Projekt **Jake**

Jake Specification

Kurzzusammenfassung:
This document contains the SRS, technical and UI specification for Jake 1.0

Autor:	Simon Wallner
Review:	-
Gruppe:	Gruppe ASE04

Nr	**Datum**	**Autor**	**Änderung**
1	2008-11-09	Simon Wallner	document created
2	2008-12-15	Christian Sutter	Re-compiled document from current SVN

Inhaltsverzeichnis

Teil I

Software Requirements Specification

1.1 Introduction

Jake is an application that simplifies sharing files in small project groups. It's main focus lies on sharing and replication of files and notes over a network.

1.1.1 Core Audience

Jakes core audience are people with minimal to medium IT training, who are comfortable working with internet and email applications.

1.1.2 Example Usage Scenario

A project group of 6-12 project members that shares about 5-100 files (ordinary office files mainly). All project members connected to the internet most of the time, while they are working. Usually only one person works on a file at one time, therefore merging conflicts are rare.

This model scenario may be a small corporate project, or a student project. It is especially useful for projects in distributed teams like architects working in Vienna and Abu Dhabi.

1.2 Terms and Definitions

This sections contains terms and definitions that are used in this document. If you are already familiar with Jake, you might want to skip this section for now.

Jake The Name of the application this software requirements specification is written for.

project A project is the core unit of Jake. It consists of *project members*, *project files* and *project notes*.

project member A project member may share project files with other project members within a project. He/she is uniquely identified by its *user ID*.

user A user of jake, who is not part of a certain project.

project files Files that are shared within a project.

project folder Every project is bind to one local *project folder* in which the project files are stored.

local files Files that reside in the project folder but are not shared.

project notes Besides files, a project may contain notes. Notes are associated to the whole project (in contrary to a note to a file)

local notes Notes that are not shared.

file system The service provided by the operating system to move, copy or remove files and folders

affecting the project A certain action is said to be *affecting the project* if the action is visible to other project members. If Alice modifies a file, this action does not affect the project (i.e no one knows that Alice made a modification) unless Alice decides to announce it.

announce Mark a file or note as "ready to be distributed".

announce message When announcing a file or note a *announce message* may be specified to provide additional information about the changes to the file, or the file itself.

member list The member list is a set(unordered, unique) of *project members* a user explicitly trusts.

explicit trust relation the *explicit trust relation* ($Alice \sim Bob$) holds true, if *Alice* explicitly trusts *Bob*. It is a neither transitive ($\exists a, b, c : a \sim b \wedge b \sim c \not\Rightarrow a \sim c$) nor symmetric ($\exists a, b : a \sim b \not\Rightarrow b \sim a$) relation.

transitive trust relation The *transitive trust relation*($Alice \sim^+ Bob$) holds true, if there exists a directed *path* ($Alice \sim^+ Bob :\Leftrightarrow \exists a_1, a_2, .., a_i : Alice \sim a_1 \sim .. \sim a_i \sim Bob$, i.e. the transitive hull) from *Alice* to *Bob*. It is a transitive ($a \sim^+ b \wedge b \sim^+ c \Rightarrow a \sim^+ c$) but not symmetric ($\exists a, b : a \sim^+ b \wedge b \not\sim^+ a$) relation.

neighborhood of project member $\mathbf{a_0}$ All project members that a_0 explicit trusts: $\{a_i : a_0 \sim_i^a\}$

directed project graph All project members along with their explicit trust relations form the directed project graph

web of trust in a web of trust each member of the web trusts each other member.

peer A *peer* is the abstraction of the client associated to a specific project, seen from a networking perspective.

Besides the above definitions this section uses keywords proposed in RFC 2119[1]. In general, every part of this specification MUST be fulfilled except the parts that are explicitly marked as OPTIONAL or similar.

[1]`http://www.ietf.org/rfc/rfc2119.txt`

1.3 Functional Requirements

1.3.1 Manage Projects

Single Instance with many Projects

At any time, there MUST NOT be more than one instance of Jake running per user session. Within this single instance several projects are supported. The number of concurrent *projects* MAY be limited. Every project MUST have its own *project folder*. Files MUST only be associated with one project, therefore sharing a file with multiple projects is not supported.

If Jake is started, the last application state is restored (as far as one exists), i.e. the last opened projects are reopened on start.

Creating a new Project

In order to create a new project, the user specifies a *project folder* and a *project name*. All existing files in the project folder become *local files*. No *user ID* and network connection is required to create a new project but the project must be bound to a *user ID* in order to invite other users and start sharing files and notes.

Deleting a project

Projects may be deleted within Jake. If a project is deleted all files relating to this project MUST be either deleted from the file system or moved to the OS trash folder, whereas the latter method SHOULD be preferred. The user MUST be prompted to review the deletion and be either approve or decline it. The consequences of this operation MUST be clearly conveyed to the user.

1.3.2 Manage Files

In Jake all files (i.e. *local* and project files) are organized hierarchically in folders. New folders may be created within Jake. Files and folders may be moved within the project folder or be deleted from it. If a file/folder is deleted from the project folder, it MAY either be moved to the OS trash or to a dedicated trash folder inside the project folder. One of this options MUST be chosen for a concrete implementation. If the file/folder is moved to a dedicated trash folder, this trash folder MUST be hidden from the user within Jake, but the user MUST BE able to examine its contents, restore files/folders and empty it.

Files may be added to the project either by moving them into the project folder via the file system or by *importing* them in Jake. These files are *local files* as long as they are not announced.

Changes in the *project folder* MUST be reflected in the application within an acceptable latency.

Local files MAY violate filename constraints (length, illegal characters), *project files* MUST NOT. If a filename violates filename contraints the file name MAY be altered in order to conform with the constraints. This SHOULD happen transparently and obvious to the user (i.e. Übung 1.pdf → Uebung_1.pdf). If filenames are changed, the user MUST be prompted to review the change. He/she may either approve or decline the change.

Batch operations MUST be provided where applicable.

Cached files from the operating system (e.g. `.DS_STORE`, `thumbs.db`, `__MACOSX`, `.trash` etc.) MUST NOT be part of a project and are therefore hidden from the user.

Announcing Files/Notes

Only *project files/notes* are shared among the *project members*. In order to contribute a *local file/note* to the project it must be *announced*.

Changes or new files/notes do not affect the project unless announced.

If the user announces a file/note (or changes to it) he/she may specify an *announce message* to provide additional informations for other project members about the modification/new file.

In order to delete a file/note from the project the delete operation is announced to the other project members.

1.3.3 Manage Notes

In Jake, a *note* is associated generally with a project. Both *local* and *project notes* exist, where *local notes* are notes that are not shared. They are NOT hierarchically ordered and reside in one common location within Jake. Notes are only accessible through Jake. Notes may be created, displayed, modified and deleted. A note consists only of a *body*. The first line of a note SHOULD be emphasized. The length of the note MAY be limited.

