

# Patcher Javascript API

3.1.3.1

Generated by Doxygen 1.7.4

Thu Oct 13 2011 14:45:19



# 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	FileError Class Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Member Function Documentation . . . . .	7
4.1.2.1	nameFromId . . . . .	7
4.1.3	Member Data Documentation . . . . .	8
4.1.3.1	DISKSPACE . . . . .	8
4.1.3.2	LOCKED . . . . .	8
4.1.3.3	NONE . . . . .	8
4.1.3.4	UNKNOWN . . . . .	8
4.2	Patch Class Reference . . . . .	8
4.2.1	Detailed Description . . . . .	9
4.2.2	Member Function Documentation . . . . .	9
4.2.2.1	fail . . . . .	9
4.2.2.2	getBytesLeft . . . . .	9
4.2.2.3	getCurrentItemCompressedSize . . . . .	9
4.2.2.4	getCurrentItemDiskFileName . . . . .	10

4.2.2.5	<a href="#">getCurrentItemDiskOffset</a>	10
4.2.2.6	<a href="#">getFileName</a>	10
4.2.2.7	<a href="#">getOutputDirectory</a>	10
4.2.2.8	<a href="#">getRemainingTime</a>	10
4.2.2.9	<a href="#">getStalled</a>	10
4.2.2.10	<a href="#">getTotalBytes</a>	11
4.2.2.11	<a href="#">getWriteRates</a>	11
4.2.2.12	<a href="#">release</a>	11
4.2.2.13	<a href="#">resume</a>	11
4.2.2.14	<a href="#">setFileName</a>	11
4.2.2.15	<a href="#">setOutputDirectory</a>	11
4.2.2.16	<a href="#">skip</a>	11
4.2.2.17	<a href="#">start</a>	12
4.2.3	<a href="#">Member Data Documentation</a>	12
4.2.3.1	<a href="#">instanceId</a>	12
4.2.3.2	<a href="#">onComplete</a>	12
4.2.3.3	<a href="#">onCorruptDestFile</a>	12
4.2.3.4	<a href="#">onCorruptSourceFile</a>	12
4.2.3.5	<a href="#">onDidPatchFile</a>	12
4.2.3.6	<a href="#">onFileError</a>	13
4.2.3.7	<a href="#">onPrepareComplete</a>	13
4.2.3.8	<a href="#">onPreparePatchFile</a>	13
4.2.3.9	<a href="#">onProgress</a>	13
4.2.3.10	<a href="#">onStart</a>	13
4.2.3.11	<a href="#">onWillPatchFile</a>	13
4.3	<a href="#">PatchManager Class Reference</a>	14
4.3.1	<a href="#">Detailed Description</a>	15
4.3.2	<a href="#">Member Function Documentation</a>	15
4.3.2.1	<a href="#">addManifestAddedEventHandler</a>	15
4.3.2.2	<a href="#">addManifestCompleteEventHandler</a>	15
4.3.2.3	<a href="#">addManifestDependencyEventHandler</a>	15
4.3.2.4	<a href="#">addManifestRemovedEventHandler</a>	16
4.3.2.5	<a href="#">addManifestRequiredCompleteEventHandler</a>	16
4.3.2.6	<a href="#">addManifestStartEventHandler</a>	16

4.3.2.7	<a href="#">addManifestStateChangeEventHandler</a>	16
4.3.2.8	<a href="#">addManifestVerifyCompleteEventHandler</a>	16
4.3.2.9	<a href="#">addManifestVerifyProgressEventHandler</a>	17
4.3.2.10	<a href="#">addManifestVerifyStartEventHandler</a>	17
4.3.2.11	<a href="#">addPatchCompleteEventHandler</a>	17
4.3.2.12	<a href="#">addPatchCorruptDestFileEventHandler</a>	17
4.3.2.13	<a href="#">addPatchCorruptSourceFileEventHandler</a>	17
4.3.2.14	<a href="#">addPatchDidPatchFileEventHandler</a>	17
4.3.2.15	<a href="#">addPatchDownloadErrorEventHandler</a>	18
4.3.2.16	<a href="#">addPatchFileErrorEventHandler</a>	18
4.3.2.17	<a href="#">addPatchManifest</a>	18
4.3.2.18	<a href="#">addPatchPrepareCompleteEventHandler</a>	18
4.3.2.19	<a href="#">addPatchPreparePatchFileEventHandler</a>	18
4.3.2.20	<a href="#">addPatchProgressEventHandler</a>	19
4.3.2.21	<a href="#">addPatchStartEventHandler</a>	19
4.3.2.22	<a href="#">addPatchWillPatchFileEventHandler</a>	19
4.3.2.23	<a href="#">manifestById</a>	19
4.3.2.24	<a href="#">release</a>	19
4.3.2.25	<a href="#">removeManifestAddedEventHandler</a>	19
4.3.2.26	<a href="#">removeManifestCompleteEventHandler</a>	20
4.3.2.27	<a href="#">removeManifestDependencyEventHandler</a>	20
4.3.2.28	<a href="#">removeManifestRemovedEventHandler</a>	20
4.3.2.29	<a href="#">removeManifestRequiredCompleteEventHandler</a>	20
4.3.2.30	<a href="#">removeManifestStartEventHandler</a>	20
4.3.2.31	<a href="#">removeManifestStateChangeEventHandler</a>	20
4.3.2.32	<a href="#">removeManifestVerifyCompleteEventHandler</a>	21
4.3.2.33	<a href="#">removeManifestVerifyProgressEventHandler</a>	21
4.3.2.34	<a href="#">removeManifestVerifyStartEventHandler</a>	21
4.3.2.35	<a href="#">removePatchCompleteEventHandler</a>	21
4.3.2.36	<a href="#">removePatchCorruptDestFileEventHandler</a>	21
4.3.2.37	<a href="#">removePatchCorruptSourceFileEventHandler</a>	21
4.3.2.38	<a href="#">removePatchDidPatchFileEventHandler</a>	22
4.3.2.39	<a href="#">removePatchDownloadErrorEventHandler</a>	22
4.3.2.40	<a href="#">removePatchFileErrorEventHandler</a>	22

