**Technical description**

[Introduction](#_bookmark0) [Technologies used](#_bookmark1) [Components](#_bookmark2)

[Startup](#_bookmark14) [Login](#_bookmark15)

[Changes tree](#_bookmark3)

[Log view and progress bar](#_bookmark4) [Locking controls](#_bookmark19) [Background operations](#_bookmark5) [Detecting if first load is done](#_bookmark6)

[Synchronization related operations](#_bookmark7) [Cleaning local data](#_bookmark8) [Sequences handling](#_bookmark9)

[Initial load initialization and loading](#_bookmark10) [Error handling](#_bookmark11)

[SymmetricDS customization](#_bookmark12)

* Configuration file of SymmetricDS client

[Workflows](#_bookmark13)

[Startup](#_bookmark14) [Login](#_bookmark15)

[Reading changes](#_bookmark16) [Synchronization](#_bookmark17) [Long operations](#_bookmark18) [Locking controls](#_bookmark19) [Closing](#_bookmark20)

# Introduction

FRMS Synchronization 2.0 is designed to replace old synchronization component within FRMS Reporting Application

(Desktop/nfi-desktop/frms.exe). Synchronization 2.0 is designed to avoid major problems with old synchronization and give clear indication to the user what is happening and when.

FRMS 2.0.7 brings database version check. FRMS 3 uses UUID as primary key for most of the synchronized tables.

# Technologies used

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Provider** | **Component** | **Version** | **Customization** | **Usage** |
|  | Java | 8 |  |  |
|  | Swing |  |  | Synchronization module user interface |
|  | SymmetricDS |  | dependency versions  to end routing fast  to be able to better stop | database replication, in this case replicating data between local database (client package) and central database (server) |
|  | Spring | 4.2.9.RELEASE |  |  |
|  | Apache Commons | latest |  |  |
|  | MyBatis | 3.4.1  1.3.1 (Spring integration) |  | database read and write access |
|  | Jasypt | 1.9.2 |  | password protection in configuration files |
|  | SLF4J |  |  | logging |
|  | PostgreSQL | 9.5 (database)  42.1.1 (driver) |  |  |

**Components**

## Startup

FRMS control panel -> Synchronization/syncclient.bat ( -lang < vi | en > )

Windows batch file parses language from parameters. Batch file then starts java, passing language and property file location as VM parameters, library location and startup class name.

## Login

Login window is implemented as JFrame window. Login window is shown first, if user credentials were not passed in parameters.

Login window authenticates credentials in background, using SwingWorker. If user is authenticated either in local or central database, user info is saved and main UI is started. Connection to both databases is mandatory, and if database versions differ (starting from FRMS version 2.0.7), or one database does not have version at all, login will fail.

## Changes tree

Changes tree is implemented as JTree. Tree has one root node, two child nodes (one for local database, another for central database) and leafs for tables in database and changes in tables.

Changes are read from SymmetricDS tables, collecting count and details from data that has been gathered but not routed to nodes.

Because there may be dozens or hundreds of changes, by default detailed changes are read only for tables, where there are less than 100 changes. User may select button "Refresh (all details)" to load and view all changes in both databases in detail. For central database this can take some time. To limit memory usage and avoid out of memory situations, no details are shown for a table with more than 200 changes.

The amount of detailed changes to show automatically is configurable with parameter sync.tree.details. It can be used e.g. in syncclient.bat like this:

... -Dsync.properties\_file=sync.properties -Dsync.tree.details=50 -cp lib/\* org.arbonaut.formis.nfis.impl.SyncClientApp

## Log view and progress bar

Log view is implemented as JEditorPane with "text/html" editor kit to allow rich formatting. Content is stored in StringBuilder and it is likely that VM will run out of heap space before maximum capacity of StringBuilder is reached. Log view also supports updateable content by keeping old content unchaged and inserting two lines only to the view. Next update or added line will replace temporary content. Progress reporting uses updateable content to report amount of imported batches and data.

Progress bar is updated depending on estimated bacth count to import + number of other stages (starting, registering, ending).

## Locking controls

Long operations will lock UI controls. When operation is started, all controls are disabled (including changes tree). After operation, some or all synchronization controls are enabled depending on local database status.

## Background operations

Long operations are run as SwingWorkers. Operation is run in background thread. Long operations may update UI directly (using SwingUtilities.invokeLater mechanism used in progress tracker to access UI elements in event thread) and report results after completion.

