.manifestById ( string *id* )**

returns the patch manifest by the assigned id

**Parameters**

<i>id</i>	patch manifest id assigned by manager
-----------	---------------------------------------

**4.2.2.24 void PatchManager.release ( )**

releases the download manager

**4.2.2.25 void PatchManager.removeManifestAddedEventHandler ( event *handler* )**

unsubscribe from the ManifestAdded event

**Parameters**

<i>handler</i>	callback function for ManifestAdded event
----------------	---

**4.2.2.26 void PatchManager.removeManifestCompleteEventHandler ( event *handler* )**

unsubscribe from the ManifestComplete event

**Parameters**

<i>handler</i>	callback function for ManifestComplete event
----------------	--

**4.2.2.27 void PatchManager.removeManifestDependencyEventHandler ( event *handler* )**

unsubscribe from the ManifestDependency event

**Parameters**

<i>handler</i>	callback function for ManifestDependency event
----------------	--

**4.2.2.28 void PatchManager.removeManifestRemovedEventHandler ( event *handler* )**

unsubscribe from the ManifestRemoved event

**Parameters**

<i>handler</i>	callback function for ManifestRemoved event
----------------	---

**4.2.2.29 void PatchManager.removeManifestRequiredCompleteEventHandler ( event *handler* )**

unsubscribe from the ManifestRequiredComplete event

**Parameters**

<i>handler</i>	callback function for ManifestRequiredComplete event
----------------	--

**4.2.2.30 void PatchManager.removeManifestStartEventHandler ( event *handler* )**

unsubscribe from the ManifestStart event

**Parameters**

<i>handler</i>	callback function for ManifestStart event
----------------	---

**4.2.2.31 void PatchManager.removeManifestStateChangeEventHandler ( event *handler* )**

unsubscribe from the ManifestStateChange event

**Parameters**

<i>handler</i>	callback function for ManifestStateChange event
----------------	---

**4.2.2.32 void PatchManager.removeManifestVerifyCompleteEventHandler ( event *handler* )**

unsubscribe from the ManifestVerifyComplete event

**Parameters**

<i>handler</i>	callback function for ManifestVerifyComplete event
----------------	--

**4.2.2.33 void PatchManager.removeManifestVerifyProgressEventHandler ( event *handler* )**

unsubscribe from the ManifestVerifyProgress event

**Parameters**

<i>handler</i>	callback function for ManifestVerifyProgress event
----------------	--

**4.2.2.34 void PatchManager.removeManifestVerifyStartEventHandler ( event *handler* )**

unsubscribe from the ManifestVerifyStart event

**Parameters**

<i>handler</i>	callback function for ManifestVerifyStart event
----------------	---

**4.2.2.35 void PatchManager.removePatchCompleteEventHandler ( event *handler* )**

unsubscribe from the PatchComplete event

**Parameters**

<i>handler</i>	callback function for PatchComplete event
----------------	---

**4.2.2.36 void PatchManager.removePatchCorruptDestFileEventHandler ( event *handler* )**

unsubscribe from the PatchCorruptDestFile event

**Parameters**

<i>handler</i>	callback function for PatchCorruptDestFile event
----------------	--

**4.2.2.37 void PatchManager.removePatchCorruptSourceFileEventHandler ( event *handler* )**

unsubscribe from the PatchCorruptSourceFile event

**Parameters**

<i>handler</i>	callback function for PatchCorruptSourceFile event
----------------	--

4.2.2.38 void PatchManager.removePatchDidPatchFileEventHandler ( event *handler* )

unsubscribe from the PatchDidPatchFile event

**Parameters**

<i>handler</i>	callback function for PatchDidPatchFile event
----------------	---

4.2.2.39 void PatchManager.removePatchDownloadErrorEventHandler ( event *handler* )

unsubscribe from the PatchDownloadError event

**Parameters**

<i>handler</i>	callback function for PatchDownloadError event
----------------	--

4.2.2.40 void PatchManager.removePatchFileErrorEventHandler ( event *handler* )

unsubscribe from the PatchFileError event

**Parameters**

<i>handler</i>	callback function for PatchFileError event
----------------	--

4.2.2.41 bool PatchManager.removePatchManifest ( string *id* )

removes a patch manifest from the manager

**Parameters**

<i>id</i>	id of the patch manifest to remove
-----------	------------------------------------

**Returns**

true if the item was successfully removed, false otherwise

4.2.2.42 void PatchManager.removePatchPrepareCompleteEventHandler ( event *handler* )

unsubscribe from the PatchPrepareComplete event

**Parameters**

<i>handler</i>	callback function for PatchPrepareComplete event
----------------	--

**4.2.2.43 void PatchManager.removePatchPreparePatchFileEventHandler ( event handler )**

unsubscribe from the PatchPreparePatchFile event

**Parameters**

<i>handler</i>	callback function for PatchPreparePatchFile event
----------------	---

**4.2.2.44 void PatchManager.removePatchProgressEventHandler ( event handler )**

unsubscribe from the PatchProgress event

**Parameters**

<i>handler</i>	callback function for PatchProgress event
----------------	---

**4.2.2.45 void PatchManager.removePatchStartEventHandler ( event handler )**

unsubscribe from the PatchStart event

**Parameters**

<i>handler</i>	callback function for PatchStart event
----------------	--

**4.2.2.46 void PatchManager.removePatchWillPatchFileEventHandler ( event handler )**

unsubscribe from the PatchWillPatchFile event

**Parameters**

<i>handler</i>	callback function for PatchWillPatchFile event
----------------	--