4.3.2.41	<a href="#">removePatchManifest</a>	22
4.3.2.42	<a href="#">removePatchPrepareCompleteEventHandler</a>	22
4.3.2.43	<a href="#">removePatchPreparePatchFileEventHandler</a>	23
4.3.2.44	<a href="#">removePatchProgressEventHandler</a>	23
4.3.2.45	<a href="#">removePatchStartEventHandler</a>	23
4.3.2.46	<a href="#">removePatchWillPatchFileEventHandler</a>	23
4.3.2.47	<a href="#">repairPatchManifest</a>	23
4.3.2.48	<a href="#">rootManifestById</a>	24
4.3.2.49	<a href="#">rootManifestById</a>	24
4.3.2.50	<a href="#">verifyManifestById</a>	24
4.4	<a href="#">PatchManifest Class Reference</a>	24
4.4.1	<a href="#">Detailed Description</a>	25
4.4.2	<a href="#">Member Function Documentation</a>	25
4.4.2.1	<a href="#">dependencyFail</a>	25
4.4.2.2	<a href="#">expandString</a>	26
4.4.2.3	<a href="#">fail</a>	26
4.4.2.4	<a href="#">getCurrentReleaseId</a>	26
4.4.2.5	<a href="#">getExtraDataValue</a>	26
4.4.2.6	<a href="#">getLastError</a>	26
4.4.2.7	<a href="#">getLocal</a>	27
4.4.2.8	<a href="#">getMaintenance</a>	27
4.4.2.9	<a href="#">getName</a>	27
4.4.2.10	<a href="#">getOutput</a>	27
4.4.2.11	<a href="#">getReleaseName</a>	27
4.4.2.12	<a href="#">getRequiredReleaseId</a>	28
4.4.2.13	<a href="#">getRequiresElevation</a>	28
4.4.2.14	<a href="#">getRetryInterval</a>	28
4.4.2.15	<a href="#">getSourceReleaseId</a>	28
4.4.2.16	<a href="#">getSourceReleaseName</a>	28
4.4.2.17	<a href="#">getSourceReleaseSHA1</a>	28
4.4.2.18	<a href="#">getTargetDirectory</a>	29
4.4.2.19	<a href="#">getTargetReleaseId</a>	29
4.4.2.20	<a href="#">getTargetReleaseName</a>	29
4.4.2.21	<a href="#">getTargetReleaseSHA1</a>	29

4.4.2.22	<a href="#">getUpcomingReleaseId</a>	29
4.4.2.23	<a href="#">getUrl</a>	29
4.4.2.24	<a href="#">release</a>	30
4.4.2.25	<a href="#">resume</a>	30
4.4.2.26	<a href="#">setLocal</a>	30
4.4.2.27	<a href="#">setOutput</a>	30
4.4.2.28	<a href="#">setRetryInterval</a>	30
4.4.2.29	<a href="#">setSourceReleaseId</a>	30
4.4.2.30	<a href="#">setSourceReleaseName</a>	30
4.4.2.31	<a href="#">setSourceReleaseSHA1</a>	31
4.4.2.32	<a href="#">setTargetDirectory</a>	31
4.4.2.33	<a href="#">setTargetReleaseId</a>	31
4.4.2.34	<a href="#">setTargetReleaseName</a>	31
4.4.2.35	<a href="#">setTargetReleaseSHA1</a>	31
4.4.2.36	<a href="#">setUrl</a>	32
4.4.2.37	<a href="#">start</a>	32
4.4.3	<a href="#">Member Data Documentation</a>	32
4.4.3.1	<a href="#">instanceId</a>	32
4.4.3.2	<a href="#">onComplete</a>	32
4.4.3.3	<a href="#">onDependency</a>	32
4.4.3.4	<a href="#">onLoadComplete</a>	32
4.4.3.5	<a href="#">onPatch</a>	32
4.4.3.6	<a href="#">onStart</a>	33
4.5	<a href="#">PatchManifestError Class Reference</a>	33
4.5.1	<a href="#">Detailed Description</a>	33
4.5.2	<a href="#">Member Function Documentation</a>	33
4.5.2.1	<a href="#">nameFromId</a>	33
4.5.3	<a href="#">Member Data Documentation</a>	34
4.5.3.1	<a href="#">DEPENDENCYFAIL</a>	34
4.5.3.2	<a href="#">DOWNLOADFAIL</a>	34
4.5.3.3	<a href="#">INVALIDARGUMENT</a>	34
4.5.3.4	<a href="#">MAINTENANCE</a>	34
4.5.3.5	<a href="#">NONE</a>	34
4.5.3.6	<a href="#">REQUIRESELEVATION</a>	34

4.5.3.7	SOURCERELEASEUNDEFINED	34
4.5.3.8	TARGETRELEASEUNDEFINED	34
4.5.3.9	UNZIPFAILED	34
4.5.3.10	UPDATEPATHUNAVAILABLE	35
4.5.3.11	USERFAIL	35
4.5.3.12	VERSIONFAIL	35
4.5.3.13	XMLPARSEFAIL	35
4.6	PatchManifestState Class Reference	35
4.6.1	Detailed Description	35
4.6.2	Member Function Documentation	36
4.6.2.1	nameFromId	36
4.6.3	Member Data Documentation	36
4.6.3.1	COMPLETE	36
4.6.3.2	DEPENDENCY	36
4.6.3.3	DOWNLOAD	36
4.6.3.4	GETMANIFEST	36
4.6.3.5	PATCH	36
4.7	PatchVersion Class Reference	36
4.7.1	Detailed Description	37
4.7.2	Member Function Documentation	37
4.7.2.1	getFileName	37
4.7.2.2	getSha1	37
4.7.2.3	release	37
4.7.2.4	resume	37
4.7.2.5	setFileName	38
4.7.2.6	start	38
4.7.3	Member Data Documentation	38
4.7.3.1	instanceId	38
4.7.3.2	onComplete	38
4.7.3.3	onFile	38
4.7.3.4	onStart	38
4.8	PatchVersionVerify Class Reference	39
4.8.1	Detailed Description	39
4.8.2	Member Function Documentation	39



4.8.2.1	getFileName	39
4.8.2.2	getQuickScan	39
4.8.2.3	release	40
4.8.2.4	setFileName	40
4.8.2.5	setQuickScan	40
4.8.2.6	start	40
4.8.3	Member Data Documentation	40
4.8.3.1	instanceId	40
4.8.3.2	onComplete	40
4.8.3.3	onCorruptFile	40
4.8.3.4	onProgress	41
4.8.3.5	onStart	41
<b>5</b>	<b>File Documentation</b>	<b>43</b>
5.1	fileerror.js File Reference	43
5.1.1	Detailed Description	43
5.1.2	Variable Documentation	43
5.1.2.1	fileError	43
5.2	patch.js File Reference	43
5.2.1	Detailed Description	44
5.2.2	Function Documentation	44
5.2.2.1	createPatch	44
5.3	patchmanager.js File Reference	44
5.3.1	Detailed Description	44
5.3.2	Variable Documentation	44
5.3.2.1	patchManager	44
5.4	patchmanifest.js File Reference	45
5.4.1	Detailed Description	45
5.4.2	Function Documentation	45
5.4.2.1	createPatchManifest	45
5.5	patchmanifesterror.js File Reference	45
5.5.1	Detailed Description	45
5.5.2	Variable Documentation	46
5.5.2.1	patchManifestError	46