## Detecting if first load is done

Initial load processing is detecting using simple rules:

if local database does not have SymmetricDS tables, initial load is not done if client is not registered to server node, initial load is not done

if sym\_node\_security.initial\_load\_enabled is 1, initial load is not completed

if business data (e.g. in table location) does not exist, initial load is not completed

## Synchronization related operations

### Cleaning local data

Local data is cleaned for district or districts that are assigned to the user in table user\_role\_loc. User level 2 defines district level user and

user level 1 defines provincial level user.

Triggers attached to database tables are disabled in the beginning of synchronization and enabled after synchronization. This is done to prevent SymmetricDS gathering delete events when old data is cleaned or any other events during synchronization. Disabling triggers also prevents possible false creation of history data.

After data is cleaned, SymmetricDS tables are also cleaned to verify that no delete events can escape local database to server.

### Sequences handling

Starting in FRMS version 3, plot, plot history and forest monitoring use UUID as primary keys. Entries in those tables can be created in FRMS without creating collisions between workstations. Entries are synchronized to and from central database based on their UUID instead of combined synchronization key (commune\_code,compt\_code,sub\_compt\_code,plot\_code and some other fields for some tables).

### Initial load initialization and loading

Initial load is initialized in central database by altering table sym\_node\_security directly. When starting initial load, initial\_load\_enabled is set to 1 and initial\_load\_time is set to null. If for some reason initial\_load\_enabled was already 1, it is set back to 0 and after a short delay set back to 1. Setting initial\_load\_enabled to 1 initiates initial load in server. Status of initial\_load\_enabled in server is monitored, and after it changes back to 0 after completing inserting reload events to client, another delay is waited for server to give time to start collecting and routing data to batches. After that loading can be started.

Number of batches that are routed to client is got from server and is used as a base for reporting progress. If batch count is changed during loading, final reported numbers are fixed to show that everything is loaded correctly.

After all batches are loaded (pull returns no batches), status of batches in initial load are verified. If all batches have OK status and initial\_load\_enabled in local database is 0 (first received batch will replicate change from 0 to 1 from initialization and last received batch will replicate change from 1 to 0 from completing inserting reload events for client), initial load has finished.

### Error handling

Error handling in Synchronization module is major part of design and workflows. The following table lists some of the possible situations where errors or situations that would cause errors if not correctly handles, can occur:

|  |  |  |  |
| --- | --- | --- | --- |
| **Error** | **Cause** | **Handling** | **Possible fix** |
| no local database connection | local database not started | error message to user | start local database |
| no central database connection | network connection, wrong configuration, server down | error message to user | retry later |
| no local database SymmetricDS tables | client not registered | normal situation |  |
| client not registered | registration to server failed | error message to user | retry later |
| registration to server fails | server refuses registration, server down | error message to user | retry later |
| initial load not started | high load on server, wrong configuration | error message to user | retry later |
| timeout during synchronization | high load on server, network connection problem | error message to user | retry later |
| server busy | too many concurrent users, server configuration | error message to user | retry later |
| all data not sent to server | high load on server, network connection problem | error message to user | retry later |
| all data not loaded from server | high load on server, network connection problem | error message to user | retry later |
| user interrupts synchronization | user interrupts synchronization | error message to user |  |

### SymmetricDS customization

SymmetricDS base version is 3.7.38. Following customizations are made to Symmetricds on client:

version changed to 3.7.38.frms-SNAPSHOT

Spring version changed to 4.2.2 (compatible with 4.2.9)

faster exit from routing. SymmetricDS is designed to be running continuously. Now routing exits faster and does not wait for possible future changes. This makes push faster

improved cancel support. In several places ongoing operations can be stopped faster and more controllably Custom filter extending DatabaseWriterFilterAdapter is used to count received data and configuration during load.

### Configuration file of symmetric DS client

### Connections, user credentials and other parameters The symmetric DS client reads from FRMS root>\synchronization\sync.properties file.