**4.2.2.47 patchManifestItem PatchManager.repairPatchManifest ( string manifestUrl, array maskedFiles )**

add patch manifest repair to the manager

**Parameters**

<i>manifestUrl</i>	url to the manifest file
<i>maskedFiles</i>	files to mask for repair

**Returns**

patchManifestItem containing the id and patch manifest



4.2.2.48 void PatchManager.rootManifestById ( string *id* )

returns the root patch manifest by the assigned id or a child id

#### Parameters

<i>id</i>	root patch manifest id assigned by manager
-----------	--

4.2.2.49 void PatchManager.rootManifestIdById ( string *id* )

returns the root patch manifest id by the assigned id or a child id

#### Parameters

<i>id</i>	root patch manifest id assigned by manager
-----------	--

4.2.2.50 void PatchManager.verifyManifestById ( string *manifestId*, bool *quickScan* )

verify manifest by id

#### Parameters

<i>manifestId</i>	id for the manifest
<i>quickScan</i>	enable quick scanning during verify

The documentation for this class was generated from the following file:

- [patchmanager.js](#)

## 4.3 PatchManifest Class Reference

Download/process a patch manifest.

### Public Member Functions

- void [dependencyFail](#) ()
- string [expandString](#) (string expand)
- void [fail](#) ()
- int [getCurrentReleaseId](#) ()
- string [getExtraDataValue](#) (int fromId, int told, string key)
- int [getLastError](#) ()
- bool [getLocal](#) ()
- bool [getMaintenance](#) ()
- string [getName](#) ()
- string [getOutput](#) ()

- string [getReleaseName](#) (int id)
- int [getRequiredReleaseId](#) ()
- bool [getRequiresElevation](#) ()
- int [getRetryInterval](#) ()
- int [getSourceReleaseId](#) ()
- string [getSourceReleaseName](#) ()
- string [getSourceReleaseSHA1](#) ()
- string [getTargetDirectory](#) ()
- int [getTargetReleaseId](#) ()
- string [getTargetReleaseName](#) ()
- string [getTargetReleaseSHA1](#) ()
- int [getUpcomingReleaseId](#) ()
- string [getUrl](#) ()
- void [release](#) ()
- void [resume](#) ()
- void [setLocal](#) (bool value)
- void [setOutput](#) (string value)
- void [setRetryInterval](#) (int value)
- void [setSourceReleaseId](#) (int value)
- void [setSourceReleaseName](#) (string value)
- void [setSourceReleaseSHA1](#) (string value)
- void [setTargetDirectory](#) (string value)
- void [setTargetReleaseId](#) (int value)
- void [setTargetReleaseName](#) (string value)
- void [setTargetReleaseSHA1](#) (string value)
- void [setUrl](#) (string value)
- void [start](#) ()

### Public Attributes

- string [instanceId](#)
- event [onComplete](#)
- event [onDependency](#)
- event [onLoadComplete](#)
- event [onPatch](#)
- event [onStart](#)

### 4.3.1 Detailed Description

Download/process a patch manifest.

### 4.3.2 Member Function Documentation

#### 4.3.2.1 void PatchManifest.dependencyFail ( )

Special failure state because one of the dependencies failed

#### 4.3.2.2 string PatchManifest.expandString ( string *expand* )

This method expands the string with the current macros

##### Parameters

<i>expand</i>	Name of string to be expanded
---------------	-------------------------------

##### Returns

expanded string

#### 4.3.2.3 void PatchManifest.fail ( )

Fail the patch manifest after an action is completed

#### 4.3.2.4 int PatchManifest.getCurrentReleaseId ( )

gets the current release declared in the version file

##### Returns

current release declared in the version file

#### 4.3.2.5 string PatchManifest.getExtraDataValue ( int *fromId*, int *told*, string *key* )

Gets extra data value from the specified update path

##### Parameters

<i>fromId</i>	Release id
<i>told</i>	Release id
<i>key</i>	Key associated with value

##### Returns

Value of key stored in extra data for a release path (Empty string if not found)

#### 4.3.2.6 int PatchManifest.getLastError ( )

gets the last error encountered by the manifest

##### Returns

last error encountered by the manifest

#### 4.3.2.7 `bool PatchManifest.getLocal ( )`

gets the local state for the patch manifest

##### **Returns**

local state for the patch manifest

#### 4.3.2.8 `bool PatchManifest.getMaintenance ( )`

gets the manifest maintenance mode

##### **Returns**

manifest maintenance mode

#### 4.3.2.9 `string PatchManifest.getName ( )`

gets the name of the payload

##### **Returns**

name of the payload

#### 4.3.2.10 `string PatchManifest.getOutput ( )`

gets the location for patch manifest

##### **Returns**

location for patch manifest

#### 4.3.2.11 `string PatchManifest.getReleaseName ( int id )`

Gets the name for the specified release

##### **Parameters**

<i>id</i>	Release id
-----------	------------

##### **Returns**

Name of the specified release (Empty string if not found)

