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. I propose using Markdown as the format for the test cases on the file system. It is much more readable by humans than JSON (the native format for Dokimion) or XML. We would define the conventions to be used so that it will be easily parseable.

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

1. [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.
* Copying a project to a different server.

## 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 all 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

This is similar to “Backup a project” above.

## 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
* Perhaps 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 Goals

Some thoughts:

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

If we put a process in place that requires a GitHub “pull request” to check anything into the test case repo, then we can trigger an Action when a pull to the main branch occurs. We would need to create a command line app that would then update Dokimion with all test cases that are different in GitHub.

# 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, or one is missing, but the test case content is different.

Actions:

* Select all test cases being shown.
* Clear all selections.
* 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.
* 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
* Show test cases which we don’t know which is newer
* Show test cases which differ somehow between Dokimion and the file system

# Using Markdown

See <https://www.markdownguide.org/> for information about Markdown. I propose defining some conventions about how to use it.

I like using Markdown instead of plain text because we can have numbered lists that automatically renumber when an item is inserted or deleted. It also supports tables, which are handy for organizing data. It is a textual format which allows GitHub differences to be meaningful, unlike something like a Word document, which has a binary file format. Markdown is supported by many editors, such as Visual Studio, Visual Studio Code, and Notepad++.

## Header Level Conventions

Header levels will contain specific keywords that the tool will use.

Header Level 1 items:

* ID
* Name
* Description
* Preconditions
* Steps
* Metadata
* Attributes
* Attachments

The ID and Name will have the id and name on the same line. All the others will be on a line by itself, followed by text.

Header Level 2 items:

* Step (used under Steps, one for each Dokimion step)
* Attachment (used under Attachments, one for each attachment)

Header Level 3 items:

* Action
* Expectation

Action and Expectation are under Step.

## Other Conventions

All the text between the Action header and the next header are part of that Action.

All the text between the Expectation header and the next header are part of that Expectation.

The Metadata header will be followed by an unordered list of item names followed by a colon followed by the item value, such as:

* Locked: true

The Attributes header will be followed by an unordered list of attribute names followed by a colon followed by the attribute value, such as:

* TestLodge Section: Deprecated

The Attachment header will be followed by an unordered list of member names followed by a colon followed by the member value, such as:

* ID: xx-yy-zzz
* Title: Foo.txt
* Created time: June 1, 2024 16:02:28 GMT
* Created by: Joe
* Data size: -1

## Example Markdown

This is an example of what the Markdown source could look like.

# ID: 127

# Name: Verify the spell-check works for hyphenated words

# Description

This is a fancy test case description...

It can be multiple paragraphs if you like.

# Preconditions

These are my preconditions...

- Can be multiple lines.

- This is a unordered, bulleted list.

- More items.

# Steps

## Step

### Action

This is formatted text.

You can mark things as \*\*bold\*\* or \*italic\* or `code`.

You can have numbered and bulleted lists.

These are the steps to take:

1. Do this first.

1. Do this next.

1. Do this last

\*\*Part two\*\*

1. Indenting the Part two header 4 spaces keeps the numbered list intact.

1. Part 2 steps

1. More part two.

You can also have tables, like:

| Centered | Right Justified |

| :---: | ---: |

| Paragraph | Text |

| Row two... | has more text |

### Expectation

Verify that no errors are reported.

## Step

### Action

Multiple steps are supported by Dokimion.

However, since the test steps in TestLodge were free-format,

the whole "test steps" section from TestLodge is in one Dokimion step.

### Expectation

# Metadata

- Automated: true

- Broken: false

- Locked: true

- Launch broken: false

- Created time: June 1, 2024 16:02:28 GMT

- Created by: Joe

- Last modified time: June 10, 2024 10:20:08 GMT

- Last modified by: Jimmy

# Attributes

\* TestLodge Section: Deprecated

\* TestLodge Suite: Regression

# Attachments

## Attachment

- ID: xx-yy-zzz

- Title: Foo.txt

- Created time: June 1, 2024 16:02:28 GMT

- Created by: Joe

- Data size: -1

## Rendered Markdown

It is possible to render the Markdown file as HTML or PDF if one would like to print out the test case. An HTML sample rendering of the previous source follows. Note: different tools will render it slightly differently.

# ID: 127

# Name: Verify the spell-check works for hyphenated words

# Description

This is a fancy test case description...  
It can be multiple paragraphs if you like.

# Preconditions

These are my preconditions...

* Can be multiple lines.
* This is a unordered, bulleted list.
* More items.

# Steps

## Step

### Action

This is formatted text.  
You can mark things as **bold** or italic or code.  
You can have numbered and bulleted lists.

These are the steps to take:

1. Do this first.
2. Do this next.
3. Do this last  
   **Part two**
4. Indenting the Part two header 4 spaces keeps the numbered list intact.
5. Part 2 steps
6. More part two.

You can also have tables, like:

| Centered | Right Justified |
| --- | --- |
| Paragraph | Text |
| Row two... | has more text |

### Expectation

Verify that no errors are reported.

## Step

### Action

Multiple steps are supported by Dokimion.  
However, since the test steps in TestLodge were free-format,  
the whole "test steps" section from TestLodge is in one Dokimion step.

### Expectation

# Metadata

* Automated: true
* Broken: false
* Locked: true
* Launch broken: false
* Created time: June 1, 2024 16:02:28 GMT
* Created by: Joe
* Last modified time: June 10, 2024 10:20:08 GMT
* Last modified by: Jimmy

# Attributes

* TestLodge Section: Deprecated
* TestLodge Suite: Regression

# Attachments

## Attachment

* ID: xx-yy-zzz
* Title: Foo.txt
* Created time: June 1, 2024 16:02:28 GMT
* Created by: Joe
* Data size: -1