IOperationListener

Operation and Cache Listeners

The `ncsa.tools.trebuchet.core.IOperationListener` interface defines all listeners which can be added to Trebuchet operations for monitoring purposes, and has the following methods:

```
void operationStarted( String operationId );
void operationFinished( String operationId );
void cacheUpdated( String operationId, long lastEntryId );
void scanningDirectory( String operationId, long id );
```

In addition, the `ncsa.tools.trebuchet.core.IUpdateOperationListener` extends the previous interface with this method:

```
void cacheEntryUpdated( String operationId, long entryId );
```

Not all listeners are required to provide full implementations of all these methods. `cacheUpdated` is called when an entry is added to a cache (usually a `ListCache`); `cacheEntryUpdated` is called when a single line is given a changed value (usually for a `CopyCache`) such as updated size or status. When such a listener needs to access a cache, it usually implements the `ncsa.tools.trebuchet.core.ICacheAccessor` interface, so that when it is added to an operation, the operation sets the appropriate cache on it.

Currently provided implementations

TAG	CLASS	PLUGIN	EXTENDS/IMPLEMENTS
`<logging-listener>`	`ncsa.tools.trebuchet.listeners.LoggingOperationListener`	`ncsa.tools.trebuchet.core`	`IUpdateOperationListener`, `ICacheAccessor`
`<list-metadata-listener>`	`ncsa.tools.trebuchet.listeners.ListToMetadataListener`	`ncsa.tools.trebuchet.listeners`	`ProgressListener`, `ICacheAccessor`
`<list-progress-listener>`	`ncsa.tools.trebuchet.listeners.ListProgressListener`	`ncsa.tools.trebuchet.listeners`	`ProgressListener`
`<file-progress-listener>`	`ncsa.tools.trebuchet.listeners.FileCopyProgressListener`	`ncsa.tools.trebuchet.listeners`	`ProgressListener`, `ICacheAccessor`
`<copy-progress-listener>`	`ncsa.tools.trebuchet.listeners.CopyOperationProgressListener`	`ncsa.tools.trebuchet.listeners`	`ProgressListener`, `ICacheAccessor`
`<transfer-completed-listener>`	`ncsa.tools.trebuchet.listeners.TransferCompletedListener`	`ncsa.tools.trebuchet.listeners`	`ProgressListener`, `ICacheAccessor`

The last five extend a special abstract class, `ProgressListener`, which enables them to send remote events. `FileCopyProgressListener` and `CopyOperationProgressListener` further extend this capability with the ability to write out a summary report of the copy operation.

<logging-listener>

Prints out the relevant entry details or id when each of the methods is called. If the cache reference is `null`, it will simply log the call.

Attributes

NAME	TYPE	DEFAULT VALUE	DESCRIPTION
printFull	`boolean`	*false*	if not true, prints abbreviated information
stdout	`boolean`	*false*	if not true, uses log4j logger

<list-metadata-listener>

Only implements the `cacheUpdated` method. If the entry referenced by the *lastEntryId* is that of a file, it converts it into a `ncsa.tools.common.types.uri.FileMetadata` object, and generates a `ncsa.tools.events.types.events.FileEvent`.

Attributes

None used in Ogrescript.

<list-progress-listener>

Keeps track of how many directories have been scanned and/or how many files have been added to the list.

Attributes

NAME	TYPE	DEFAULT VALUE	DESCRIPTION
reportEvery	`long`	250	send a `cacheUpdated` or `scanningDirectory` event modulo this number

---

<file-progress-listener>

Implements `cacheEntryUpdated` to send `ncsa.tools.trebuchet.events.FileCopyProgressEvent` events.

If there are states specified, the entry state must match one of them in order for the event to be sent. States can be added individually or as a map.

Elements

TAG	TYPE	COUNT	DESCRIPTION
`<report>`	`ncsa.tools.trebuchet.events.ProgressReport`	0:1	wrapper for a progress-report file (see below)
`<send-states>`	`java.util.Map`	0:1	map of qualifying states for which to send events
`<send-status>`	`java.lang.String`	0:N	qualifying state for sending events

<copy-progress-listener>

Implements `cacheEntryUpdated` and `operationFinished` to send `ncsa.tools.trebuchet.events.CopyOperationProgressEvent` events. This is a summary (cumulative) event which reports on the total files copied.

Events are sent when a file copy is successful or at the end of the operation. If one of the attributes is set, these are also taken into account to determine if the event should be sent.

Attributes

NAME	TYPE	DEFAULT VALUE	DESCRIPTION
timeout	`java.lang.Long`	`null`	if set, an event will only be sent if the interval since the last event sent has exceeded the timeout
fileInterval	`java.lang.Long`	`null`	if set, an event will only be sent if the number of files completed modulo this interval == 0
updateInterval	`java.lang.Long`	`null`	if set, an event will only be sent if the number of times the update method has been called modulo this interval == 0

Elements

TAG	TYPE	COUNT	DESCRIPTION
`<report>`	`ncsa.tools.trebuchet.events.ProgressReport`	0:1	wrapper for a progress-report file (see below)

<transfer-completed-listener>

Implements `cacheEntryUpdated` and `operationFinished` to send `ncsa.tools.events.types.events.FileEvent` events. If the *bulk* attribute is set, a single event recording metadata for all transfers is sent when the operation finishes; else an event is sent on successful transfer.

Attributes

NAME	TYPE	DEFAULT VALUE	DESCRIPTION
bulk	`boolean`	*false*	send a single event recording the metadata for all file transfers successfully completed
endpointScheme	`java.lang.String`	*gridftp*	if *useSrc* is true, and the source is local, swap out 'file:/' for this scheme
metadataUpdateOnly	`boolean`	*true*	set the UPDATE_ACTION property on the event to METADATA_ONLY; else, set it to DATA_AND_METADATA
useSrc	`boolean`	*false*	use the source URI rather than the target as the location of the file

---

<progress-report>

Generate a progress report file from the progress operation.

Attributes

NAME	TYPE	DEFAULT VALUE	DESCRIPTION
append	`boolean`	*false*	append to the indicated file

file	`java.io.File`	(required)	write report to this file