**4.3.2.12** `int PatchManifest.getRequiredReleaseId ( )`

gets the required release declared in the patch manifest

**Returns**

required release declared in the patch manifest

**4.3.2.13** `bool PatchManifest.getRequiresElevation ( )`

gets the manifest elevation requirement

**Returns**

manifest elevation requirement

**4.3.2.14** `int PatchManifest.getRetryInterval ( )`

gets the interval in seconds after completion the patch manifest will restart

**Returns**

interval in seconds after completion the patch manifest will restart

**4.3.2.15** `int PatchManifest.getSourceReleaseId ( )`

gets the id of the source release

**Returns**

id of the source release

**4.3.2.16** `string PatchManifest.getSourceReleaseName ( )`

gets the name of the source release

**Returns**

name of the source release

**4.3.2.17** `string PatchManifest.getSourceReleaseSHA1 ( )`

gets the sha1 of the source release

**Returns**

sha1 of the source release

#### 4.3.2.18 string PatchManifest.getTargetDirectory ( )

gets the directory patch will be installed in

##### Returns

directory patch will be installed in

#### 4.3.2.19 int PatchManifest.getTargetReleaseId ( )

gets the id of the target release

##### Returns

id of the target release

#### 4.3.2.20 string PatchManifest.getTargetReleaseName ( )

gets the name of the target release

##### Returns

name of the target release

#### 4.3.2.21 string PatchManifest.getTargetReleaseSHA1 ( )

gets the sha1 of the target release

##### Returns

sha1 of the target release

#### 4.3.2.22 int PatchManifest.getUpcomingReleaseId ( )

gets the upcoming release declared in the patch manifest

##### Returns

upcoming release declared in the patch manifest

#### 4.3.2.23 string PatchManifest.getUrl ( )

gets the url to metafile

##### Returns

url to metafile

**4.3.2.24 void PatchManifest.release ( )**

releases the object

**4.3.2.25 void PatchManifest.resume ( )**

Resume the patch manifest after an action is completed

**4.3.2.26 void PatchManifest.setLocal ( bool *value* )**

sets the local state for the patch manifest

**Parameters**

<i>value</i>	local state for the patch manifest
--------------	------------------------------------

**4.3.2.27 void PatchManifest.setOutput ( string *value* )**

sets the location for patch manifest

**Parameters**

<i>value</i>	location for patch manifest
--------------	-----------------------------

**4.3.2.28 void PatchManifest.setRetryInterval ( int *value* )**

sets the interval in seconds after completion the patch manifest will restart

**Parameters**

<i>value</i>	interval in seconds after completion the patch manifest will restart
--------------	--

**4.3.2.29 void PatchManifest.setSourceReleaseId ( int *value* )**

sets the id of the source release

**Parameters**

<i>value</i>	id of the source release
--------------	--------------------------

**4.3.2.30 void PatchManifest.setSourceReleaseName ( string *value* )**

sets the name of the source release

**Parameters**

<i>value</i>	name of the source release
--------------	----------------------------

**4.3.2.31 void PatchManifest.setSourceReleaseSHA1 ( string *value* )**

sets the sha1 of the source release

**Parameters**

<i>value</i>	sha1 of the source release
--------------	----------------------------

**4.3.2.32 void PatchManifest.setTargetDirectory ( string *value* )**

sets the directory patch will be installed in

**Parameters**

<i>value</i>	directory patch will be installed in
--------------	--------------------------------------

**4.3.2.33 void PatchManifest.setTargetReleaseId ( int *value* )**

sets the id of the target release

**Parameters**

<i>value</i>	id of the target release
--------------	--------------------------

**4.3.2.34 void PatchManifest.setTargetReleaseName ( string *value* )**

sets the name of the target release

**Parameters**

<i>value</i>	name of the target release
--------------	----------------------------

**4.3.2.35 void PatchManifest.setTargetReleaseSHA1 ( string *value* )**

sets the sha1 of the target release

**Parameters**

<i>value</i>	sha1 of the target release
--------------	----------------------------



**4.3.2.36 void PatchManifest.setUrl ( string *value* )**

sets the url to metafile

**Parameters**

<i>value</i>	url to metafile
--------------	-----------------

**4.3.2.37 void PatchManifest.start ( )**

Start processing the patch manifest

**4.3.3 Member Data Documentation****4.3.3.1 string PatchManifest.instanceId**

Instance id used to bind proxy object to native object

**4.3.3.2 event PatchManifest.onComplete**

Triggered when the manifest is complete

**Parameters**

<i>successful</i>	true if the manifest was successful, false otherwise
-------------------	--

**4.3.3.3 event PatchManifest.onDependency**

Triggered for each dependency in the patch manifest

**Parameters**

<i>url</i>	Url of the dependency
------------	-----------------------

**4.3.3.4 event PatchManifest.onLoadComplete**

Triggered when a patch manifest has loaded

**4.3.3.5 event PatchManifest.onPatch**

Triggered for each patch required

**Parameters**

<i>fromId</i>	Release id for the current patch
<i>told</i>	Release id for the patch destination

#### 4.3.3.6 event PatchManifest.onStart

Triggered when a manifest is started

The documentation for this class was generated from the following file:

- [patchmanifest.js](#)

## 4.4 PatchManifestError Class Reference

[Patch](#) manifest error constants and helper functions.

### Public Member Functions

- string [nameFromId](#) (int id)

### Public Attributes

- int [DEPENDENCYFAIL](#)
- int [DOWNLOADFAIL](#)
- int [INVALIDARGUMENT](#)
- int [MAINTENANCE](#)
- int [NONE](#)
- int [REQUIRESELEVATION](#)
- int [SOURCERELEASEUNDEFINED](#)
- int [TARGETRELEASEUNDEFINED](#)
- int [UNZIPFAILED](#)
- int [UPDATEPATHUNAVAILABLE](#)
- int [USERFAIL](#)
- int [VERSIONFAIL](#)
- int [XMLPARSEFAIL](#)

#### 4.4.1 Detailed Description

[Patch](#) manifest error constants and helper functions.

