OpenEngSB Report Domain Manual

Version 2.0.0

Table of Contents

Chapter 1. Report Domain

The report domain is the tool domain for report generation and management tools.

1.1. Description

The report domain supports basic report generation functionality, like event logging and manual report building. Furthermore it provides basic report management features, like persistent storage of reports and a category system for report storage.

1.2. Functional Interface

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

```
public interface ReportDomain extends Domain {

    /**
     * Generate the report using the information stored under the given reportId and stop collecting d
     * report. The report is added to the specified report category and named as specified by the {@co
     * parameter. The category is created if it is not already present. If there is already a report w
     * under this category it is overwritten.
     *
     * @return the generated report
     * @throws NoSuchReportException if no report with the given {@code reportId} is currently collect
     */
    Report generateReport(String reportId, String category, String reportName) throws NoSuchReportExce

    /**
     *
     * Generate a report based on the currently available data stored for the given {@code reportId}.
     * generated but not stored. Furthermore the data collection is not stopped for this report. This
     * if a preview of the report is needed, but the data collection is not finished yet.
     *
     * @return the generated report draft
     * @throws NoSuchReportException if no report with the given {@code reportId} is currently collect
     */
    Report getDraft(String reportId, String draftName) throws NoSuchReportException;

    /**
     * Start a report data collection. If the data collection process is finished the report can be ge
     * the {@link #generateReport(String, String, String)} method specifying the reportId returned by
     *
     * @return the reportId that can later be used to generate the report by calling
     *         {@link #generateReport(String, String, String)}
     */
    String collectData();

    /**
     * Add the given report part to the report data currently collected for the given {@code reportId}
     * is appended at the end of the report.
     *
     * @throws NoSuchReportException if no report with the given {@code reportId} is currently collect
     */
    void addReportPart(String reportId, ReportPart reportPart) throws NoSuchReportException;

    /**
     * Analyzes the given event and adds all information stored in the event to the report data collec
```

```
     * with the given {@code reportId}, which was initialized by calling {@link #collectData(IdType, S
     *
     * @throws NoSuchReportException if no report with the given {@code reportId} is currently collect
     */
    void processEvent(String reportId, Event event) throws NoSuchReportException;

    /**
     * Get all finished reports of the given category. Reports, which are currently generated and coll
     * included. If the given category does not exist an empty list is returned.
     */
    List<Report> getAllReports(String category);

    /**
     * Store the given report in the report store under the given category. The category is created if
     * present. If there is already a report with the same name under this category it is overwritten.
     */
    void storeReport(String category, Report report);

    /**
     * Remove the given report from the given category. The report is deleted and cannot be restored i
     * stored under another category. If no such report exist no operation is performed.
     */
    void removeReport(String category, Report report);

    /**
     * Get all report categories.
     */
    List<String> getAllCategories();

    /**
     * Remove the given category and all reports stored in this category. If no category with the spec
     * nothing is done.
     */
    void removeCategory(String category);

    /**
     * Creates the given category, which is empty at the start. Reports can later be added by either c
     * {@link #storeReport(String, Report)} or {@link #generateReport(String, String, String)} specify
     * category. If a category exists with the given name no operation is performed.
     */
    void createCategory(String category);
}
```

1.3. Event Interface

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

```
public interface ReportDomainEvents extends DomainEvents {

}
```