### The sync.propertied file contains the following properties:

|  |  |  |
| --- | --- | --- |
| parameter | value | example |
| frms.onlinedb.username | username for client synchronization not same as in engine configuration | formisuser |
| frms.onlinedb.password | password for client synchronization |  |
| frms.onlinedb.url | online PG url for client synchronization (server) | jdbc:postgresql://168.11.20.105:5433/data\_forest |
| frms.localdb.username | username to access local database | formisuser |
| frms.localdb.password | password to access local database |  |
| frms.localdb.url | url to access local database | jdbc:postgresql://localhost:5433/data\_forest |
| frms.symmetric.reg\_url | url to access synchronization (server) same as in engine configuration | http://168.11.20.105:8090/sync/corp-000 |
| frms.initial.timeout | Initial load time out in seconds | frms.initial.timeout=180 |

### As the sync.properties file contains user names and passwords for the databases the following parameters are encrypted with Jasypt encryption program (<http://www.jasypt.org/download.html>):

### frms.onlinedb.username

### frms.onlinedb.password

### frms.onlinedb.url

### frms.localdb.username

### frms.localdb.password

### Creating a encrypted string with Jasypt is done as following:

### Generate encrypted strings using Jasypt. Use command encrypt.bat password=<encryption password> input=<string to encrypt>

### Verify generated code using command decrypt.bat password=<encryption password> input=<output from encrypt.bat>

### Copy generated code to property file inside ENC(...), e.g. frms.onlinedb.password=ENC(3DDY7jw6NnQoiNL/Ok35Hg==)

### Example of the encrypted sync.properties file:

### frms.onlinedb.username=ENC(S7YiId45ofpvzOSz60rL/H0qCrnjasJp)

### frms.onlinedb.password=ENC(rXrd3fcB6F3aI16GgHjrL0Uxv9XJ6QbS6hhxxoTxyH8=)

### frms.onlinedb.url=ENC(yK+O0YtJ1XWT2IyMOEsEOcwRJGvJTEyrLHdhCY33uwYUmOz6U79jFewJ8kfTT9mov5Y/clH8ZFw=)

### frms.localdb.username=ENC(gJ2S4nqmPK4yFHAeoVyDt9/jfuhbCorG)

### frms.localdb.password=ENC(cXb0JAdBwsUdMr7e73si+0wj/6L03vVx)

### frms.localdb.url=jdbc:postgresql://localhost:5433/data\_forest

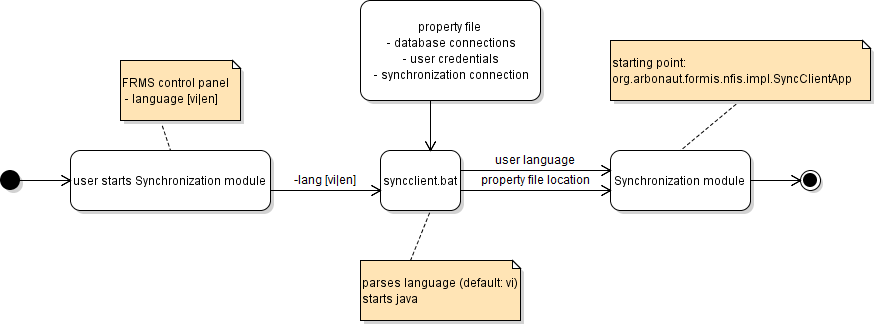
### frms.symmetric.reg\_url=http://smgis2:8090/sync/corp-000