#### 4.4.2 Member Function Documentation

##### 4.4.2.1 string PatchManifestError.nameFromId ( int id )

converts a patch manifest error to a string

### Parameters

<i>id</i>	patch manifest error
-----------	----------------------

**Returns**

stringified name of file error.

**4.4.3 Member Data Documentation****4.4.3.1 int PatchManifestError.DEPENDENCYFAIL**

Dependency failed

**4.4.3.2 int PatchManifestError.DOWNLOADFAIL**

Download failed

**4.4.3.3 int PatchManifestError.INVALIDARGUMENT**

Invalid argument

**4.4.3.4 int PatchManifestError.MAINTENANCE**

Maintenance flag turned on

**4.4.3.5 int PatchManifestError.NONE**

No error

**4.4.3.6 int PatchManifestError.REQUIRESELEVATION**

Elevation required

**4.4.3.7 int PatchManifestError.SOURCERELEASEUNDEFINED**

Source release undefined

**4.4.3.8 int PatchManifestError.TARGETRELEASEUNDEFINED**

Target release undefined

**4.4.3.9 int PatchManifestError.UNZIPFAILED**

Unzip failed

#### 4.4.3.10 int PatchManifestError.UPDATEPATHUNAVAILABLE

Update path unavailable

#### 4.4.3.11 int PatchManifestError.USERFAIL

Fail method was called

#### 4.4.3.12 int PatchManifestError.VERSIONFAIL

Version check failed

#### 4.4.3.13 int PatchManifestError.XMLPARSEFAIL

Parsing failed

The documentation for this class was generated from the following file:

- [patchmanifesterror.js](#)

## 4.5 PatchManifestState Class Reference

[Patch](#) manifest state constants and helper functions.

### Public Member Functions

- string [nameFromId](#) (int id)

### Public Attributes

- int [COMPLETE](#)
- int [DEPENDENCY](#)
- int [DOWNLOAD](#)
- int [GETMANIFEST](#)
- int [PATCH](#)

#### 4.5.1 Detailed Description

[Patch](#) manifest state constants and helper functions.

### 4.5.2 Member Function Documentation

#### 4.5.2.1 string PatchManifestState.nameFromId ( int id )

converts a patch manifest state to a string

##### Parameters

<i>id</i>	patch manifest state
-----------	----------------------

##### Returns

stringified name of file error.

### 4.5.3 Member Data Documentation

#### 4.5.3.1 int PatchManifestState.COMPLETE

[Patch](#) Complete

#### 4.5.3.2 int PatchManifestState.DEPENDENCY

[Patch](#) dependency required

#### 4.5.3.3 int PatchManifestState.DOWNLOAD

[Patch](#) downloading

#### 4.5.3.4 int PatchManifestState.GETMANIFEST

Get manifest state

#### 4.5.3.5 int PatchManifestState.PATCH

Applying patch

The documentation for this class was generated from the following file:

- [patchmanifeststate.js](#)

## 4.6 PatchVersion Class Reference

Enumerate a version file.

### Public Member Functions

- string [getFileName](#) ()
- string [getSha1](#) ()
- void [release](#) ()
- void [resume](#) ()
- void [setFileName](#) (string value)
- void [start](#) ()

### Public Attributes

- string [instanceId](#)
- event [onComplete](#)
- event [onFile](#)
- event [onStart](#)

#### 4.6.1 Detailed Description

Enumerate a version file.

#### 4.6.2 Member Function Documentation

##### 4.6.2.1 string PatchVersion.getFileName ( )

gets the file name of the version file

##### Returns

file name of the version file

##### 4.6.2.2 string PatchVersion.getSha1 ( )

gets the SHA1 of the version file

##### Returns

SHA1 of the version file

##### 4.6.2.3 void PatchVersion.release ( )

releases the object

##### 4.6.2.4 void PatchVersion.resume ( )

Resume the file enumeration

#### 4.6.2.5 void PatchVersion.setFileName ( string *value* )

sets the file name of the version file

##### Parameters

<i>value</i>	file name of the version file
--------------	-------------------------------

#### 4.6.2.6 void PatchVersion.start ( )

Starts the patch version enumeration

### 4.6.3 Member Data Documentation

#### 4.6.3.1 string PatchVersion.instanceId

Instance id used to bind proxy object to native object

#### 4.6.3.2 event PatchVersion.onComplete

Triggered when the version file enumeration is complete

##### Parameters

<i>successful</i>	true if the download was successful, false otherwise
-------------------	--

#### 4.6.3.3 event PatchVersion.onFile

Triggered when the version file enumeration has detected a file

##### Parameters

<i>name</i>	Name of the file
<i>lastModified</i>	Last modified date of the file
<i>size</i>	Size of the file
<i>sha1</i>	SHA1 of the file

#### 4.6.3.4 event PatchVersion.onStart

Triggered when version file enumeration is started

The documentation for this class was generated from the following file:

- [patchversion.js](#)

## 4.7 PatchVersionVerify Class Reference

Verify a version file.

### Public Member Functions

- string [getFileName](#) ()
- bool [getQuickScan](#) ()
- void [release](#) ()
- void [setFileName](#) (string value)
- void [setQuickScan](#) (bool value)
- void [start](#) ()

### Public Attributes

- string [instanceId](#)
- event [onComplete](#)
- event [onCorruptFile](#)
- event [onProgress](#)
- event [onStart](#)

### 4.7.1 Detailed Description

Verify a version file.

