OpenEngSB SCM Domain Manual

Version 2.0.0

Table of Contents

Chapter 1. SCM Domain

The source code management (SCM) domain is the tool domain for all SCM tools, like Git or Subversion.

1.1. Description

The SCM Domain polls external repositories for changes of content under source control and provides functionality to copy/export the repository content for further processing.

1.2. Functional Interface

The following listing presents the Java Domain Interface. This interface also contains information about events raised by this domain.

```
/**
 * ScmDomain is an abstraction for working with SCM tools.
 *
 */
public interface ScmDomain extends Domain {

    /**
     * Looks up changes in a remote repository and updates the local repository
     * or checks out a new local repository and returns a list of
     * {@link CommitRef} with the revisions produced since the last update or
     * <code>null</code>.
     */
    List<CommitRef> update();

    /**
     * Exports the files and directories of the HEAD revision from a repository
     * without the SCM specific data in a compressed format.
     */
    File export();

    /**
     * Exports the files and directories of a revision identified by the
     * {@link CommitRef} from a repository without the SCM specific data in a
     * compressed format.
     */
    File export(CommitRef ref);

    /**
     * Check if file identified by its {@code fileName} exists in the HEAD
     * revision and returns <code>true</code> if it does.
     */
    boolean exists(String file);

    /**
     * Retrieves a single {@link File} from a repository identified by its
     * {@code fileName} if it exists in the HEAD revision. If the file does not
     * exist in the revision <code>null</code> will be returned.
     */
    File get(String file);

    /**
     * Check if file identified by its {@code fileName} exists in a revision
```

```
     * identified by the {@link CommitRef} and returns <code>true</code> if it
     * does.
     */
    boolean exists(String fileName, CommitRef version);

    /**
     * Retrieves a single {@link File} from a repository identified by its
     * {@code fileName} if it exists in the revision identified by the
     * {@link CommitRef}. If the file does not exist in the revision
     * <code>null</code> will be returned.
     */
    File get(String fileName, CommitRef ref);

    /**
     * Returns the {@link CommitRef} of the current HEAD in the repository or
     * <code>null</code>.
     */
    CommitRef getHead();

    /**
     * Adds one or more {@link File} existing in the working directory to the
     * repository and commits them with the passed {@code comment}. Returns the
     * {@link CommitRef} of the commit triggered.
     */
    CommitRef add(String comment, File... file);

    /**
     * Removes one or more {@link File} from the working directory of the
     * repository and commits them with a passed {@code comment}. Returns the
     * {@link CommitRef} of the commit triggered.
     */
    CommitRef remove(String comment, File... file);

    /**
     * Tags the actual HEAD of the repository with the passed {@code tagName}
     * and returns the corresponding {@link TagRef} or <code>null</code>.
     */
    TagRef tagRepo(String tagName);

    /**
     * Tags the commit of the repository identified by the {@link CommitRef}
     * with the passed {@code tagName} and returns the corresponding
     * {@link TagRef} or <code>null</code>.
     */
    TagRef tagRepo(String tagName, CommitRef ref);

    /**
     * Resolves and returns the {@link CommitRef} for a {@link TagRef} or
     * <code>null</code> if the reference does not exist.
     */
    CommitRef getCommitRefForTag(TagRef ref);

}
```

1.3. Event Interface

The following interface presents the events an appointment connector can throw:

```
public interface ScmDomainEvents extends DomainEvents {

}
```