---

5.6	<a href="#">patchmanifeststate.js File Reference</a>	46
5.6.1	<a href="#">Detailed Description</a>	46
5.6.2	<a href="#">Variable Documentation</a>	46
5.6.2.1	<a href="#">patchManifestState</a>	46
5.7	<a href="#">patchversion.js File Reference</a>	46
5.7.1	<a href="#">Detailed Description</a>	47
5.7.2	<a href="#">Function Documentation</a>	47
5.7.2.1	<a href="#">createPatchVersion</a>	47
5.8	<a href="#">patchversionverify.js File Reference</a>	47
5.8.1	<a href="#">Detailed Description</a>	47
5.8.2	<a href="#">Function Documentation</a>	47
5.8.2.1	<a href="#">createPatchVersionVerify</a>	47

## 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="#">FileError</a> (File error constants and helper functions ) . . . . .	7
<a href="#">Patch</a> (Extract a patch ) . . . . .	8
<a href="#">PatchManager</a> (Manages patches and provides an integrated event structure )	14
<a href="#">PatchManifest</a> (Download/process a patch manifest ) . . . . .	24
<a href="#">PatchManifestError</a> ( <a href="#">Patch</a> manifest error constants and helper functions ) . . .	33
<a href="#">PatchManifestState</a> ( <a href="#">Patch</a> manifest state constants and helper functions ) . .	35
<a href="#">PatchVersion</a> (Enumerate a version file ) . . . . .	36
<a href="#">PatchVersionVerify</a> (Verify a version file ) . . . . .	39



## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">fileerror.js</a> (File containing file error constants and helper functions ) . . . . .	43
<a href="#">patch.js</a> (File containing <a href="#">Patch</a> class and creation function ) . . . . .	43
<a href="#">patchmanager.js</a> (File containing patch manager ) . . . . .	44
<a href="#">patchmanifest.js</a> (File containing <a href="#">PatchManifest</a> class and creation function ) .	45
<a href="#">patchmanifesterror.js</a> (File containing patch manifest error constants and helper functions ) . . . . .	45
<a href="#">patchmanifeststate.js</a> (File containing patch manifest state constants and helper functions ) . . . . .	46
<a href="#">patchversion.js</a> (File containing <a href="#">PatchVersion</a> class and creation function ) . .	46
<a href="#">patchversionverify.js</a> (File containing <a href="#">PatchVersionVerify</a> class and creation function ) . . . . .	47





# Chapter 4

## Class Documentation

### 4.1 FileError Class Reference

File error constants and helper functions.

#### Public Member Functions

- string `nameFromId` (string `id`)

#### Public Attributes

- int `DISKSPACE`
- int `LOCKED`
- int `NONE`
- int `UNKNOWN`

#### 4.1.1 Detailed Description

File error constants and helper functions.

#### 4.1.2 Member Function Documentation

##### 4.1.2.1 string FileError.nameFromId ( string *id* )

converts a file error to a string

#### Parameters

<i>id</i>	file error.
-----------	-------------

**Returns**

stringified name of file error.

**4.1.3 Member Data Documentation****4.1.3.1 int FileError.DISKSPACE**

File path run out of disk space

**4.1.3.2 int FileError.LOCKED**

File locked

**4.1.3.3 int FileError.NONE**

No error

**4.1.3.4 int FileError.UNKNOWN**

Unknown error

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

- [fileerror.js](#)

**4.2 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](#)

#### 4.2.1 Detailed Description

Extract a patch.

#### 4.2.2 Member Function Documentation

##### 4.2.2.1 void Patch.fail ( )

Fail the patch after an action has stalled.

##### 4.2.2.2 string Patch.getBytesLeft ( )

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

#### Returns

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

##### 4.2.2.3 string Patch.getCurrentItemCompressedSize ( )

gets the compressed size of the file in the zip.

#### Returns

Compressed size of the file in the zip.

**4.2.2.4** `string Patch.getCurrentItemDiskFileName ( )`

gets the the zip file currently being used to patched.

**Returns**

the zip file currently being used to patched.

**4.2.2.5** `string Patch.getCurrentItemDiskOffset ( )`

gets the offset in the zip where the patch file exists.

**Returns**

Offset in the zip where the patch file exists.

**4.2.2.6** `string Patch.GetFileName ( )`

gets the file name of path.

**Returns**

File name of path.

**4.2.2.7** `string Patch.getOutputDirectory ( )`

gets the location where patch is applied.

**Returns**

Location where patch is applied.

**4.2.2.8** `double Patch.getRemainingTime ( )`

gets the estimated time remaining in a patch.

**Returns**

estimated time remaining in a patch.

**4.2.2.9** `bool Patch.getStalled ( )`

gets the is the patch stalled waiting for a decision.

**Returns**

Is the patch stalled waiting for a decision.

**4.2.2.10** `string Patch.getTotalBytes ( )`

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

**Returns**

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

**4.2.2.11** `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.

**4.2.2.12** `void Patch.release ( )`

releases the object

**4.2.2.13** `void Patch.resume ( )`

Resume the patch file after an action has stalled.

**4.2.2.14** `void Patch.setFileName ( string value )`

sets the file name of path.

**Parameters**

<i>value</i>	File name of path.
--------------	--------------------

**4.2.2.15** `void Patch.setOutputDirectory ( string value )`

sets the location where patch is applied.

**Parameters**

<i>value</i>	Location where patch is applied.
--------------	----------------------------------

**4.2.2.16** `void Patch.skip ( )`

Skip the patch file after an action has stalled.

#### 4.2.2.17 void Patch.start ( )

Starts the patch manifest.

### 4.2.3 Member Data Documentation

#### 4.2.3.1 string Patch.instanceId

Instance id used to bind proxy object to native object

#### 4.2.3.2 event Patch.onComplete

Triggered when the patch is complete.

##### Parameters

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

#### 4.2.3.3 event Patch.onCorruptDestFile

Triggered when the patch has detected a corrupt dest file.

##### Parameters

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

#### 4.2.3.4 event Patch.onCorruptSourceFile

Triggered when the patch has detected a corrupt source file.

##### Parameters

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

#### 4.2.3.5 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.

#### 4.2.3.6 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.

#### 4.2.3.7 event **Patch.onPrepareComplete**

Triggered when the patch preparation is complete.

##### Parameters

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

#### 4.2.3.8 event **Patch.onPreparePatchFile**

Triggered before patch starts on each file in the patch.

##### Parameters

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

#### 4.2.3.9 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)
----------------	---

#### 4.2.3.10 event **Patch.onStart**

Triggered when patch is started.

#### 4.2.3.11 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](#)

## 4.3 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 [rootManifestByIdById](#) (string id)
- void [verifyManifestById](#) (string manifestId, bool quickScan)

### 4.3.1 Detailed Description

Manages patches and provides an integrated event structure.