Announcing notes works analogous to files.

1.3.4 Tags

Files have a *set* (in the mathematical sense; no duplicate elements, unordered) of *tags*. Tags for notes are OPTIONAL. A *tag* is a string. Constraints (length, no spaces, illegal characters, etc.) MAY be enforced on tags. Changes to tags immediately *affect the project*, and therefore don't need to be announced.

1.3.5 Project Members and the Web of Trust

All project members are equal, no one has special rights or stands above others.

All project members in a project along with their *explicit trust relation* form the directed *project graph*. The *explicit trust relations* are represented by the edges and the project members are represented by the nodes.

Each project member has a set of project members that he explicitly trusts. It is called the *member list.*

the web of trust section will be reformulated shortly

In general the *project graph* is not complete.

Adding and Removing Users

A user is added to the project if an existing project member explicitly trusts him/her. A user remains a project member as long as at least one project member explicitly trusts this user.

Removing project members is not easy. A project member may only be completely removed from a project if he/she is not connected to the project graph anymore. This means, no other user has an *explicit trust relation* to that user ($\nexists a_i : a_i \sim^+ a_0$)

For Example: $Alice \sim^+ Bob$, $Bob \sim^+ Eve$ and $Alice \sim^+ Eve$. If Alice removes Eve from her *member list* Eve still remains a project member, as long as there is at least one project member that trusts Eve ($\exists a_i : a_i \sim^+ Eve$). Users SHOULD be made aware of the web of trust and its implications.

Global Member List

Jake MUST provide a *global member list.* This list contains all project members of the project, i.e. all project members that are connected to the project member who creates the list. This list MUST be created with best effort, it is not reliable. This fact MUST be clearly communicated to the user.

Add/Remove Member Alert

A project member SHOULD be alerted if someone in the project adds or removes a member. Best effort, non reliable.

1.3.6 Auto Add/Remove

A project member may set an *auto add/remove flag* at every project member in his/her member list (written as $\sim^\pm$). Lets make things clear with a short example: $Alice \sim^\pm Bob$. If Bob adds Carol Alice automatically adds Carol as well. If Bob removes Carol from his list Alice does that as well. The following problem may occur when removing project members: $Alice \sim^\pm Bob$, $Alice \sim^\pm Carol$, $Bob \sim Eve$ and $Carol \sim Eve$. Now Bob removes Eve from his member list. In that case Alice does not remove Eve.

Auto add is always stronger than *auto remove.* Alice does not remove Eve unless Carol removes Eve and therefore no *auto add* relations persist.

The *auto add/remove relation* ($a \sim^\pm b$) is transitive ($a \sim^\pm b \wedge b \sim^\pm c \Rightarrow a \sim^\pm c$) but not symmetric ($a \sim^\pm b \nRightarrow b \sim^\pm a$)

1.3.7 Sharing and Synchronization

A peer only communicates with other peers from the same project. A peer MUST only download files/notes from peers that are in its *member list*. A peer MUST know which files are available from the peers in the *member list* and offer this list to the user.

Folders are not synchronized explicitly. The user may not synchronize empty folders. Folders are only synchronized if they are needed by the contained files. (e.g if a file `foo/bar.txt` is created and synchronized, the folder `foo/` will be created to contain the file `bar.txt`. Empty folders MUST NOT be deleted automatically.

A peer SHOULD always deliver any file/note to another peer in the project if requested. It MUST provide information about the available files to other peers from the project. It MUST NOT deliver any data to peers that are not in the project (at the time of the request, as far as the peer knows)

The peer MUST always offer the user to download the latest available version of a file. New versions of notes MUST always be downloaded as soon as they become available.

Deleting Files and Notes

If a user announces a delete operation for a file/note every peer that receives the delete operation MUST delete the file/note. Conflicts may occur in the process of deleting files, folders and notes.

A *synchronization conflict* may occur under the following condition: A project member modifies a project file. If another version of the same file becomes available that is newer (timestamp) then the base version of the modified file (i.e the version of the file before it was modified) a conflict occurs.

Conflict Resolution

Both versions of the conflictuous file and additional information (last edit, last edited by, size, etc) are offered to the user. The user may examine both files and perform either one of the following actions:

- Choose the locally modified file and overwrite the remote version.
- Choose the remote version and discard the local changes by updating the local file to the remote version.
- open both files and merge the files manually, then try to announce the merged file. A new conflict may occur.
- postpone the conflict resolution to a later time. The process of conflict resolution may be restarted at any time.

Additionally both con conflicting version may be saved to a different location.

Resolving conflicts in notes is limited to choosing one version or resolving the conflict later.

Soft Lock

A file or note may be *soft locked*. A *soft lock* consists of a *locking message*. This *locking message* MAY be limited in length. Every user may append, modify or delete a soft lock.

Whenever an action is to be performed on a locked file or note that may change the file/note, the user MUST be prompted to review this operation and either decline or approve it. The locking message MUST be displayed along the prompt.

As with everything, the propagation of the soft lock can only be done with best effort.

1.3.8 Application Start

Jake may be started without any project loaded and without being logged in. In this case, new projects may be created, the user may create a new jabber account, the user may login into the network or receive invitations to projects (requires login)

1.3.9 Pause/Resume a Project

Projects may be paused and resumed. If a project is paused, changes MUST NOT be synchronized with other project members, changes to the project folder in the file system MUST NOT be monitored. The user may not work on a paused projects in Jake (no adding, no changing, no announcing, no browsing, no whatever...). The user may delete a paused project.

Paused projects may be resumed at any time.

1.3.10 Open/Close a Project

Projects may be opened and closed. As in other applications a project must be opened in order work on it. Jake is not aware of closed projects, as if the project never existed.

1.3.11 log in/out

The user may log on/off to/from the network service (XMPP in that case).

1.3.12 Searching

Jake MUST provide appropriate means for searching files, notes, tags, project members.

1.3.13 i18n

Jake SHOULD provide appropriate means to support internationalization, in terms of a user language specific interface.

1.3.14 Preferences

Project specific preferences are stored along the corresponding project. Global preferences MUST be stored in one global file/location. User credentials MUST NOT be stored along the project so that the user might not give away his/her credentials with the project accidentally.

Preference files SHOULD be hidden from the user.

1.3.15 Creating a new Jabber Account

Jake MUST provide sufficient means to create new Jabber accounts from within the application.

1.3.16 General Constraints

Jake MAY enforce additional constraints on strings if necessary. Though these constraints SHOULD affect the user as little as possible (replacing illegal characters in file names with '_' is ok, only allowing `[a-zA-Z]` is not!)

Exceptions

Jake MUST be able to tackle exceptions like `not enough disk space`, `no read/write access to file`, missing config files and things alike in a convenient manner.