### 4.7.2 Member Function Documentation

#### 4.7.2.1 string PatchVersionVerify.getFileName ( )

gets the file name of the version file

#### Returns

file name of the version file

#### 4.7.2.2 bool PatchVersionVerify.getQuickScan ( )

gets the quickScan state (enables a quicker but less safe verification)

#### Returns

quickScan state (enables a quicker but less safe verification)



#### 4.7.2.3 void PatchVersionVerify.release ( )

releases the object

#### 4.7.2.4 void PatchVersionVerify.setFileName ( string *value* )

sets the file name of the version file

##### Parameters

<i>value</i>	file name of the version file
--------------	-------------------------------

#### 4.7.2.5 void PatchVersionVerify.setQuickScan ( bool *value* )

sets the quickScan state (enables a quicker but less safe verification)

##### Parameters

<i>value</i>	quickScan state (enables a quicker but less safe verification)
--------------	--

#### 4.7.2.6 void PatchVersionVerify.start ( )

Starts the patch manifest

### 4.7.3 Member Data Documentation

#### 4.7.3.1 string PatchVersionVerify.instanceId

Instance id used to bind proxy object to native object

#### 4.7.3.2 event PatchVersionVerify.onComplete

Triggered when the verification is complete

##### Parameters

<i>successful</i>	true if the download was successful, false otherwise
-------------------	--

#### 4.7.3.3 event PatchVersionVerify.onCorruptFile

Triggered when the verification has detected a corrupt file

##### Parameters

<i>name</i>	Name of the corrupt file
-------------	--------------------------

#### 4.7.3.4 event `PatchVersionVerify.onProgress`

Triggered when the verification has made progress

##### Parameters

<i>percent</i>	(-1.0: Still calculating) (0.0 to 1.0: Percent completed)
----------------	---

#### 4.7.3.5 event `PatchVersionVerify.onStart`

Triggered when verification is started

The documentation for this class was generated from the following file:

- [patchversionverify.js](#)

## Chapter 5

# File Documentation

### 5.1 patch.js File Reference

File containing [Patch](#) class and creation function.

#### Classes

- class [Patch](#)  
*Extract a patch.*

#### Functions

- void [createPatch](#) ()

#### 5.1.1 Detailed Description

File containing [Patch](#) class and creation function.

#### 5.1.2 Function Documentation

##### 5.1.2.1 void createPatch ( )

Create instance of patch

### 5.2 patchmanager.js File Reference

File containing patch manager.

## Classes

- class [PatchManager](#)  
*Manages patches and provides an integrated event structure.*

## Variables

- [PatchManager](#) `patchManager`

### 5.2.1 Detailed Description

File containing patch manager.

### 5.2.2 Variable Documentation

#### 5.2.2.1 [PatchManager](#) `patchManager`

precreated global instance of the patch manager

## 5.3 [patchmanifest.js](#) File Reference

File containing [PatchManifest](#) class and creation function.

## Classes

- class [PatchManifest](#)  
*Download/process a patch manifest.*

## Functions

- void [createPatchManifest](#) ()

### 5.3.1 Detailed Description

File containing [PatchManifest](#) class and creation function.

### 5.3.2 Function Documentation

#### 5.3.2.1 void [createPatchManifest](#) ( )

Create instance of `patchManifest`

## 5.4 patchmanifesterror.js File Reference

File containing patch manifest error constants and helper functions.

### Classes

- class [PatchManifestError](#)  
*Patch manifest error constants and helper functions.*

### Variables

- [PatchManifestError](#) [patchManifestError](#)

#### 5.4.1 Detailed Description

File containing patch manifest error constants and helper functions.

#### 5.4.2 Variable Documentation

##### 5.4.2.1 PatchManifestError patchManifestError

precreated global instance of [PatchManifestError](#)

## 5.5 patchmanifeststate.js File Reference

File containing patch manifest state constants and helper functions.

### Classes

- class [PatchManifestState](#)  
*Patch manifest state constants and helper functions.*

### Variables

- [PatchManifestState](#) [patchManifestState](#)

#### 5.5.1 Detailed Description

File containing patch manifest state constants and helper functions.

## 5.5.2 Variable Documentation

### 5.5.2.1 PatchManifestState patchManifestState

precreated global instance of [PatchManifestState](#)

## 5.6 patchversion.js File Reference

File containing [PatchVersion](#) class and creation function.

### Classes

- class [PatchVersion](#)  
*Enumerate a version file.*

### Functions

- void [createPatchVersion](#) ()

### 5.6.1 Detailed Description

File containing [PatchVersion](#) class and creation function.

### 5.6.2 Function Documentation

#### 5.6.2.1 void createPatchVersion ( )

Create instance of patchVersion

## 5.7 patchversionverify.js File Reference

File containing [PatchVersionVerify](#) class and creation function.

### Classes

- class [PatchVersionVerify](#)  
*Verify a version file.*

### Functions

- void [createPatchVersionVerify](#) ()

### 5.7.1 Detailed Description

File containing [PatchVersionVerify](#) class and creation function.