### 4.3.2 Member Function Documentation

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

subscribe to the ManifestAdded event

##### Parameters

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

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

subscribe to the ManifestComplete event

##### Parameters

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

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

subscribe to the ManifestDependency event

**Parameters**

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

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

subscribe to the ManifestRemoved event

**Parameters**

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

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

subscribe to the ManifestRequiredComplete event

**Parameters**

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

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

subscribe to the ManifestStart event

**Parameters**

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

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

subscribe to the ManifestStateChange event

**Parameters**

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

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

subscribe to the ManifestVerifyComplete event

**Parameters**

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

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

subscribe to the ManifestVerifyProgress event

**Parameters**

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

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

subscribe to the ManifestVerifyStart event

**Parameters**

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

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

subscribe to the PatchComplete event

**Parameters**

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

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

subscribe to the PatchCorruptDestFile event

**Parameters**

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

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

subscribe to the PatchCorruptSourceFile event

**Parameters**

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

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

subscribe to the PatchDidPatchFile event

**Parameters**

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

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

subscribe to the PatchDownloadError event

**Parameters**

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

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

subscribe to the PatchFileError event

**Parameters**

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

**4.3.2.17 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

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

subscribe to the PatchPrepareComplete event

**Parameters**

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

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

subscribe to the PatchPreparePatchFile event

**Parameters**

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

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

subscribe to the PatchProgress event

**Parameters**

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

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

subscribe to the PatchStart event

**Parameters**

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

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

subscribe to the PatchWillPatchFile event

**Parameters**

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

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

returns the patch manifest by the assigned id

**Parameters**

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

**4.3.2.24 void PatchManager.release ( )**

releases the download manager

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

unsubscribe from the ManifestAdded event

**Parameters**

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

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

unsubscribe from the ManifestComplete event

**Parameters**

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

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

unsubscribe from the ManifestDependency event

**Parameters**

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

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

unsubscribe from the ManifestRemoved event

**Parameters**

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

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

unsubscribe from the ManifestRequiredComplete event

**Parameters**

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

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

unsubscribe from the ManifestStart event

**Parameters**

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

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

unsubscribe from the ManifestStateChange event

**Parameters**

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

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

unsubscribe from the ManifestVerifyComplete event

**Parameters**

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

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

unsubscribe from the ManifestVerifyProgress event

**Parameters**

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

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

unsubscribe from the ManifestVerifyStart event

**Parameters**

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

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

unsubscribe from the PatchComplete event

**Parameters**

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

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

unsubscribe from the PatchCorruptDestFile event

**Parameters**

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

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

unsubscribe from the PatchCorruptSourceFile event

**Parameters**

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

**4.3.2.38 void PatchManager.removePatchDidPatchFileEventHandler ( event *handler* )**

unsubscribe from the PatchDidPatchFile event

**Parameters**

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

**4.3.2.39 void PatchManager.removePatchDownloadErrorEventHandler ( event *handler* )**

unsubscribe from the PatchDownloadError event

**Parameters**

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

**4.3.2.40 void PatchManager.removePatchFileErrorEventHandler ( event *handler* )**

unsubscribe from the PatchFileError event

**Parameters**

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

**4.3.2.41 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

**4.3.2.42 void PatchManager.removePatchPrepareCompleteEventHandler ( event *handler* )**

unsubscribe from the PatchPrepareComplete event

**Parameters**

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



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

unsubscribe from the PatchPreparePatchFile event

**Parameters**

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

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

unsubscribe from the PatchProgress event

**Parameters**

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

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

unsubscribe from the PatchStart event

**Parameters**

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

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

unsubscribe from the PatchWillPatchFile event

**Parameters**

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

**4.3.2.47 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

#### 4.3.2.48 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
-----------	--

#### 4.3.2.49 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
-----------	--

#### 4.3.2.50 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](#)

## 4.4 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 [getRequiredReleaseld](#) ()
- bool [getRequiresElevation](#) ()
- int [getRetryInterval](#) ()
- int [getSourceReleaseld](#) ()
- string [getSourceReleaseName](#) ()
- string [getSourceReleaseSHA1](#) ()
- string [getTargetDirectory](#) ()
- int [getTargetReleaseld](#) ()
- string [getTargetReleaseName](#) ()
- string [getTargetReleaseSHA1](#) ()
- int [getUpcomingReleaseld](#) ()
- string [getUrl](#) ()
- void [release](#) ()
- void [resume](#) ()
- void [setLocal](#) (bool value)
- void [setOutput](#) (string value)
- void [setRetryInterval](#) (int value)
- void [setSourceReleaseld](#) (int value)
- void [setSourceReleaseName](#) (string value)
- void [setSourceReleaseSHA1](#) (string value)
- void [setTargetDirectory](#) (string value)
- void [setTargetReleaseld](#) (int value)
- void [setTargetReleaseName](#) (string value)
- void [setTargetReleaseSHA1](#) (string value)
- void [setUrl](#) (string value)
- void [start](#) ()

### Public Attributes

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

#### 4.4.1 Detailed Description

Download/process a patch manifest.

#### 4.4.2 Member Function Documentation

##### 4.4.2.1 void PatchManifest.dependencyFail ( )

Special failure state because one of the dependencies failed.

#### 4.4.2.2 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.

#### 4.4.2.3 void PatchManifest.fail ( )

Fail the patch manifest after an action is completed.

#### 4.4.2.4 int PatchManifest.getCurrentReleaseId ( )

gets the the current release declared in the version file.

##### Returns

the current release declared in the version file.

#### 4.4.2.5 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.

#### 4.4.2.6 int PatchManifest.getLastError ( )

gets the last error encountered by the manifest.

##### Returns

last error encountered by the manifest.

#### 4.4.2.7 bool PatchManifest.getLocal ( )

gets the true if the manifest is local, false otherwise.

##### Returns

True if the manifest is local, false otherwise.

#### 4.4.2.8 bool PatchManifest.getMaintenance ( )

gets the true if manifest has maintenance flag, false otherwise.

##### Returns

True if manifest has maintenance flag, false otherwise.

#### 4.4.2.9 string PatchManifest.getName ( )

gets the name of the payload.

##### Returns

Name of the payload.

#### 4.4.2.10 string PatchManifest.getOutput ( )

gets the location for patch manifest.

##### Returns

Location for patch manifest.

#### 4.4.2.11 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.

#### 4.4.2.12 `int PatchManifest.getRequiredReleaseId ( )`

gets the the required release declared in the patch manifest.

##### **Returns**

the required release declared in the patch manifest.

#### 4.4.2.13 `bool PatchManifest.getRequiresElevation ( )`

gets the true if elevation required, false otherwise.

##### **Returns**

True if elevation required, false otherwise.

#### 4.4.2.14 `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.

#### 4.4.2.15 `int PatchManifest.getSourceReleaseId ( )`

gets the the id of the source release.

##### **Returns**

the id of the source release.

#### 4.4.2.16 `string PatchManifest.getSourceReleaseName ( )`

gets the the name of the source release.

##### **Returns**

the name of the source release.

#### 4.4.2.17 `string PatchManifest.getSourceReleaseSHA1 ( )`

gets the the sha1 of the source release.

##### **Returns**

the sha1 of the source release.

4.4.2.18 string PatchManifest.getTargetDirectory ( )

gets the directory patch will be installed in.

**Returns**

Directory patch will be installed in.

4.4.2.19 int PatchManifest.getTargetReleaseId ( )

gets the the id of the target release.

**Returns**

the id of the target release.