1.3.17 Error Messages

Jakes error messages should be simple, clear and easy to understand. Problems should be described from a user's perspective and SHOULD NOT use terms and models that lie outside the user context (i.e. "Network Error: could not open port 666"). If the user wants additional technical information it SHOULD be presented along the error message.

1.3.18 Log Files

Jake MUST provide log files for files, notes and the whole project. A Log file is a chronological list that contains operations performed, the user who caused the operations, when the operation happened and which parts of the project have been influenced. It MAY also contain additional information. Compilation of a log is best effort, non reliable.

1.4 Change History

This section contains changes to this document. Each change must be described detailedly.

Rev. 1, 2008-11-9, Simon Wallner

created document

Rev. 2, 2008-11-15, Simon Wallner

Updated the SRS according the results of the SRS feedback meeting. Some parts are still unclear

Rev. 3, 2008-11-19, Simon Wallner

Updated the SRS according the last meetings and feedback. The following has changed:

- tags for notes are now OPTIONAL
- added option for saving conflicting files to another location
- fixed some typos
- clarified the definition of *transitive trust relation*, changed the notation from $\sim^*$ to $\sim^+$. It might be more easily mistaken for the *auto add/remove relation* but it is now consistent with the transitive hull syntax used in wikipedia.
- note MUST always be downloaded if available, aka autopull notes

Some points are still unclear:

- Version conflicts with deleted files are not specified.
- MUST delete operations always be announced immediately/automatically? aka auto announce delete?
- the web of trust section is still unclear and I'm not sure if we should keep it or drop it.
- user switching: what happens if the user logs off from the network? Will all associated projects be closed? How about working offline?

Rev. 4, 2008-11-24, Simon Wallner

Updated SRS according to a mail from Johannes: Folders are not synchronized directly anymore. It is not supported to share empty folders. Folder may not be tagged.

Teil II

Use Cases

1.5 Terms and definitions, Fields

ID Unique *ID* of the use case. Once set, it is immutable.

Name Descriptive name of the use case. It should be short yet long enough to describe the use case.

Scope The scope of the use case, e.g *per project, per file, etc.* The scope may be chained together like *per project: per file.*

Summary Short summary of the use case.

Goal/Rationale The *goal* and the *rationale* of the use case. What's its goal and why does it matter to the user.

Preconditions Preconditions of the use case that must be met in order to perform this use case.

Triggers Triggers that kick-start the use case.

Primary Scenario The basic course of events.

Alternative Path Deviating paths from the basic course of events.

Postconditions Conditions that must be met after the use case has been executed.

Frequency of use How often the use case will be used. This field is based on a wild guess and is usually something like, *daily, weekly, monthly, etc.*

See also References to other use cases or other related information, e.g. *log in* refers to *log out* and vice versa.

1.6 General

1.6.1 GEN-01: Launch Jake

ID	GEN-01
Name	Launch Jake
Scope	per client instance
Summary	Jake is started and the main window is displayed
Goal/Rationale	The user wants to start Jake when it is not currently running
Preconditions	A running operating system with Jake installed and all JVM and library prerequisites met
Triggers	Launching Jake from the OS
Primary Scenario	1. Jake starts and the main window is displayed. 2. Open projects from the previous session (if any) are restored.
Alternative Path	-
Postconditions	Jake is running. If the user has used Jake before and the previous session's login credentials have been saved, Jake automatically kick-starts use case NET-01.
Exceptions	-
Frequency of use	Daily/Always
See also	-

1.7 Network Operations

1.7.1 NET-01: Log in to Jabber

ID	NET-01
Name	Log in to Jabber
Scope	per client instance
Summary	In order to use Jake, a connection to a Jabber network is required. This connection is established by the user entering their network credentials.
Goal/Rationale	The user wants to connect to the network in order to use Jake
Preconditions	Jake is running (GEN-01) and the user is not yet logged in
Triggers	User interaction (active) or application launch with saved credentials (passive)
Primary Scenario	1. User chooses to log in. 2. The user is prompted to enter username and password for the network. 3. Jake attempts to connect to the server and log in. 4. The user is informed of a successful connection.
Alternative Path	If there are saved credentials from a previous session, Jake uses these to automatically login without explicit user action.
Postconditions	The user is logged in.
Exceptions	If the credentials provided in step 2 are incorrect or the Jabber server cannot be reached, the user shall be informed of this and given the option to resume at step 2 in the primary scenario.
Frequency of use	Daily / Always (on startup)
See also	-

1.7.2 NET-02: Log out from Jabber

ID	NET-02
Name	Log out from Jabber
Scope	per client instance
Summary	Log out from the currently used network ID
Goal/Rationale	In some situations, the user may want to explicitly disconnect from the network (and not just exit the application), e.g. when they want to register a different account or do not want other people using their account to be able to use Jake with their ID. For this reason, they are able to log out, leaving the application in a similar state as after its first run.
Preconditions	The user is logged in (NET-01)
Triggers	User action (active)
Primary Scenario	1. The user chooses to log out. 2. The user is logged out from the network.
Alternative Path	-
Postconditions	The user is logged out.
Exceptions	-
Frequency of use	Monthly / Seldom
See also	-

1.7.3 NET-03: Register Jabber ID

ID	NET-03
Name	Register Jabber ID
Scope	per client instance
Summary	Users who do not yet have a Jabber account need to create one before using Jake. Jake guides them through the process so they can create an account on a server from within Jake.
Goal/Rationale	The user wants a Jabber account to be able to use Jake
Preconditions	Jake is running (GEN-01)
Triggers	User action (active)
Primary Scenario	1. The user chooses to register. 2. The user is prompted to pick a server from a preconfigured list or enter one of their own choice. 3. The user is prompted to fill in a form with the required details (username, password). 4. The account is created on the server. 5. The user is automatically logged in with the new account that was just created.
Alternative Path	The user may cancel this operation, returning to NET-01.
Postconditions	The user has a new Jabber account and is connected to the network with it, now able to use Jake.
Exceptions	If the credentials provided in step 3 are incorrect (e.g. password to short, username already taken) or the Jabber server selected in step 2 cannot be reached, the user shall be informed of this and given the option to resume at the respective step in the primary scenario.
Frequency of use	Yearly / Once (usually one jabber id is enough)
See also	-

1.8 Project Management

1.8.1 PRJ-01: Create Project

ID	PRJ-01
Name	Create Project
Scope	per client isntance
Summary	Creates a Jake project from a (new or existing) local folder
Goal/Rationale	The user wants to create a new project to share with other users
Preconditions	Jake is running (GEN-01)
Triggers	User action (active)
Primary Scenario	1. The user chooses to create a new project. 2. The user specifies a local project folder (which may, but doesn't have to, already exist) and a project name. 3. The project is created and added to the list of active projects. 4. Any files which already exist in the local folder become local files of this project.
Alternative Path	-
Postconditions	The new project has been created and any files already present in the project folder are ready to be shared.
Exceptions	If the chosen directory cannot be accessed (e.g. because of filesystem issues or restrictive permissions) or either the directory itself or a subdirectory of it already is a Jake project, the user is informed of this and given the option to resume at step 2 in the creation process.
Frequency of use	Weekly / Often
See also	-