### # Max time (seconds) to wait for initial load to become ready

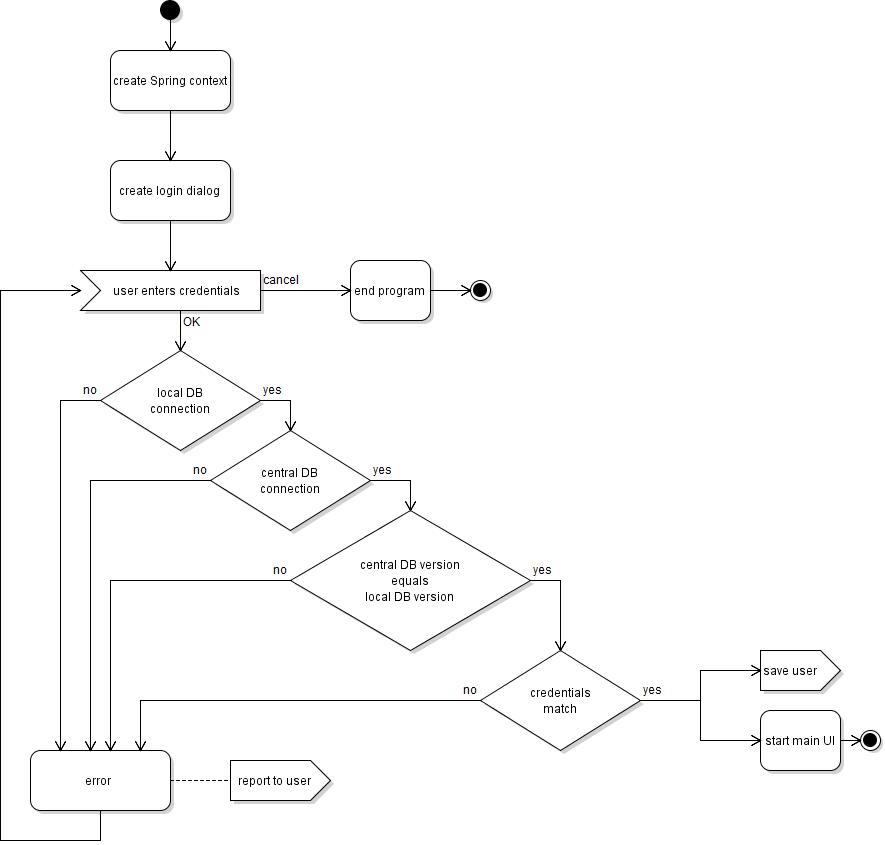
### frms.initial.timeout=180

# Workflows

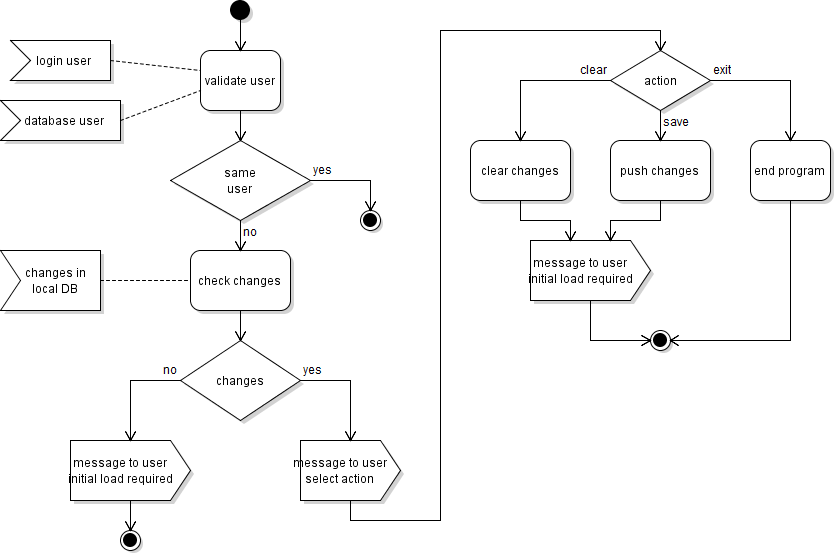
## Startup

****

**Login**

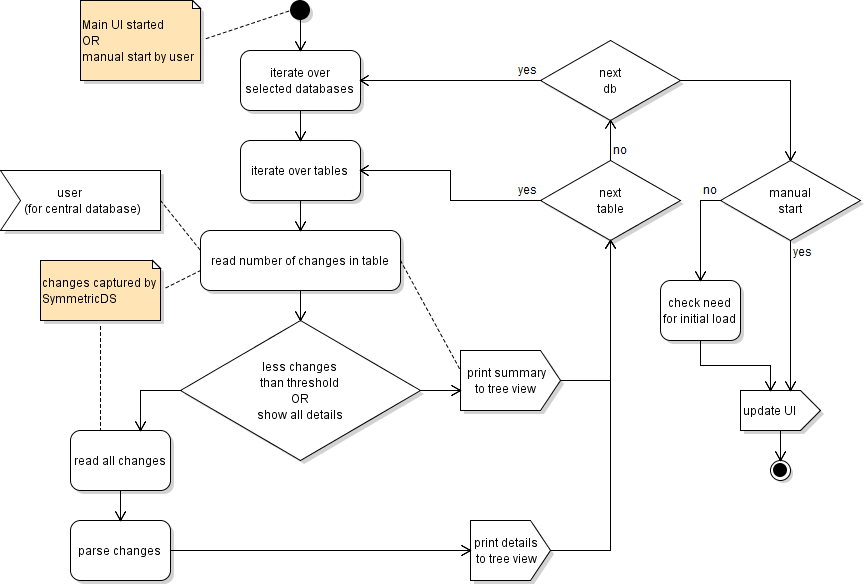


When user is changed, initial load is mandatory. If previous user has made changes to local database, new user must either save changes, discard changes or close Synchronization module.