4.4.2.20 string PatchManifest.getTargetReleaseName ( )

gets the the name of the target release.

**Returns**

the name of the target release.

4.4.2.21 string PatchManifest.getTargetReleaseSHA1 ( )

gets the the sha1 of the target release.

**Returns**

the sha1 of the target release.

4.4.2.22 int PatchManifest.getUpcomingReleaseId ( )

gets the the upcoming release declared in the patch manifest.

**Returns**

the upcoming release declared in the patch manifest.

4.4.2.23 string PatchManifest.getUrl ( )

gets the url to metafile.

**Returns**

Url to metafile.

**4.4.2.24 void PatchManifest.release ( )**

releases the object

**4.4.2.25 void PatchManifest.resume ( )**

Resume the patch manifest after an action is completed.

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

sets the true if the manifest is local, false otherwise.

**Parameters**

<i>value</i>	True if the manifest is local, false otherwise.
--------------	---

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

sets the location for patch manifest.

**Parameters**

<i>value</i>	Location for patch manifest.
--------------	------------------------------

**4.4.2.28 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.
--------------	---

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

sets the the id of the source release.

**Parameters**

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

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

sets the the name of the source release.



**Parameters**

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

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

sets the the sha1 of the source release.

**Parameters**

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

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

sets the directory patch will be installed in.

**Parameters**

<i>value</i>	Directory patch will be installed in.
--------------	---------------------------------------

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

sets the the id of the target release.

**Parameters**

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

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

sets the the name of the target release.

**Parameters**

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

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

sets the the sha1 of the target release.

**Parameters**

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

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

sets the url to metafile.

**Parameters**

<i>value</i>	Url to metafile.
--------------	------------------

**4.4.2.37 void PatchManifest.start ( )**

Start processing the patch manifest.

**4.4.3 Member Data Documentation****4.4.3.1 string PatchManifest.instanceId**

Instance id used to bind proxy object to native object

**4.4.3.2 event PatchManifest.onComplete**

Triggered when the manifest is complete.

**Parameters**

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

**4.4.3.3 event PatchManifest.onDependency**

Triggered for each dependency in the patch manifest.

**Parameters**

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

**4.4.3.4 event PatchManifest.onLoadComplete**

Triggered when a patch manifest has loaded.

**4.4.3.5 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.

#### 4.4.3.6 event PatchManifest.onStart

Triggered when a manifest is started.

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

- [patchmanifest.js](#)

## 4.5 PatchManifestError Class Reference

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

### Public Member Functions

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

### 4.5.1 Detailed Description

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

### 4.5.2 Member Function Documentation

#### 4.5.2.1 string PatchManifestError.nameFromId ( string id )

converts a patch manifest error to a string

#### Parameters

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

**Returns**

stringified name of file error.

**4.5.3 Member Data Documentation****4.5.3.1 int PatchManifestError.DEPENDENCYFAIL**

Dependency failed

**4.5.3.2 int PatchManifestError.DOWNLOADFAIL**

Download failed

**4.5.3.3 int PatchManifestError.INVALIDARGUMENT**

Invalid argument

**4.5.3.4 int PatchManifestError.MAINTENANCE**

Maintenance flag turned on

**4.5.3.5 int PatchManifestError.NONE**

No error

**4.5.3.6 int PatchManifestError.REQUIRESELEVATION**

Elevation required

**4.5.3.7 int PatchManifestError.SOURCERELEASEUNDEFINED**

Source release undefined

**4.5.3.8 int PatchManifestError.TARGETRELEASEUNDEFINED**

Target release undefined

**4.5.3.9 int PatchManifestError.UNZIPFAILED**

Unzip failed

## 4.5.3.10 int PatchManifestError.UPDATEPATHUNAVAILABLE

Update path unavailable

## 4.5.3.11 int PatchManifestError.USERFAIL

Fail method was called

## 4.5.3.12 int PatchManifestError.VERSIONFAIL

Version check failed

## 4.5.3.13 int PatchManifestError.XMLPARSEFAIL

Parsing failed

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

- [patchmanifesterror.js](#)

## 4.6 PatchManifestState Class Reference

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

### Public Member Functions

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

### Public Attributes

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

#### 4.6.1 Detailed Description

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

## 4.6.2 Member Function Documentation

### 4.6.2.1 string PatchManifestState.nameFromId ( string *id* )

converts a patch manifest state to a string

#### Parameters

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

#### Returns

stringified name of file error.

## 4.6.3 Member Data Documentation

### 4.6.3.1 int PatchManifestState.COMPLETE

[Patch](#) Complete

### 4.6.3.2 int PatchManifestState.DEPENDENCY

[Patch](#) dependency required

### 4.6.3.3 int PatchManifestState.DOWNLOAD

[Patch](#) downloading

### 4.6.3.4 int PatchManifestState.GETMANIFEST

Get manifest state

### 4.6.3.5 int PatchManifestState.PATCH

Applying patch

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

- [patchmanifeststate.js](#)

## 4.7 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](#)

### 4.7.1 Detailed Description

Enumerate a version file.

### 4.7.2 Member Function Documentation

#### 4.7.2.1 string PatchVersion.getFileName ( )

gets the file name of the version file.

#### Returns

File name of the version file.

#### 4.7.2.2 string PatchVersion.getSha1 ( )

gets the SHA1 of the version file.

#### Returns

SHA1 of the version file.

#### 4.7.2.3 void PatchVersion.release ( )

releases the object

#### 4.7.2.4 void PatchVersion.resume ( )

Resume the file enumeration.

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

sets the file name of the version file.

##### Parameters

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

#### 4.7.2.6 void PatchVersion.start ( )

Starts the patch version enumeration.

### 4.7.3 Member Data Documentation

#### 4.7.3.1 string PatchVersion.instanceId

Instance id used to bind proxy object to native object

#### 4.7.3.2 event PatchVersion.onComplete

Triggered when the version file enumeration is complete.

##### Parameters

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

#### 4.7.3.3 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

#### 4.7.3.4 event PatchVersion.onStart

Triggered when version file enumeration is started.

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

- [patchversion.js](#)



## 4.8 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](#)

### 4.8.1 Detailed Description

Verify a version file.

### 4.8.2 Member Function Documentation

#### 4.8.2.1 string PatchVersionVerify.getFileName ( )

gets the file name of the version file.

#### Returns

File name of the version file.

#### 4.8.2.2 bool PatchVersionVerify.getQuickScan ( )

gets the enables a quicker but less safe verification.

#### Returns

Enables a quicker but less safe verification.

#### 4.8.2.3 void PatchVersionVerify.release ( )

releases the object

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

sets the file name of the version file.

##### Parameters

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

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

sets the enables a quicker but less safe verification.

##### Parameters

<i>value</i>	Enables a quicker but less safe verification.
--------------	---

#### 4.8.2.6 void PatchVersionVerify.start ( )

Starts the patch manifest.