1.8.2 PRJ-02: Receive project invitation

ID	PRJ-02
Name	Receive project invitation
Scope	-
Summary	When users receive project invitations, they may accept them, resulting in that project being created locally.
Goal/Rationale	In order for Jake to be useful to the user, they need to be member of at least one project. By receiving a project invitation, they can easily join a project.
Preconditions	The user is logged in (NET-01).
Triggers	An invitation is received (passive)
Primary Scenario	1. An invitation to a project from another Jake user is received. 2. The user is asked to accept/deny the invitation. 3. The user selects a target directory for the new project. 4. The project folder is created. 5. The project is added to the current projects.
Alternative Path	If the user declines the invitation, it may either be discarded completely or remain in a list of pending invitations for accepting later.
Postconditions	The project specified by the invitation is now active in the local Jake workspace.
Exceptions	If the chosen target directory already exists and is NOT empty, the user is warned of the problems this may cause and prompted if they want to proceed.
Frequency of use	Weekly/Often
See also	-

1.8.3 PRJ-03: Open project

ID	PRJ-03
Name	Open project
Scope	per client instance
Summary	A user may open an existing project by specifying the location of the project folder. The specified project is loaded.
Goal/Rationale	The user wants to reopen an existing project that is not currently open in Jake, either because it has been closed manually, or for some other reason (e.g. reinstalled Jake)
Preconditions	Jake is running (GEN-01) and the project to be opened is not already open.
Triggers	User action (active)
Primary Scenario	1. The user chooses to open an existing project. 2. The user is asked to select the project folder in the local filesystem. 3. The project is opened.
Alternative Path	If an invalid path is selected or the folder chosen is not a Jake project, the user is given the option to resume at step 2 in the process. Additionally, the application could suggest that the user creates a new project at that path (PRJ-01).
Postconditions	The project is open and ready to use
Exceptions	-
Frequency of use	Monthly/Sometimes
See also	-

1.8.4 PRJ-04: Close project

ID	PRJ-04
Name	Close project
Scope	per project
Summary	A user may close a project. Closed projects are not displayed in Jake.
Goal/Rationale	The user wants to close a project, e.g. when it is not needed for a longer period of time.
Preconditions	Jake is running (GEN-01) and the project to be closed is open and selected.
Triggers	User action (active)
Primary Scenario	1. The user chooses to close the project. 2. Confirmation is requested from the user. 3. The project is closed.
Alternative Path	-
Postconditions	The project is closed, no longer active and no longer visible in Jake.
Exceptions	-
Frequency of use	Monthly/Sometimes
See also	-

1.8.5 PRJ-05: Pause project

ID	PRJ-05
Name	Pause project
Scope	per project
Summary	The user may pause a project. Paused projects are still displayed in Jake but the user may not work on them.
Goal/Rationale	The user wants to temporarily suspend a project (e.g. when it is currently not needed)
Preconditions	Jake is running (GEN-01) and the project is open, active (= not paused) and selected.
Triggers	User action (active)
Primary Scenario	1. The user chooses to pause the project. 2. The project is paused.
Alternative Path	-
Postconditions	The project is paused (not active).
Exceptions	-
Frequency of use	Weekly/Sometimes
See also	-

1.8.6 PRJ-06: Resume project

ID	PRJ-06
Name	Resume project
Scope	per project
Summary	The user may resume a paused project. Resuming a paused project allows the user to work on it again.
Goal/Rationale	The user wants to resume a project that has been paused in order to continue working on it.
Preconditions	Jake is running (GEN-01) and the project to be resumed is open, selected and has previously been paused (PRJ-05)
Triggers	User action (active)
Primary Scenario	1. The user chooses to resume the project. 2. The project is resumed.
Alternative Path	-
Postconditions	The project is active again and the user can once again work with it.
Exceptions	-
Frequency of use	Weekly/Sometimes
See also	-

1.8.7 PRJ-07: Delete project

ID	PRJ-07
Name	Delete project
Scope	per project
Summary	The user may delete a project. By deleting a project, its project folder is deleted.
Goal/Rationale	The user wants to delete a project, e.g. because they are sure they will not need it anymore.
Preconditions	Jake is running (GEN-01) and the project to be deleted is open and selected
Triggers	User action (active)
Primary Scenario	1. The user chooses to delete the project. 2. The user is made aware of the lack of undoability inherent in this action and confirmation is requested. 3. The project is removed from Jake. 4. The folder is deleted from the filesystem or moved to the OS trash.
Alternative Path	-
Postconditions	The project no longer exists within Jake and the local folder has been deleted.
Exceptions	If there is a problem deleting the folder in the filesystem (e.g. locked files), the user is informed of this and restarting the computer and manually deleting the folder is suggested. The project is nonetheless removed from Jake.
Frequency of use	Monthly/Sometimes
See also	-

1.8.8 PRJ-08: Search inside project

ID	PRJ-08
Name	Search inside project
Scope	per project
Summary	The user may perform a search inside the project. All elements (files, folders, users, notes, tags, etc.) that match the search term are displayed in a results section.
Goal/Rationale	In larger projects, single files/folders/notes and users may be hard to find due to the large amount of items in the project. The user wants to find one or more items based on certain criteria.
Preconditions	Jake is running (GEN-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user enters search terms in a search element. 2. Jake presents the results matching the query.
Alternative Path	-
Postconditions	The user is presented with the results of the query and can then perform the usual actions on them.
Exceptions	-
Frequency of use	Very frequently
See also	-

1.9 People Management

1.9.1 PPL-01: Invite user to project (”Add user”)

ID	PPL-01
Name	Invite user to project (”Add user”)
Scope	per project: per user
Summary	A project member may invite other users to a specific project. The invited user is identified by their user id.
Goal/Rationale	The user wants to invite another user to participate in the current project
Preconditions	The user is logged in (NET-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user chooses to invite another user. 2. The user is asked to enter the Jabber ID of the user to be invited. 3. An invitation is sent to the invitee.
Alternative Path	-
Postconditions	An invitation has been sent to the user in question.
Exceptions	If the user entered does not exist on the server or the server for this Jabber ID cannot be contacted, the user is informed of this and given the option to resume at step 2.
Frequency of use	Weekly/Often
See also	-

1.9.2 PPL-02: Explicitly trust project member

ID	PPL-02
Name	Explicitly trust project member
Scope	per project: per project member
Summary	In order to receive files from another project member, the user must explicitly grant trust to this specific project member.
Goal/Rationale	The user wants to receive files from another project member.
Preconditions	Jake is running (GEN-01), a project is open, active and selected and the project member to be trusted exists in the context of the project.
Triggers	User action (active)
Primary Scenario	1. The user chooses to explicitly trust another member. 2. The member is identified as trusted.
Alternative Path	-
Postconditions	The trustee is now trusted and the user can receive files from them.
Exceptions	-
Frequency of use	Weekly/Often
See also	-