### 5.7.2 Function Documentation

#### 5.7.2.1 void createPatchVersionVerify ( )

Create instance of patchVersionVerify

# Index

- addManifestAddedEventHandler
  - PatchManager, [14](#)
- addManifestCompleteEventHandler
  - PatchManager, [14](#)
- addManifestDependencyEventHandler
  - PatchManager, [14](#)
- addManifestRemovedEventHandler
  - PatchManager, [15](#)
- addManifestRequiredCompleteEventHandler
  - PatchManager, [15](#)
- addManifestStartEventHandler
  - PatchManager, [15](#)
- addManifestStateChangeEventHandler
  - PatchManager, [15](#)
- addManifestVerifyCompleteEventHandler
  - PatchManager, [15](#)
- addManifestVerifyProgressEventHandler
  - PatchManager, [15](#)
- addManifestVerifyStartEventHandler
  - PatchManager, [16](#)
- addPatchCompleteEventHandler
  - PatchManager, [16](#)
- addPatchCorruptDestFileEventHandler
  - PatchManager, [16](#)
- addPatchCorruptSourceFileEventHandler
  - PatchManager, [16](#)
- addPatchDidPatchFileEventHandler
  - PatchManager, [16](#)
- addPatchDownloadErrorEventHandler
  - PatchManager, [17](#)
- addPatchFileErrorEventHandler
  - PatchManager, [17](#)
- addPatchManifest
  - PatchManager, [17](#)
- addPatchPrepareCompleteEventHandler
  - PatchManager, [17](#)
- addPatchPreparePatchFileEventHandler
  - PatchManager, [17](#)
- addPatchProgressEventHandler
  - PatchManager, [18](#)
- addPatchStartEventHandler
  - PatchManager, [18](#)
- addPatchWillPatchFileEventHandler
  - PatchManager, [18](#)
- COMPLETE
  - PatchManifestState, [35](#)
- createPatch
  - patch.js, [41](#)
- createPatchManifest
  - patchmanifest.js, [42](#)
- createPatchVersion
  - patchversion.js, [44](#)
- createPatchVersionVerify
  - patchversionverify.js, [45](#)
- DEPENDENCY
  - PatchManifestState, [35](#)
- DEPENDENCYFAIL
  - PatchManifestError, [33](#)
- dependencyFail
  - PatchManifest, [24](#)
- DOWNLOAD
  - PatchManifestState, [35](#)
- DOWNLOADFAIL
  - PatchManifestError, [33](#)
- expandString
  - PatchManifest, [24](#)
- fail
  - Patch, [8](#)
  - PatchManifest, [25](#)
- getBytesLeft
  - Patch, [8](#)
- getCurrentItemCompressedSize
  - Patch, [8](#)
- getCurrentItemDiskFileName
  - Patch, [8](#)
- getCurrentItemDiskOffset
  - Patch, [8](#)
- getCurrentReleaseId



- PatchManifest, 25
- getExtraDataValue
  - PatchManifest, 25
- getFileName
  - Patch, 9
  - PatchVersion, 36
  - PatchVersionVerify, 38
- getLastError
  - PatchManifest, 25
- getLocal
  - PatchManifest, 25
- getMaintenance
  - PatchManifest, 26
- GETMANIFEST
  - PatchManifestState, 35
- getName
  - PatchManifest, 26
- getOutput
  - PatchManifest, 26
- getOutputDirectory
  - Patch, 9
- getQuickScan
  - PatchVersionVerify, 38
- getReleaseName
  - PatchManifest, 26
- getRemainingTime
  - Patch, 9
- getRequiredReleaseId
  - PatchManifest, 26
- getRequiresElevation
  - PatchManifest, 27
- getRetryInterval
  - PatchManifest, 27
- getSha1
  - PatchVersion, 36
- getSourceReleaseId
  - PatchManifest, 27
- getSourceReleaseName
  - PatchManifest, 27
- getSourceReleaseSHA1
  - PatchManifest, 27
- getStalled
  - Patch, 9
- getTargetDirectory
  - PatchManifest, 27
- getTargetReleaseId
  - PatchManifest, 28
- getTargetReleaseName
  - PatchManifest, 28
- getTargetReleaseSHA1
  - PatchManifest, 28
- getTotalBytes
  - Patch, 9
- getUpcomingReleaseId
  - PatchManifest, 28
- getUrl
  - PatchManifest, 28
- getWriteRates
  - Patch, 9
- instanceId
  - Patch, 10
  - PatchManifest, 31
  - PatchVersion, 37
  - PatchVersionVerify, 39
- INVALIDARGUMENT
  - PatchManifestError, 33
- MAINTENANCE
  - PatchManifestError, 33
- manifestById
  - PatchManager, 18
- nameFromId
  - PatchManifestError, 32
  - PatchManifestState, 35
- NONE
  - PatchManifestError, 33
- onComplete
  - Patch, 11
  - PatchManifest, 31
  - PatchVersion, 37
  - PatchVersionVerify, 39
- onCorruptDestFile
  - Patch, 11
- onCorruptFile
  - PatchVersionVerify, 39
- onCorruptSourceFile
  - Patch, 11
- onDependency
  - PatchManifest, 31
- onDidPatchFile
  - Patch, 11
- onFile
  - PatchVersion, 37
- onFileError
  - Patch, 11
- onLoadComplete
  - PatchManifest, 31