### 4.8.3 Member Data Documentation

#### 4.8.3.1 string PatchVersionVerify.instanceId

Instance id used to bind proxy object to native object

#### 4.8.3.2 event PatchVersionVerify.onComplete

Triggered when the verification is complete.

##### Parameters

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

#### 4.8.3.3 event PatchVersionVerify.onCorruptFile

Triggered when the verification has detected a corrupt file.

##### Parameters

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

#### 4.8.3.4 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)
----------------	---

#### 4.8.3.5 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 fileerror.js File Reference

File containing file error constants and helper functions.

#### Classes

- class [FileError](#)  
*File error constants and helper functions.*

#### Variables

- [FileError](#) `fileError`

#### 5.1.1 Detailed Description

File containing file error constants and helper functions.

#### 5.1.2 Variable Documentation

##### 5.1.2.1 FileError fileError

precreated global instance of [FileError](#)

### 5.2 patch.js File Reference

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

## Classes

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

## Functions

- void [createPatch](#) ()

### 5.2.1 Detailed Description

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

### 5.2.2 Function Documentation

#### 5.2.2.1 void [createPatch](#) ( )

Create instance of patch

## 5.3 patchmanager.js File Reference

File containing patch manager.

## Classes

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

## Variables

- [PatchManager](#) [patchManager](#)

### 5.3.1 Detailed Description

File containing patch manager.

### 5.3.2 Variable Documentation

#### 5.3.2.1 [PatchManager](#) [patchManager](#)

precreated global instance of the patch manager

## 5.4 patchmanifest.js File Reference

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

### Classes

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

### Functions

- void [createPatchManifest](#) ()

#### 5.4.1 Detailed Description

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

#### 5.4.2 Function Documentation

##### 5.4.2.1 void [createPatchManifest](#) ( )

Create instance of patchManifest

## 5.5 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](#)

#### 5.5.1 Detailed Description

File containing patch manifest error constants and helper functions.

## 5.5.2 Variable Documentation

### 5.5.2.1 PatchManifestError patchManifestError

precreated global instance of [PatchManifestError](#)

## 5.6 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

### 5.6.1 Detailed Description

File containing patch manifest state constants and helper functions.

### 5.6.2 Variable Documentation

#### 5.6.2.1 PatchManifestState patchManifestState

precreated global instance of [PatchManifestState](#)

## 5.7 patchversion.js File Reference

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

### Classes

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

### Functions

- void [createPatchVersion](#) ()



### 5.7.1 Detailed Description

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

### 5.7.2 Function Documentation

#### 5.7.2.1 void createPatchVersion ( )

Create instance of patchVersion

## 5.8 patchversionverify.js File Reference

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

### Classes

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

### Functions

- void [createPatchVersionVerify](#) ( )

### 5.8.1 Detailed Description

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

### 5.8.2 Function Documentation

#### 5.8.2.1 void createPatchVersionVerify ( )

Create instance of patchVersionVerify

# Index

addManifestAddedEventHandler  
PatchManager, 15

addManifestCompleteEventHandler  
PatchManager, 15

addManifestDependencyEventHandler  
PatchManager, 15

addManifestRemovedEventHandler  
PatchManager, 16

addManifestRequiredCompleteEventHandler  
PatchManager, 16

addManifestStartEventHandler  
PatchManager, 16

addManifestStateChangeEventHandler  
PatchManager, 16

addManifestVerifyCompleteEventHandler  
PatchManager, 16

addManifestVerifyProgressEventHandler  
PatchManager, 16

addManifestVerifyStartEventHandler  
PatchManager, 17

addPatchCompleteEventHandler  
PatchManager, 17

addPatchCorruptDestFileEventHandler  
PatchManager, 17

addPatchCorruptSourceFileEventHandler  
PatchManager, 17

addPatchDidPatchFileEventHandler  
PatchManager, 17

addPatchDownloadErrorEventHandler  
PatchManager, 18

addPatchFileErrorEventHandler  
PatchManager, 18

addPatchManifest  
PatchManager, 18

addPatchPrepareCompleteEventHandler  
PatchManager, 18

addPatchPreparePatchFileEventHandler  
PatchManager, 18

addPatchProgressEventHandler  
PatchManager, 19

addPatchStartEventHandler  
PatchManager, 19

addPatchWillPatchFileEventHandler  
PatchManager, 19

COMPLETE  
PatchManifestState, 36

createPatch  
patch.js, 44

createPatchManifest  
patchmanifest.js, 45

createPatchVersion  
patchversion.js, 47

createPatchVersionVerify  
patchversionverify.js, 47

DEPENDENCY  
PatchManifestState, 36

DEPENDENCYFAIL  
PatchManifestError, 34

dependencyFail  
PatchManifest, 25

DISKSPACE  
FileError, 8

DOWNLOAD  
PatchManifestState, 36

DOWNLOADFAIL  
PatchManifestError, 34

expandString  
PatchManifest, 25

fail  
Patch, 9  
PatchManifest, 26

FileError, 7

DISKSPACE, 8

LOCKED, 8

nameFromId, 7

NONE, 8

UNKNOWN, 8

fileError  
fileerror.js, 43

- fileerror.js, [43](#)
  - fileError, [43](#)
- getBytesLeft
  - Patch, [9](#)
- getCurrentItemCompressedSize
  - Patch, [9](#)
- getCurrentItemDiskFileName
  - Patch, [9](#)
- getCurrentItemDiskOffset
  - Patch, [10](#)
- getCurrentReleaseId
  - PatchManifest, [26](#)
- getExtraDataValue
  - PatchManifest, [26](#)
- getFileName
  - Patch, [10](#)
  - PatchVersion, [37](#)
  - PatchVersionVerify, [39](#)
- getLastError
  - PatchManifest, [26](#)
- getLocal
  - PatchManifest, [26](#)
- getMaintenance
  - PatchManifest, [27](#)
- GETMANIFEST
  - PatchManifestState, [36](#)
- getName
  - PatchManifest, [27](#)
- getOutput
  - PatchManifest, [27](#)
- getOutputDirectory
  - Patch, [10](#)
- getQuickScan
  - PatchVersionVerify, [39](#)
- getReleaseName
  - PatchManifest, [27](#)
- getRemainingTime
  - Patch, [10](#)
- getRequiredReleaseId
  - PatchManifest, [27](#)
- getRequiresElevation
  - PatchManifest, [28](#)
- getRetryInterval
  - PatchManifest, [28](#)
- getSha1
  - PatchVersion, [37](#)
- getSourceReleaseId
  - PatchManifest, [28](#)
- getSourceReleaseName
  - PatchManifest, [28](#)
- getSourceReleaseSHA1
  - PatchManifest, [28](#)
- getStalled
  - Patch, [10](#)
- getTargetDirectory
  - PatchManifest, [28](#)
- getTargetReleaseId
  - PatchManifest, [29](#)
- getTargetReleaseName
  - PatchManifest, [29](#)
- getTargetReleaseSHA1
  - PatchManifest, [29](#)
- getTotalBytes
  - Patch, [10](#)
- getUpcomingReleaseId
  - PatchManifest, [29](#)
- getUrl
  - PatchManifest, [29](#)
- getWriteRates
  - Patch, [11](#)
- instanceId
  - Patch, [12](#)
  - PatchManifest, [32](#)
  - PatchVersion, [38](#)
  - PatchVersionVerify, [40](#)
- INVALIDARGUMENT
  - PatchManifestError, [34](#)
- LOCKED
  - FileError, [8](#)
- MAINTENANCE
  - PatchManifestError, [34](#)
- manifestById
  - PatchManager, [19](#)
- nameFromId
  - FileError, [7](#)
  - PatchManifestError, [33](#)
  - PatchManifestState, [36](#)
- NONE
  - FileError, [8](#)
  - PatchManifestError, [34](#)
- onComplete
  - Patch, [12](#)
  - PatchManifest, [32](#)
  - PatchVersion, [38](#)
  - PatchVersionVerify, [40](#)