1.9.3 PPL-03: Revoke trust (”Remove user”)

ID	PPL-03
Name	Revoke trust (”Remove user”)
Scope	per project: per project member
Summary	The user may revoke the explicit trust granted to a specific project member.
Goal/Rationale	The user wants to revoke trust granted to a specific project member (e.g. because they share malicious files)
Preconditions	Jake is running (GEN-01), a project is open, active and selected and the project member for which trust should be revoked exists in the context of the project.
Triggers	User action (active)
Primary Scenario	1. The user chooses to revoke trust from a project member. 2. The user is asked to confirm this action. 3. Trust is revoked from the chosen member.
Alternative Path	-
Postconditions	The project member in question is no longer trusted.
Exceptions	-
Frequency of use	Monthly/Sometimes
See also	-

1.9.4 PPL-04: Set auto add/remove

ID	PPL-04
Name	Set auto add/remove
Scope	per project: per project member
Summary	The user may set/remove the auto add/remove flag on project members on his/her project member list.
Goal/Rationale	The user wants to have changes to the project member list of certain other members propagated to their own
Preconditions	Jake is running (GEN-01), a project is open, active and selected and the specified user exists in the context of the project
Triggers	User action (active)
Primary Scenario	1. The user chooses to set auto add/remove on a project member. 2. The flag is set for this project member.
Alternative Path	-
Postconditions	Peers from the specified user will now always be added to the user's own member list (and removed, see SRS).
Exceptions	-
Frequency of use	Weekly/Sometimes (e.g. when a new user is added)
See also	-

1.9.5 PPL-05: Show project member list

ID	PPL-05
Name	Show project member list
Scope	per project
Summary	Display a list of project members.
Goal/Rationale	The user wants to see all members of the current project
Preconditions	Jake is running (GEN-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user chooses to display the project member list. 2. The project member list is displayed.
Alternative Path	-
Postconditions	The project member list is displayed.
Exceptions	-
Frequency of use	Daily/Often
See also	-

1.9.6 PPL-07: Receive member alert

ID	PPL-07
Name	Receive member alert
Scope	per project
Summary	If a user is added/removed from the project by another project member, the user will be notified.
Goal/Rationale	The user wants to know who is added to or removed from the project by other members
Preconditions	The user is logged in (NET-01) and the project in question is open and active
Triggers	Incoming member alert (passive)
Primary Scenario	1. A member alert is received by Jake. 2. The user is notified of the new/removed project member.
Alternative Path	-
Postconditions	The user has been made aware of the new/removed project member
Exceptions	-
Frequency of use	Sometimes/weekly
See also	-

1.10 File Management - Local/Offline Operations

1.10.1 FIM-01: Show files

ID	FIM-01
Name	Show files
Scope	per project
Summary	The files in the project are displayed
Goal/Rationale	The user wants to see the files that are part of a project
Preconditions	Jake is running (GEN-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user chooses to display the files of a project. 2. The files are displayed.
Alternative Path	-
Postconditions	The files, if any, are displayed.
Exceptions	-
Frequency of use	Very often
See also	-

1.10.2 FIM-02: Import files

ID	FIM-02
Name	Import files
Scope	per project
Summary	The user specifies files/folder to be imported into Jake. The rename assistant may be awakened by this action.
Goal/Rationale	The user wants to add further files into the project folder
Preconditions	Jake is running (GEN-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user chooses to add further files. 2. The user is prompted to select the desired files. 3. The files are added to the project and copied to the project folder.
Alternative Path	The user may just add the files by means of copying them into the project folder in the local filesystem. If a file doesn't conform to the Jake naming standards, the user will be prompted if they want to rename the file (see FIM-11)
Postconditions	The files are now part of the project
Exceptions	-
Frequency of use	Daily/Often
See also	-

1.10.3 FIM-03: Reflect changes

ID	FIM-03
Name	Reflect changes
Scope	per project: per set of files/folders
Summary	Whenever changes in the project folder occur, Jake reflects it to the user.
Goal/Rationale	Changes in the project folder should be mirrored within Jake
Preconditions	Jake is running (GEN-01) and the project where the change occurred is open and active
Triggers	A change occurs in the filesystem (passive)
Primary Scenario	1. New files are added to the project view and files that no longer exist are removed from it. Files that have changed are updated accordingly.
Alternative Path	-
Postconditions	The change is reflected within the project view
Exceptions	If a newly created file doesn't conform to the Jake naming standards, the user will be prompted if they want to rename the file (see FIM-11)
Frequency of use	Very often
See also	-

1.10.4 FIM-04: Delete files/folders

ID	FIM-04
Name	Delete files/folders
Scope	per project: per set of files/folders
Summary	Files/folders are deleted after requesting confirmation.
Goal/Rationale	The user wants to delete one or more files/folder from within Jake
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one file or folder
Triggers	User action (active)
Primary Scenario	1. The user selects one or more files/folders to be deleted. 2. The user chooses to delete these items. 3. The user is prompted for confirmation. 4. The items are deleted or moved to the OS trash.
Alternative Path	If one or more of the selected files are under a soft lock, the user is informed of this and prompted if they really want to proceed..
Postconditions	The files are no longer part of the project and are moved to the OS trash
Exceptions	If the files can't be deleted for some reason (e.g. access violation because they are still in use by some other program), the user is informed of this and it may be suggested that they close the program and/or try to restart their computer and manually delete the files.
Frequency of use	Daily/Often
See also	-

1.10.5 FIM-05: Rename file/folder

ID	FIM-05
Name	Rename file/folder
Scope	per project: per file/folder
Summary	Rename a file or a folder.
Goal/Rationale	The user wants to rename a file or folder from within Jake.
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one file or folder
Triggers	User action (active)
Primary Scenario	1. The user selects a file/folder to be renamed. 2. The user chooses to rename this item. 3. The user is prompted to enter a new name for the item in question. 4. The item is renamed.
Alternative Path	If the selected file is under a soft lock, the user is informed of this and prompted if they really want to proceed.
Postconditions	The item is renamed both within the project and in the local file system
Exceptions	If there is an error renaming the file (e.g. a file with the same name already exists in the same folder), the user is informed of this may resume the process at step 3.
Frequency of use	Daily/Often
See also	-