- onPatch
  - PatchManifest, 31
- onPrepareComplete
  - Patch, 12
- onPreparePatchFile
  - Patch, 12
- onProgress
  - Patch, 12
  - PatchVersionVerify, 39
- onStart
  - Patch, 12
  - PatchManifest, 32
  - PatchVersion, 37
  - PatchVersionVerify, 40
- onWillPatchFile
  - Patch, 12
- PATCH
  - PatchManifestState, 35
- Patch, 7
  - fail, 8
  - getBytesLeft, 8
  - getCurrentItemCompressedSize, 8
  - getCurrentItemDiskFileName, 8
  - getCurrentItemDiskOffset, 8
  - getFileName, 9
  - getOutputDirectory, 9
  - getRemainingTime, 9
  - getStalled, 9
  - getTotalBytes, 9
  - getWriteRates, 9
  - instanceId, 10
  - onComplete, 11
  - onCorruptDestFile, 11
  - onCorruptSourceFile, 11
  - onDidPatchFile, 11
  - onFileError, 11
  - onPrepareComplete, 12
  - onPreparePatchFile, 12
  - onProgress, 12
  - onStart, 12
  - onWillPatchFile, 12
  - release, 10
  - resume, 10
  - setFileName, 10
  - setOutputDirectory, 10
  - skip, 10
  - start, 10
- patch.js, 41
  - createPatch, 41
- PatchManager, 13
  - addManifestAddedEventHandler, 14
  - addManifestCompleteEventHandler, 14
  - addManifestDependencyEventHandler, 14
  - addManifestRemovedEventHandler, 15
  - addManifestRequiredCompleteEventHandler, 15
  - addManifestStartEventHandler, 15
  - addManifestStateChangeEventHandler, 15
  - addManifestVerifyCompleteEventHandler, 15
  - addManifestVerifyProgressEventHandler, 15
  - addManifestVerifyStartEventHandler, 16
  - addPatchCompleteEventHandler, 16
  - addPatchCorruptDestFileEventHandler, 16
  - addPatchCorruptSourceFileEventHandler, 16
  - addPatchDidPatchFileEventHandler, 16
  - addPatchDownloadErrorEventHandler, 17
  - addPatchFileErrorEventHandler, 17
  - addPatchManifest, 17
  - addPatchPrepareCompleteEventHandler, 17
  - addPatchPreparePatchFileEventHandler, 17
  - addPatchProgressEventHandler, 18
  - addPatchStartEventHandler, 18
  - addPatchWillPatchFileEventHandler, 18
  - manifestById, 18
  - release, 18
  - removeManifestAddedEventHandler, 18
  - removeManifestCompleteEventHandler, 18
  - removeManifestDependencyEventHandler, 19
  - removeManifestRemovedEventHandler, 19
  - removeManifestRequiredCompleteEventHandler, 19
  - removeManifestStartEventHandler, 19
  - removeManifestStateChangeEventHandler, 19
  - removeManifestVerifyCompleteEventHandler, 20

- removeManifestVerifyProgressEventHandler, [getTargetReleaseId, 28](#)  
[20](#)
- removeManifestVerifyStartEventHandler, [getTargetReleaseName, 28](#)  
[20](#) [getTargetReleaseSHA1, 28](#)  
[getUpcomingReleaseId, 28](#)
- removePatchCompleteEventHandler, [20](#) [getUrl, 28](#)
- removePatchCorruptDestFileEventHandler, [instanceId, 31](#)  
[20](#) [onComplete, 31](#)
- removePatchCorruptSourceFileEventHandler, [onDependency, 31](#)  
[20](#) [onLoadComplete, 31](#)
- removePatchDidPatchFileEventHandler, [onPatch, 31](#)  
[21](#) [onStart, 32](#)
- removePatchDownloadErrorEventHandler, [release, 28](#)  
[21](#) [resume, 29](#)
- removePatchFileErrorEventHandler, [21](#) [setLocal, 29](#)
- removePatchManifest, [21](#) [setOutput, 29](#)
- removePatchPrepareCompleteEventHandlers, [setRetryInterval, 29](#)  
[21](#) [setSourceReleaseId, 29](#)
- removePatchPreparePatchFileEventHandler, [setSourceReleaseName, 29](#)  
[22](#) [setSourceReleaseSHA1, 30](#)
- removePatchProgressEventHandler, [22](#) [setTargetDirectory, 30](#)
- removePatchStartEventHandler, [22](#) [setTargetReleaseId, 30](#)
- removePatchWillPatchFileEventHandler, [setTargetReleaseName, 30](#)  
[22](#) [setTargetReleaseSHA1, 30](#)  
[setUrl, 30](#)  
[start, 31](#)
- repairPatchManifest, [22](#)
- rootManifestById, [22](#)
- rootManifestByIdByld, [23](#)
- verifyManifestByld, [23](#)
- patchManager
  - patchmanager.js, [42](#)
- patchmanager.js, [41](#)
  - patchManager, [42](#)
- PatchManifest, [23](#)
  - dependencyFail, [24](#)
  - expandString, [24](#)
  - fail, [25](#)
  - getCurrentReleaseId, [25](#)
  - getExtraDataValue, [25](#)
  - getLastError, [25](#)
  - getLocal, [25](#)
  - getMaintenance, [26](#)
  - getName, [26](#)
  - getOutput, [26](#)
  - getReleaseName, [26](#)
  - getRequiredReleaseId, [26](#)
  - getRequiresElevation, [27](#)
  - getRetryInterval, [27](#)
  - getSourceReleaseId, [27](#)
  - getSourceReleaseName, [27](#)
  - getSourceReleaseSHA1, [27](#)
  - getTargetDirectory, [27](#)
- patchmanifest.js, [42](#)
  - createPatchManifest, [42](#)
- PatchManifestError, [32](#)
  - DEPENDENCYFAIL, [33](#)
  - DOWNLOADFAIL, [33](#)
  - INVALIDARGUMENT, [33](#)
  - MAINTENANCE, [33](#)
  - nameFromId, [32](#)
  - NONE, [33](#)
  - REQUIRESELEVATION, [33](#)
  - SOURCERELEASEUNDEFINED, [33](#)
  - TARGETRELEASEUNDEFINED, [33](#)
  - UNZIPFAILED, [33](#)
  - UPDATEPATHUNAVAILABLE, [33](#)
  - USERFAIL, [34](#)
  - VERSIONFAIL, [34](#)
  - XMLPARSEFAIL, [34](#)
- patchManifestError
  - patchmanifesterror.js, [43](#)
- patchmanifesterror.js, [43](#)
  - patchManifestError, [43](#)
- PatchManifestState, [34](#)
  - COMPLETE, [35](#)
  - DEPENDENCY, [35](#)
  - DOWNLOAD, [35](#)