- onCorruptDestFile
  - Patch, [12](#)
- onCorruptFile
  - PatchVersionVerify, [40](#)
- onCorruptSourceFile
  - Patch, [12](#)
- onDependency
  - PatchManifest, [32](#)
- onDidPatchFile
  - Patch, [12](#)
- onFile
  - PatchVersion, [38](#)
- onFileError
  - Patch, [12](#)
- onLoadComplete
  - PatchManifest, [32](#)
- onPatch
  - PatchManifest, [32](#)
- onPrepareComplete
  - Patch, [13](#)
- onPreparePatchFile
  - Patch, [13](#)
- onProgress
  - Patch, [13](#)
  - PatchVersionVerify, [40](#)
- onStart
  - Patch, [13](#)
  - PatchManifest, [33](#)
  - PatchVersion, [38](#)
  - PatchVersionVerify, [41](#)
- onWillPatchFile
  - Patch, [13](#)
- PATCH
  - PatchManifestState, [36](#)
- Patch, [8](#)
  - fail, [9](#)
  - getBytesLeft, [9](#)
  - getCurrentItemCompressedSize, [9](#)
  - getCurrentItemDiskFileName, [9](#)
  - getCurrentItemDiskOffset, [10](#)
  - getFileName, [10](#)
  - getOutputDirectory, [10](#)
  - getRemainingTime, [10](#)
  - getStalled, [10](#)
  - getTotalBytes, [10](#)
  - getWriteRates, [11](#)
  - instanceId, [12](#)
  - onComplete, [12](#)
  - onCorruptDestFile, [12](#)
  - onCorruptSourceFile, [12](#)
  - onDidPatchFile, [12](#)
  - onFileError, [12](#)
  - onPrepareComplete, [13](#)
  - onPreparePatchFile, [13](#)
  - onProgress, [13](#)
  - onStart, [13](#)
  - onWillPatchFile, [13](#)
  - release, [11](#)
  - resume, [11](#)
  - setFileName, [11](#)
  - setOutputDirectory, [11](#)
  - skip, [11](#)
  - start, [11](#)
- patch.js, [43](#)
  - createPatch, [44](#)
- PatchManager, [14](#)
  - addManifestAddedEventHandler, [15](#)
  - addManifestCompleteEventHandler, [15](#)
  - addManifestDependencyEventHandler, [15](#)
  - addManifestRemovedEventHandler, [16](#)
  - addManifestRequiredCompleteEventHandler, [16](#)
  - addManifestStartEventHandler, [16](#)
  - addManifestStateChangeEventHandler, [16](#)
  - addManifestVerifyCompleteEventHandler, [16](#)
  - addManifestVerifyProgressEventHandler, [16](#)
  - addManifestVerifyStartEventHandler, [17](#)
  - addPatchCompleteEventHandler, [17](#)
  - addPatchCorruptDestFileEventHandler, [17](#)
  - addPatchCorruptSourceFileEventHandler, [17](#)
  - addPatchDidPatchFileEventHandler, [17](#)
  - addPatchDownloadErrorEventHandler, [18](#)
  - addPatchFileErrorEventHandler, [18](#)
  - addPatchManifest, [18](#)
  - addPatchPrepareCompleteEventHandler, [18](#)
  - addPatchPreparePatchFileEventHandler, [18](#)
  - addPatchProgressEventHandler, [19](#)
  - addPatchStartEventHandler, [19](#)
  - addPatchWillPatchFileEventHandler, [19](#)
  - manifestById, [19](#)

- release, [19](#)
- removeManifestAddedEventHandler, [19](#)
- removeManifestCompleteEventHandler, [19](#)
- removeManifestDependencyEventHandler, [20](#)
- removeManifestRemovedEventHandler, [20](#)
- removeManifestRequiredCompleteEventHandler, [20](#)
- removeManifestStartEventHandler, [20](#)
- removeManifestStateChangeEventHandler, [20](#)
- removeManifestVerifyCompleteEventHandler, [21](#)
- removeManifestVerifyProgressEventHandler, [21](#)
- removeManifestVerifyStartEventHandler, [21](#)
- removePatchCompleteEventHandler, [21](#)
- removePatchCorruptDestFileEventHandler, [21](#)
- removePatchCorruptSourceFileEventHandler, [21](#)
- removePatchDidPatchFileEventHandler, [22](#)
- removePatchDownloadErrorEventHandler, [22](#)
- removePatchFileErrorEventHandler, [22](#)
- removePatchManifest, [22](#)
- removePatchPrepareCompleteEventHandlers, [22](#)
- removePatchPreparePatchFileEventHandler, [23](#)
- removePatchProgressEventHandler, [23](#)
- removePatchStartEventHandler, [23](#)
- removePatchWillPatchFileEventHandler, [23](#)
- repairPatchManifest, [23](#)
- rootManifestById, [23](#)
- rootManifestByIdById, [24](#)
- verifyManifestById, [24](#)
- patchManager
  - patchmanager.js, [44](#)
- patchmanager.js, [44](#)
  - patchManager, [44](#)
- PatchManifest, [24](#)
  - dependencyFail, [25](#)
  - expandString, [25](#)
  - fail, [26](#)
  - getCurrentReleaseId, [26](#)
  - getExtraDataValue, [26](#)
  - getLastError, [26](#)
  - getLocal, [26](#)
  - getMaintenance, [27](#)
  - getName, [27](#)
  - getOutput, [27](#)
  - getReleaseName, [27](#)
  - getRequiredReleaseId, [27](#)
  - getRequiresElevation, [28](#)
  - getRetryInterval, [28](#)
  - getSourceReleaseId, [28](#)
  - getSourceReleaseName, [28](#)
  - getSourceReleaseSHA1, [28](#)
  - getTargetDirectory, [28](#)
  - getTargetReleaseId, [29](#)
  - getTargetReleaseName, [29](#)
  - getTargetReleaseSHA1, [29](#)
  - getUpcomingReleaseId, [29](#)
  - getUrl, [29](#)
  - instanceId, [32](#)
  - onComplete, [32](#)
  - onDependency, [32](#)
  - onLoadComplete, [32](#)
  - onPatch, [32](#)
  - onStart, [33](#)
  - release, [29](#)
  - resume, [30](#)
  - setLocal, [30](#)
  - setOutput, [30](#)
  - setRetryInterval, [30](#)
  - setSourceReleaseId, [30](#)
  - setSourceReleaseName, [30](#)
  - setSourceReleaseSHA1, [31](#)
  - setTargetDirectory, [31](#)
  - setTargetReleaseId, [31](#)
  - setTargetReleaseName, [31](#)
  - setTargetReleaseSHA1, [31](#)
  - setUrl, [31](#)
  - start, [32](#)
- patchmanifest.js, [45](#)
  - createPatchManifest, [45](#)
- PatchManifestError, [33](#)
  - DEPENDENCYFAIL, [34](#)
  - DOWNLOADFAIL, [34](#)
  - INVALIDARGUMENT, [34](#)
  - MAINTENANCE, [34](#)
  - nameFromId, [33](#)
  - NONE, [34](#)
  - REQUIRESELEVATION, [34](#)