1.10.6 FIM-06: Move files/folders

ID	FIM-06
Name	Move files/folders
Scope	per project: per set of files/folders
Summary	Move files or folders within the project folder.
Goal/Rationale	The user wants to move files or folders within the project directory using Jake
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one file or folder as well as a suitable move target.
Triggers	User action (active)
Primary Scenario	1. The user selects one or more files or folders to be moved. 2. The user indicates where they want to move these items. 3. The items are moved, the project view is updated and the changes are applied to the local file system.
Alternative Path	If one or more of the files in question are under a soft lock, the user is informed of this and prompted if they really want to proceed.
Postconditions	The items are in their respective new location both within the project and on the filesystem.
Exceptions	If the operation is invalid in some way (e.g. file access violation), the user is informed of this and no changes are applied. If items of the same name already exist in the target folder, the user is informed of this and given the option to either overwrite all or none of these files.
Frequency of use	Often/Daily
See also	-

1.10.7 FIM-07: Create folder

ID	FIM-07
Name	Create folder
Scope	per project: per containing folder
Summary	Create a new folder from within Jake
Goal/Rationale	The user wants to create a new folder inside a project
Preconditions	Jake is running (GEN-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user selects a parent folder for the new folder. 2. The user chooses to create a new folder at the specified location. 3. The new folder is created.
Alternative Path	-
Postconditions	The new folder is created in the filesystem and displayed in the project.
Exceptions	If the location or folder name are invalid, the user is informed of this and no changes are applied.
Frequency of use	Often/Daily
See also	-

1.10.8 FIM-08: Launch file

ID	FIM-08
Name	Launch file
Scope	per project: per file
Summary	Launch a file with its associated external application from within Jake.
Goal/Rationale	The user wants to launch a file and display its contents
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one file
Triggers	User action (active)
Primary Scenario	1. The user selects a file. 2. The user chooses to launch the selected file. 3. The file is launched by means of the OS.
Alternative Path	If the selected file is under a soft lock, the user is informed of this and prompted if they really want to proceed.
Postconditions	The file is opened in the external application.
Exceptions	If the file cannot be launched for some reason (e.g. access violation, the user is informed accordingly.)
Frequency of use	Very often
See also	-

1.10.9 FIM-09: Launch folder in OS-specific file browser

ID	FIM-09
Name	Launch folder in OS-specific file browser
Scope	per project: per folder
Summary	Open a folder in the os specific file browser.
Goal/Rationale	The user wants to open a folder in the OS's file browser
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one folder
Triggers	User action (active)
Primary Scenario	1. The user selects a folder. 2. The user chooses to launch the selected folder. 3. The folder is launched in the OS file browser.
Alternative Path	-
Postconditions	The folder is open in the OS file browser
Exceptions	If the OS file browser cannot be launched, the user is informed accordingly.
Frequency of use	Very often
See also	-

1.10.10 FIM-10: Run rename assistant

ID	FIM-10
Name	Run rename assistant
Scope	per project: per set of files/folders
Summary	Run the rename assistant to rename files/folders with illegal name.
Goal/Rationale	The user wants to share files/folders that have illegal filenames
Preconditions	Jake is running (GEN-01), a project is open, active and selected and one or more files with illegal names exist within it
Triggers	User action (active)
Primary Scenario	1. The user chooses to have their illegally-named files renamed so they can be shared. 2. The user is prompted whether they really want to perform this operation and is shown a list of files that would be affected. 3. The files are renamed.
Alternative Path	-
Postconditions	The files with illegal filenames have been renamed.
Exceptions	If one or more files could not be renamed (e.g. access violation or a file with the same "clean"name exists already), the user is informed of this and it is suggested that they fix the problem manually.
Frequency of use	Sometimes/Weekly
See also	-

1.10.11 FIM-11: Offer renaming

ID	FIM-11
Name	Offer renaming
Scope	per project: per files/folders
Summary	The rename assistant is implicitly started when files with illegal filenames are added to the project folder.
Goal/Rationale	The user wants to share files/folders that are currently unable to be shared because of an illegal filename.
Preconditions	Jake is running (GEN-01), a project is open and active and one or more files/folders with illegal names exist within
Triggers	On import of illegally-named files (passive), triggered by FIM-02.
Primary Scenario	1. The user is informed about the existence of illegally named files, including a list of affected files, and is asked whether they want to rename them right now. 2. The files are renamed.
Alternative Path	If the user rejects the renaming prompt, the files continue to exist both within the project and on the local filesystem. The user can then manually or automatically rename them at a later date.
Postconditions	The files with illegal filenames have been renamed.
Exceptions	If one or more files could not be renamed (e.g. access violation or a file with the same "clean"name exists already), the user is informed of this and it is suggested that they fix the problem manually.
Frequency of use	Sometimes/weekly
See also	-

1.10.12 FIM-12: Add tag to file

ID	FIM-12
Name	Add tag to file
Scope	per project: per file
Summary	Add a tag to a file (one file and one tag at a time).
Goal/Rationale	The user wants to add a tag to a file.
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one file
Triggers	User action (active)
Primary Scenario	1. The user selects a file. 2. The user chooses to add a tag. 3. The user is prompted to enter a tag name. 4. The tag is added to the file. 5. The tag change is announced to other project members.
Alternative Path	If the file is already tagged with a tag of the same name, the new tag is ignored.
Postconditions	The file is tagged with a tag of the chosen name and this change is announced
Exceptions	-
Frequency of use	Very Often
See also	-

1.10.13 FIM-13: Remove tag from file

ID	FIM-13
Name	Remove tag from file
Scope	per project: per file
Summary	Remove a tag from a file. (one tag from one file at a time).
Goal/Rationale	The user wants to remove a tag from a file.
Preconditions	Jake is running (GEN-01), a project is open, active and selected and a tagged file exists within
Triggers	User action (active)
Primary Scenario	1. The user selects a tagged file. 2. The user chooses to remove a tag from the file. 3. The tag is removed from the file. 4. The tag change is announced to other project members.
Alternative Path	-
Postconditions	The file is no longer tagged with a tag of that name and this change is announced
Exceptions	-
Frequency of use	Very Often
See also	-

1.10.14 FIM-14: Show file properties

ID	FIM-14
Name	Show file properties
Scope	per project: per file
Summary	Show detailed file properties, i.e name, size, last edit, status, etc.
Goal/Rationale	The user wants to get more information about a file in the project
Preconditions	Jake is running (GEN-01) and a project is open, active, selected and contains at least one file
Triggers	User action (active)
Primary Scenario	1. The user selects a file. 2. The user chooses to display further information. 3. Further information about the file is shown.
Alternative Path	-
Postconditions	The user received detailed information about the selected file
Exceptions	-
Frequency of use	Sometimes
See also	-

1.11 File Management - Online/Sharing Operations

1.11.1 FIM-15: Announce file

ID	FIM-15
Name	Announce file
Scope	per project: per file
Summary	Announce a file and specify a announce message.
Goal/Rationale	The user wants to announce a file to the rest of the project members
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one file that has been changed (added, modified, moved, deleted,...)
Triggers	User action (active) or locally modified file with auto-announce enabled (passive)
Primary Scenario	1. The user selects a file. 2. The user chooses to announce the selected file. 3. The file is announced.
Alternative Path	The file is automatically announced should the auto-announce feature be enabled.
Postconditions	The file is announced and a suitable log entry has been created.
Exceptions	-
Frequency of use	Very often
See also	-

