In the following, the “local file system” is assumed to be a clone of a GitHub repository.

# Workflows

The following workflows are desired for working with Dokimion.

## Create a new test case in the local file system and send it to a server

1. This involves hand editing XML files. Perhaps a dedicated editing tool would help?

## Edit an existing test case in the local file system and send it to a server

1. This also involves hand editing XML files.
2. [I need to look into what to do with the “last modified time” stored in the file.]

## Backup a project on a server to the local file system

The backup files can be used for:

* Restoring individual test cases if they get corrupted or deleted.
* Restoring a whole project.

## Restore a whole project from the local file system

1. Do we need to be able to specify a subset of the test cases?
2. If so, would filtering using project attributes be helpful?
3. This can be tricky since Dokimion only marks a project as deleted instead of deleting it from its database. Creating a project with the same name/id will undelete the old project with its test cases.

## Restore a (corrupted? deleted?) test case from the local file system

Restoring a deleted test case requires a change to the Dokimion API (see below). Otherwise, the new test case will have a different test case number than the old one had.

## Update the local file system with changes on a server

## Copy a whole project from one server to another

This could be used, for example, for copying a project from the production server to a staging server to provide content to test features and bug fixes.

Do we need to be able to specify a subset of the test cases?

If so, would filtering using project attributes be helpful?

Would backing up the project from one server to the file system and then restoring that project to the other server be “good enough”? Two steps instead of one. The files would not need to be put into GitHub.

# Tools

Currently, I have two tools:

* Updater, which saves and restores individual test cases. (But it doesn’t work if a test case is deleted.)
* Cloner, which copies a whole project from one server to another. (But only if the project never existed on the target server.)

I propose obsoleting the Cloner and extending the Updater tool to:

* Save/restore the project data, not just test case data
* Support server to server copies

# Changes to Dokimion API

We need to be able to undelete a test case. Currently, if we try to PUT the “deleted” field to “false”, the API returns a “Not found” error. I propose allowing PUTs to deleted test cases, but not GETs. Also, the PUT to a deleted test case should not perform the “last modified time” check.

# Pie-in-Sky

* Do testcase development/edits in GitHub.
* Press one button to send to Dokimion. GitHub action?
* Separate file for metadata?
* Don’s definition of “test case” is: Title, description, attributes, steps
* Plugin for Notepad++? “Language syntax”?
* Be creative!

# UI Idea for the maintenance tool

Have a table showing the test cases and some buttons to take action on the selected test cases.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Title In Dokimion** | **ID** | **Status** | **Select** | **Title In File System** |
| TT-1477(Retain paragraph breaks when posting to Paratext) | 10 | = | □ | TT-1477(Retain paragraph breaks when posting to Paratext) |
| TT-1585 (remove isDeveloper flag so code on dev and qa will use new ui) | 11 | > | √ | TT-1585 (remove isDeveloper flag so code on dev and qa will use new ui) |
| 1636 - (Remove the "Delete User" according from the New profile screen) | 15 | » | √ |  |
|  | 19 | « | □ | Members Button - INVITATIONS TAB: |

Status:

* = means the test case is identical in both Dokimion and the file system.
* > means the test case in Dokimion is newer than in the file system (eg. ID 11).
* < means the test case in the file system is newer than in Dokimion.
* » means the test case is only in Dokimion (eg, ID 15).
* « means the test case is only in the file system (eg. ID 19).
* <> means that the timestamps are the same, but the test case content is different.

Actions:

* Update selected test cases from file system to Dokimion.
* Update selected test cases from Dokimion to file system.
* Show differences between the Dokimion and the file system for selected test cases.
* Select all showing
* Clear all selections
* Edit selected test cases in file system
* (Actions will never act on hidden test cases.)

Filters:

* Show all test cases
* Show test cases where Dokimion is newer
* Show test cases where the file system is newer
* Show test cases which Dokimion is missing
* Show test cases which the file system is missing