- GETMANIFEST, 35
- nameFromId, 35
- PATCH, 35
- patchManifestState
  - patchmanifeststate.js, 44
- patchmanifeststate.js, 43
- patchManifestState, 44
- PatchVersion, 35
  - getFileName, 36
  - getSha1, 36
  - instanceId, 37
  - onComplete, 37
  - onFile, 37
  - onStart, 37
  - release, 36
  - resume, 36
  - setFileName, 36
  - start, 37
- patchversion.js, 44
  - createPatchVersion, 44
- PatchVersionVerify, 38
  - getFileName, 38
  - getQuickScan, 38
  - instanceId, 39
  - onComplete, 39
  - onCorruptFile, 39
  - onProgress, 39
  - onStart, 40
  - release, 38
  - setFileName, 39
  - setQuickScan, 39
  - start, 39
- patchversionverify.js, 44
  - createPatchVersionVerify, 45
- release
  - Patch, 10
  - PatchManager, 18
  - PatchManifest, 28
  - PatchVersion, 36
  - PatchVersionVerify, 38
- removeManifestAddedEventHandler
  - PatchManager, 18
- removeManifestCompleteEventHandler
  - PatchManager, 18
- removeManifestDependencyEventHandler
  - PatchManager, 19
- removeManifestRemovedEventHandler
  - PatchManager, 19
- removeManifestRequiredCompleteEventHandler
  - Patch, 10
  - PatchManager, 19
- removeManifestStartEventHandler
  - PatchManager, 19
- removeManifestStateChangeEventHandler
  - PatchManager, 19
- removeManifestVerifyCompleteEventHandler
  - PatchManager, 20
- removeManifestVerifyProgressEventHandler
  - PatchManager, 20
- removeManifestVerifyStartEventHandler
  - PatchManager, 20
- removePatchCompleteEventHandler
  - PatchManager, 20
- removePatchCorruptDestFileEventHandler
  - PatchManager, 20
- removePatchCorruptSourceFileEventHandler
  - PatchManager, 20
- removePatchDidPatchFileEventHandler
  - PatchManager, 21
- removePatchDownloadErrorEventHandler
  - PatchManager, 21
- removePatchFileErrorEventHandler
  - PatchManager, 21
- removePatchManifest
  - PatchManager, 21
- removePatchPrepareCompleteEventHandler
  - PatchManager, 21
- removePatchPreparePatchFileEventHandler
  - PatchManager, 22
- removePatchProgressEventHandler
  - PatchManager, 22
- removePatchStartEventHandler
  - PatchManager, 22
- removePatchWillPatchFileEventHandler
  - PatchManager, 22
- repairPatchManifest
  - PatchManager, 22
- REQUIRESELEVATION
  - PatchManifestError, 33
- resume
  - Patch, 10
  - PatchManifest, 29
  - PatchVersion, 36
- rootManifestById
  - PatchManager, 22
- rootManifestById
  - PatchManager, 23
- setFileName
  - Patch, 10

- PatchVersion, [36](#)
  - PatchVersionVerify, [39](#)
- setLocal
  - PatchManifest, [29](#)
- setOutput
  - PatchManifest, [29](#)
- setOutputDirectory
  - Patch, [10](#)
- setQuickScan
  - PatchVersionVerify, [39](#)
- setRetryInterval
  - PatchManifest, [29](#)
- setSourceReleaseId
  - PatchManifest, [29](#)
- setSourceReleaseName
  - PatchManifest, [29](#)
- setSourceReleaseSHA1
  - PatchManifest, [30](#)
- setTargetDirectory
  - PatchManifest, [30](#)
- setTargetReleaseId
  - PatchManifest, [30](#)
- setTargetReleaseName
  - PatchManifest, [30](#)
- setTargetReleaseSHA1
  - PatchManifest, [30](#)
- setUrl
  - PatchManifest, [30](#)
- skip
  - Patch, [10](#)
- SOURCERELEASEUNDEFINED
  - PatchManifestError, [33](#)
- start
  - Patch, [10](#)
  - PatchManifest, [31](#)
  - PatchVersion, [37](#)
  - PatchVersionVerify, [39](#)
- TARGETRELEASEUNDEFINED
  - PatchManifestError, [33](#)
- UNZIPFAILED
  - PatchManifestError, [33](#)
- UPDATEPATHUNAVAILABLE
  - PatchManifestError, [33](#)
- USERFAIL
  - PatchManifestError, [34](#)
- verifyManifestById
  - PatchManager, [23](#)
- VERSIONFAIL
  - PatchManifestError, [34](#)
- XMLPARSEFAIL
  - PatchManifestError, [34](#)