1.11.2 FIM-16: Download file

ID	FIM-16
Name	Download file
Scope	per project: per file
Summary	Download a file from a remote peer
Goal/Rationale	The user wants to download a file from a remote peer
Preconditions	The user is logged in (NET-01) and a project is open, active and selected that has a newer remote version of a file available
Triggers	User action (active) or remotely modified file with auto-pull enabled (passive)
Primary Scenario	1. The user selects a file. 2. The user chooses to download the selected file. 3. The file is downloaded.
Alternative Path	The file is automatically downloaded should the auto-pull feature be enabled.
Postconditions	The file exists locally (both in project & filesystem)
Exceptions	-
Frequency of use	Very often
See also	-

1.11.3 FIM-17: Resolve conflict

ID	FIM-17
Name	Resolve conflict
Scope	per project: per file
Summary	Run the conflict resolution assistant.
Goal/Rationale	When a conflict occurs, the user wants to receive simple assistance in resolving it.
Preconditions	The user is logged in (NET-01) and a project is open and active and an announce has been performed
Triggers	A local announce and a remote change of file status have resulted in a conflict situation (passive) or user action (active, choosing to resolve an earlier conflict)
Primary Scenario	1. The user is informed of the conflict and prompted for action to resolve it (and can optionally view, save and edit/merge the two conflicting items). 2. The user chooses a way of resolving the conflict (overwrite remote, overwrite local). 3. The selected action is performed.
Alternative Path	The user can choose to temporarily dismiss the message and resolve the conflict later by some means that results in step 1 in this use case being triggered.
Postconditions	There is no longer an acute conflict or the user is aware of actions that need to be taken.
Exceptions	A new conflict may occur when announcing a merged file and the user is informed of this should it happen.
Frequency of use	Sometimes
See also	-

1.11.4 FIM-18: Set soft lock

ID	FIM-18
Name	Set soft lock
Scope	per project: per file
Summary	Set the softlock for a file, optionally specifying a locking message.
Goal/Rationale	The user wants to edit a file alone, trying to make sure that nobody else edits it at the same time
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one file
Triggers	User action (active)
Primary Scenario	1. The user selects a file. 2. The user chooses to set the file's soft lock, optionally specifying a message to indicate their reason for setting the lock.
Alternative Path	-
Postconditions	The file is locked.
Exceptions	-
Frequency of use	Sometimes
See also	-

1.11.5 FIM-19: Unset soft lock

ID	FIM-19
Name	Unset soft lock
Scope	per project: per file
Summary	Unset the soft lock for a file.
Goal/Rationale	The user has finished editing a file that they have locked and wants to enable other project members to work on it again.
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one locked file
Triggers	User action (active)
Primary Scenario	1. The user selects a locked file. 2. The user chooses to unset the file's soft lock. 3. The file is unlocked.
Alternative Path	-
Postconditions	The file is no longer locked.
Exceptions	-
Frequency of use	Sometimes
See also	-

1.12 Note Management - Local/Offline Operations

1.12.1 NTE-01: Show notes

ID	NTE-01
Name	Show notes
Scope	per project
Summary	Notes can be read from within Jake
Goal/Rationale	The user wants to display notes associated with a project
Preconditions	Jake is running (GEN-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user chooses to display the notes of a project. 2. The notes are displayed.
Alternative Path	If no notes exist (yet), this should be made obvious to the user.
Postconditions	The user can browse through the notes.
Exceptions	-
Frequency of use	Very often
See also	-

1.12.2 NTE-02: Add note

ID	NTE-02
Name	Add note
Scope	per project
Summary	Add a note by specifying its content
Goal/Rationale	The user wants to add a new note
Preconditions	Jake is running (GEN-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user chooses to add a new note. 2. The user is prompted for the note content. 3. The note is added.
Alternative Path	-
Postconditions	The new note is part of the project
Exceptions	If the note content is empty, the user is informed of this and given the option to resume at step 2 in the process.
Frequency of use	Often
See also	-

1.12.3 NTE-03: Delete note

ID	NTE-03
Name	Delete note
Scope	per project: per note
Summary	Delete note(s)
Goal/Rationale	The user wants to delete an unneeded note.
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one note.
Triggers	User action (active)
Primary Scenario	1. The user selects one or more notes to be deleted. 2. The user chooses to delete these notes. 3. The user is prompted for confirmation and informed that this action cannot be undone. 4. The notes are deleted.
Alternative Path	-
Postconditions	The deleted notes no longer exist within the project
Exceptions	-
Frequency of use	Often
See also	-

1.12.4 NTE-04: Edit note

ID	NTE-04
Name	Edit note
Scope	per project: per note
Summary	Editing a note to change its content.
Goal/Rationale	The user wants to edit an existing note to update its contents
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one note.
Triggers	User action(active)
Primary Scenario	1. The user selects a note to be edited. 2. The user is prompted for the notes new content (starting off with its old content). 3. The note is updated.
Alternative Path	-
Postconditions	The note now contains the specified new content and its new ”title” (first line) is displayed in the note view
Exceptions	-
Frequency of use	Often
See also	-

1.12.5 NTE-05: Add tag to note

ID	NTE-05
Name	Add tag to note
Scope	per project: per note
Summary	The user may add a tag to a note (one note at a time).
Goal/Rationale	The user wants to add a tag to a note.
Preconditions	Jake is running (GEN-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user selects a note. 2. The user chooses to add a tag. 3. The user is prompted to enter a tag name. 4. The tag is added to the note. 5. The tag change is announced to other project members.
Alternative Path	If the note is already tagged with a tag of the same name, the new tag is ignored.
Postconditions	The note is tagged with a tag of the chosen name and this change is announced
Exceptions	-
Frequency of use	Very Often
See also	-

1.12.6 NTE-06: Remove tag from note

ID	NTE-06
Name	Remove tag from note
Scope	per project: per note
Summary	The user may remove a tag from a note. (one tag from one note at a time)
Goal/Rationale	The user wants to remove a tag from a note.
Preconditions	Jake is running (GEN-01), a project is open, active and selected and a tagged note exists within
Triggers	User action (active)
Primary Scenario	1. The user selects a tagged note. 2. The user chooses to remove a tag from the note. 3. The tag is removed from the note. 4. The tag change is announced to other project members.
Alternative Path	-
Postconditions	The note is no longer tagged with a tag of that name and this change is announced
Exceptions	-
Frequency of use	Very Often
See also	-

1.13 Note Management - Online/Sharing Operations

1.13.1 NTE-07: Announce note

ID	NTE-07
Name	Announce note
Scope	per project: per note
Summary	Announce a note and specify an announce message.
Goal/Rationale	The user wants to announce a note to the rest of the project members
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one note that has not yet been announced
Triggers	User action (active) or locally modified note with auto-announce enabled (passive)
Primary Scenario	1. The user selects a note. 2. The user chooses to announce the selected note. 3. The note is announced.
Alternative Path	The note is automatically announced should the auto-announce feature be enabled.
Postconditions	The note is announced and a suitable log entry has been created.
Exceptions	-
Frequency of use	Very often
See also	-