- SOURCERELEASEUNDEFINED, [34](#)
- TARGETRELEASEUNDEFINED, [34](#)
- UNZIPFAILED, [34](#)
- UPDATEPATHUNAVAILABLE, [34](#)
- USERFAIL, [35](#)
- VERSIONFAIL, [35](#)
- XMLPARSEFAIL, [35](#)
- patchManifestError
  - patchmanifesterror.js, [46](#)
- patchmanifesterror.js, [45](#)
  - patchManifestError, [46](#)
- PatchManifestState, [35](#)
  - COMPLETE, [36](#)
  - DEPENDENCY, [36](#)
  - DOWNLOAD, [36](#)
  - GETMANIFEST, [36](#)
  - nameFromId, [36](#)
  - PATCH, [36](#)
- patchManifestState
  - patchmanifeststate.js, [46](#)
- patchmanifeststate.js, [46](#)
  - patchManifestState, [46](#)
- PatchVersion, [36](#)
  - getFileName, [37](#)
  - getSha1, [37](#)
  - instanceId, [38](#)
  - onComplete, [38](#)
  - onFile, [38](#)
  - onStart, [38](#)
  - release, [37](#)
  - resume, [37](#)
  - setFileName, [37](#)
  - start, [38](#)
- patchversion.js, [46](#)
  - createPatchVersion, [47](#)
- PatchVersionVerify, [39](#)
  - getFileName, [39](#)
  - getQuickScan, [39](#)
  - instanceId, [40](#)
  - onComplete, [40](#)
  - onCorruptFile, [40](#)
  - onProgress, [40](#)
  - onStart, [41](#)
  - release, [39](#)
  - setFileName, [40](#)
  - setQuickScan, [40](#)
  - start, [40](#)
- patchversionverify.js, [47](#)
  - createPatchVersionVerify, [47](#)
- release
  - Patch, [11](#)
  - PatchManager, [19](#)
  - PatchManifest, [29](#)
  - PatchVersion, [37](#)
  - PatchVersionVerify, [39](#)
- removeManifestAddedEventHandler
  - PatchManager, [19](#)
- removeManifestCompleteEventHandler
  - PatchManager, [19](#)
- removeManifestDependencyEventHandler
  - PatchManager, [20](#)
- removeManifestRemovedEventHandler
  - PatchManager, [20](#)
- removeManifestRequiredCompleteEventHandler
  - PatchManager, [20](#)
- removeManifestStartEventHandler
  - PatchManager, [20](#)
- removeManifestStateChangeEventHandler
  - PatchManager, [20](#)
- removeManifestVerifyCompleteEventHandler
  - PatchManager, [21](#)
- removeManifestVerifyProgressEventHandler
  - PatchManager, [21](#)
- removeManifestVerifyStartEventHandler
  - PatchManager, [21](#)
- removePatchCompleteEventHandler
  - PatchManager, [21](#)
- removePatchCorruptDestFileEventHandler
  - PatchManager, [21](#)
- removePatchCorruptSourceFileEventHandler
  - PatchManager, [21](#)
- removePatchDidPatchFileEventHandler
  - PatchManager, [22](#)
- removePatchDownloadErrorEventHandler
  - PatchManager, [22](#)
- removePatchFileErrorEventHandler
  - PatchManager, [22](#)
- removePatchManifest
  - PatchManager, [22](#)
- removePatchPrepareCompleteEventHandler
  - PatchManager, [22](#)
- removePatchPreparePatchFileEventHandler
  - PatchManager, [23](#)
- removePatchProgressEventHandler
  - PatchManager, [23](#)
- removePatchStartEventHandler
  - PatchManager, [23](#)
- removePatchWillPatchFileEventHandler
  - PatchManager, [23](#)

repairPatchManifest  
    PatchManager, [23](#)  
REQUIRESELEVATION  
    PatchManifestError, [34](#)  
resume  
    Patch, [11](#)  
    PatchManifest, [30](#)  
    PatchVersion, [37](#)  
rootManifestByld  
    PatchManager, [23](#)  
rootManifestIdByld  
    PatchManager, [24](#)  
  
setFileName  
    Patch, [11](#)  
    PatchVersion, [37](#)  
    PatchVersionVerify, [40](#)  
setLocal  
    PatchManifest, [30](#)  
setOutput  
    PatchManifest, [30](#)  
setOutputDirectory  
    Patch, [11](#)  
setQuickScan  
    PatchVersionVerify, [40](#)  
setRetryInterval  
    PatchManifest, [30](#)  
setSourceReleaseId  
    PatchManifest, [30](#)  
setSourceReleaseName  
    PatchManifest, [30](#)  
setSourceReleaseSHA1  
    PatchManifest, [31](#)  
setTargetDirectory  
    PatchManifest, [31](#)  
setTargetReleaseId  
    PatchManifest, [31](#)  
setTargetReleaseName  
    PatchManifest, [31](#)  
setTargetReleaseSHA1  
    PatchManifest, [31](#)  
setUrl  
    PatchManifest, [31](#)  
skip  
    Patch, [11](#)  
SOURCERELEASEUNDEFINED  
    PatchManifestError, [34](#)  
start  
    Patch, [11](#)  
    PatchManifest, [32](#)  
  
    PatchVersion, [38](#)  
    PatchVersionVerify, [40](#)  
  
TARGETRELEASEUNDEFINED  
    PatchManifestError, [34](#)  
  
UNKNOWN  
    FileError, [8](#)  
UNZIPFAILED  
    PatchManifestError, [34](#)  
UPDATEPATHUNAVAILABLE  
    PatchManifestError, [34](#)  
USERFAIL  
    PatchManifestError, [35](#)  
  
verifyManifestByld  
    PatchManager, [24](#)  
VERSIONFAIL  
    PatchManifestError, [35](#)  
  
XMLPARSEFAIL  
    PatchManifestError, [35](#)