## Reading changes

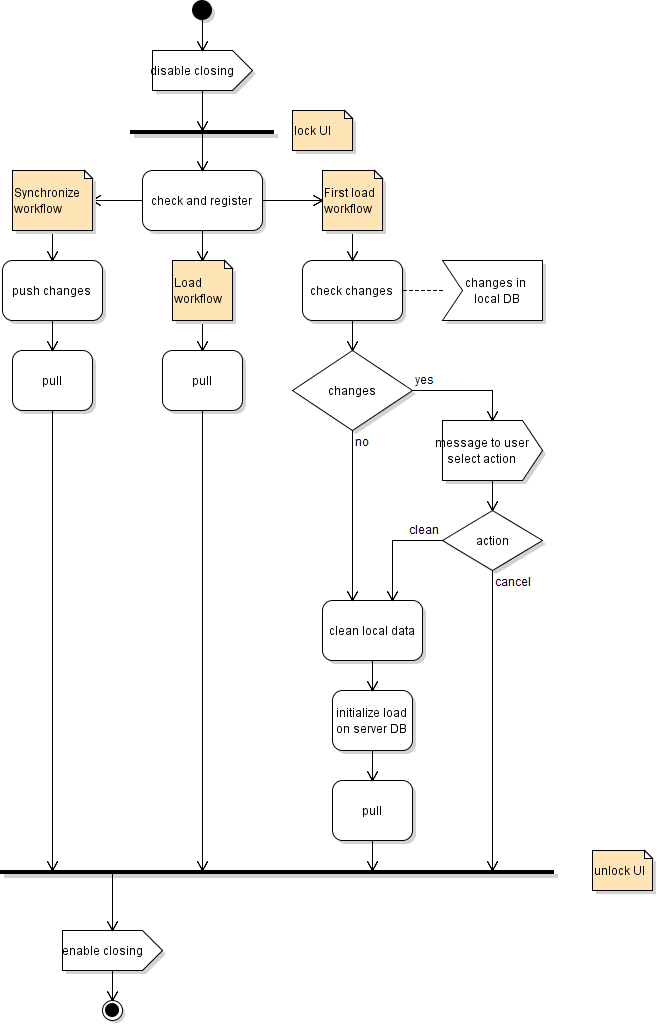
Reading changes is run as long operation in background thread.



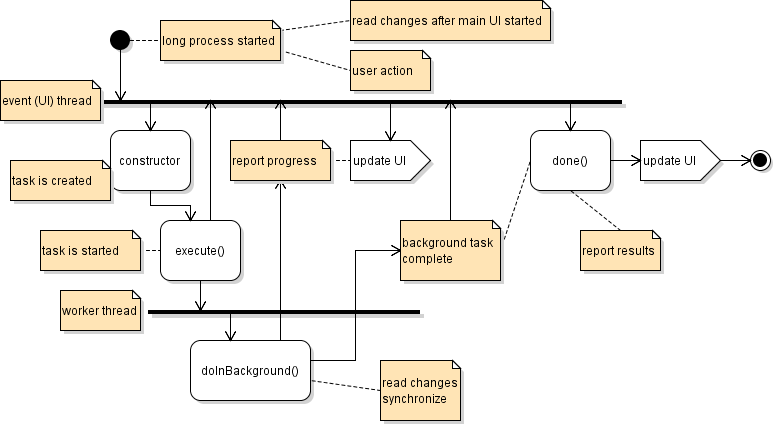
## 

## Synchronization

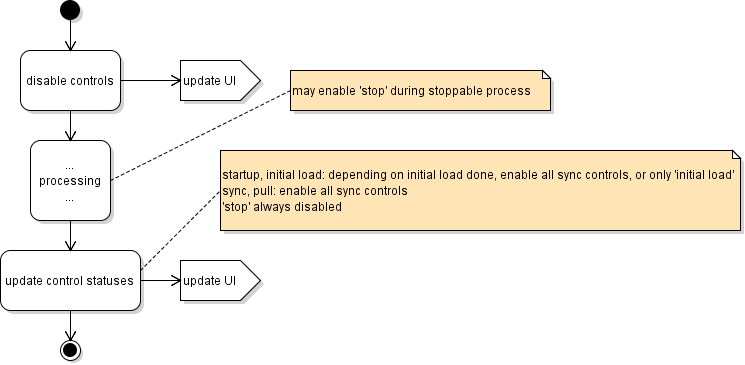
Synchronization operations will always check registration to server node. If client is started for the first time, client is registered to server. All actions are run as long operations in background thread.



## Long operations



**Locking controls**



**Closing**