1.13.2 NTE-08: Download note

ID	NTE-08
Name	Download note
Scope	per project: per note
Summary	Download a note from a remote peer
Goal/Rationale	The user wants to download a note from a remote peer
Preconditions	The user is logged in (NET-01) and a project is open, active and selected that has a newer remote version of a note available
Triggers	User action (active) or remotely modified note with auto-pull enabled (passive)
Primary Scenario	1. The user selects a note (stub). 2. The user chooses to download the selected note. 3. The note is downloaded.
Alternative Path	The note is automatically downloaded should the auto-pull feature be enabled.
Postconditions	The note is part of the local project
Exceptions	-
Frequency of use	Very often
See also	-

1.13.3 NTE-09: Resolve note conflict

ID	NTE-09
Name	Resolve note conflict
Scope	per project: per note
Summary	Run the conflict resolution assistant.
Goal/Rationale	When a conflict occurs, the user wants to receive simple assistance in resolving it.
Preconditions	The user is logged in (NET-01) and a project is open and active and an announce has been performed on a note
Triggers	A local announce and a remote change of note status have resulted in a conflict situation (passive) or user action (active, choosing to resolve an earlier conflict)
Primary Scenario	1. The user is informed of the conflict and prompted for action to resolve it. 2. The user chooses a way of resolving the conflict (overwrite remote, overwrite local). 3. The selected action is performed.
Alternative Path	The user can choose to temporarily dismiss the message and resolve the conflict later by some means that results in step 1 in this use case being triggered.
Postconditions	There is no longer an acute conflict or the user is aware of actions that need to be taken.
Exceptions	A new conflict may occur when announcing the chosen and the user is informed of this should it happen.
Frequency of use	Sometimes
See also	-

1.13.4 NTE-10: Set soft lock (note)

ID	NTE-10
Name	Set soft lock (note)
Scope	per project: per note
Summary	Set the softlock for a note, optionally specifying a locking message.
Goal/Rationale	The user wants to edit a note alone, trying to make sure that nobody else edits it at the same time
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one note
Triggers	User action (active)
Primary Scenario	1. The user selects a note. 2. The user chooses to set the note's soft lock, optionally specifying a message to indicate their reason for setting the lock.
Alternative Path	-
Postconditions	The note is locked
Exceptions	-
Frequency of use	Sometimes
See also	-

1.13.5 NTE-11: Unset soft lock (note)

ID	NTE-11
Name	Unset soft lock (note)
Scope	per project: per note
Summary	Unset the soft lock for a note.
Goal/Rationale	The user has finished editing a note that they have locked and wants to enable other project members to work on it again.
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one locked note
Triggers	User action (active)
Primary Scenario	1. The user selects a locked note. 2. The user chooses to unset the note's soft lock. 3. The note is unlocked.
Alternative Path	-
Postconditions	The note is no longer locked
Exceptions	-
Frequency of use	Sometimes
See also	-

1.14 Log Operations

1.14.1 LOG-01: Display global project log

ID	LOG-01
Name	Display global project log
Scope	per project
Summary	A list of recent changes is available for the user to browse through
Goal/Rationale	The user wants to see all recent changes in the project
Preconditions	Jake is running (GEN-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user chooses to display the project log. 2. The project log is displayed.
Alternative Path	If there hasn't (yet) been any activity in the project, the user is informed of this.
Postconditions	The user can view the project log
Exceptions	-
Frequency of use	Sometimes
See also	-

1.14.2 LOG-02: Display file log

ID	LOG-02
Name	Display file log
Scope	per project: per file
Summary	Display the log messages associated with a specific file
Goal/Rationale	The user wants to see all recent changes of a file
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one file
Triggers	User action (active)
Primary Scenario	1. The user selects a file. 2. The user chooses to display the log of this file. 3. The file log is displayed.
Alternative Path	-
Postconditions	The user can view the file log
Exceptions	-
Frequency of use	Sometimes
See also	-

1.14.3 LOG-03: Display note log

ID	LOG-03
Name	Display note log
Scope	per project: per note
Summary	Display the log messages associated with a specific note
Goal/Rationale	The user wants to see all recent changes of a note
Preconditions	Jake is running (GEN-01) and a project is open, active and selected and contains at least one note
Triggers	User action (active)
Primary Scenario	1. The user selects a note. 2. The user chooses to display the log of this note. 3. The note log is displayed.
Alternative Path	-
Postconditions	The user can view the note log
Exceptions	-
Frequency of use	Sometimes
See also	-

1.15 Preferences

1.15.1 PRF-01: Edit project preferences

ID	PRF-01
Name	Edit project preferences
Scope	per project
Summary	Project preferences, such as auto-pull/-announce, can be changed by the user
Goal/Rationale	The user wants to change preferences of the project
Preconditions	Jake is running (GEN-01) and a project is open, active and selected
Triggers	User action (active)
Primary Scenario	1. The user chooses to edit project preferences. 2. The user is given a range of options to choose from and change. 3. If the user changes an option, the changes are applied immediately.
Alternative Path	Options considered "dangerous", should there be any, will prompt additional confirmation from the user
Postconditions	The preferences are saved and will be considered from this point on.
Exceptions	If the user enters an invalid value for an option, they will be informed of this and must correct it.
Frequency of use	Sometimes
See also	-

1.15.2 PRF-02: Edit global preferences

ID	PRF-02
Name	Edit global preferences
Scope	per client instance
Summary	Global preferences, such as i18n, can be changed by the user
Goal/Rationale	The user wants to change global preferences
Preconditions	Jake is running (GEN-01)
Triggers	User action (active)
Primary Scenario	1. The user chooses to edit global preferences. 2. The user is given a range of options to choose from and change. 3. If the user changes an option, the changes are applied immediately.
Alternative Path	Options considered ”dangerous”, should there be any, will prompt additional confirmation from the user
Postconditions	The preferences are saved and will be considered from this point on.
Exceptions	If the user enters an invalid value for an option, they will be informed of this and must correct it.
Frequency of use	Sometimes
See also	-

1.16 Change History

Rev 1, 2008-11-29, Simon

Merged the use caes from the wiki into this document.

Rev 2, 2008-11-30, Chris

Extended use cases

Rev 3, 2008-12-01, Simon

Added a *see also* section and ommited the *status* field.

Rev 4. 2008-12-16, Chris

Changes according to the review, some slight generala changes, improved some summaries.

Rev 5, 2009-01-03, Simon

Added a *scope* section to the use cases, added a short explanation of the fileds at the beginning of the use cases. Added the change history based on the svn log.