SYSTEMATIC

**Detailed Design Description**
KSP-12746

# SFTP Service

Prepared for:

KOMBIT

**Detailed Design Description**

Project: KSP-12746
Revision: Revision: 1.23 Date: 25 Jan 2017
Document: SSE/12746/DDD/0025

# Table of Contents

# 1 Introduction

This document details how the SFTP Service has been integrated into the Serviceplatform. The SFTP Service consists of two parts: An SFTP Server and an SFTP component integrating with the SFTP Server.

The SFTP Service enables user systems to exchange files through the Serviceplatform. In this context the Serviceplatform functions as a delivery system, that given an address for a file, delivers it to the specified recipient(s).

The exchanging of files is done by the SFTP Server. Each user system registered as a user of the SFTP Service, has allocated two private folders on the SFTP Server: a folder for outgoing files, and a folder for incoming files. The exchange of a file is done by moving it from the sending system's out folder to the recipient system's in folder.

There exist two different use cases that users of the SFTP Service can choose to use. These are referred to as UC01 and UC02.

In UC01 the sending system uploads the file that it wishes to transfer to its outgoing folder on the SFTP server along with a trigger file, which contains information about the file, and which system the file should be transferred to. The Serviceplatform detects that the trigger file has been uploaded, parses it, and if the trigger file is valid, the file is transferred to the recipient system. The recipient system is then responsible to detecting that a file has been placed in its folder for incoming files.

There exists two means of specifying the recipient in the trigger file. One is to specify the actual recipient in the trigger file, in which case the file is transferred to the recipient system. The alternative is to specify what is called a virtual user as the recipient in the trigger file. In this case the sending system has to send along additional routing information in the trigger file, which is then matched against routing rules configured on Serviceplatformen in an attempt to identify the actual recipient system. Routing of files using virtual users is reffered to as SFTP Dynamic Routing.

In UC02 the sending system still uploads the file it wishes to send to its outgoing folder on the SFTP server, but instead of uploading a trigger file to the SFTP server, a call is made to the SFTP webservice with a trigger object specifying which system the file should be sent to. After transferring the file the Serviceplatform makes a call to the recipient system to notify it that a file has been delivered to it. The recipient system then makes a call to the Serviceplatform with a business receipt which the Serviceplatform forwards to the sending system.

To avoid files piling up on the SFTP server a standard cleanup job automatically deletes files after a certain timeperiode. If this is not suitable for a given usecase a usersystem can configure their own cleanup by defining a watchlist. The cleanup job will monitor all files on SFTP server, all files on a watchlist is never deleted automatically but a notification mail is sent to usersystem instead.

# 2 Terminology

For this document the following terms will be used:

- **Sending system:** The system that wants to send a file using the SFTP Service.
- **Recipient system:** A receiving system.
- **Trigger object:** An xml structure that contains addressing information for a given file.
- **Technical receipt:** An xml structure that contains the result of a validation of a trigger object.
- **Business receipt:** An xml structure that contains a business response from the recipient system.
- **Recipient notification:** An xml structure that contains a notification to the recipient system that a file has been transferred to it.
- **SFTP webservice:** The webservice developed as part of the SFTP service
- **SFTP virtual user:** A virtual SFTP usersystem is simply a sftp username stored on Serviceplatformen, where special routing is performed if this username is specified as the recipient in a trigger file. The user is not present on the SFTP server.
- **SFTP Dynamic Routing:** A term for routing where the recipient specified in the trigger file is a virtual user.

# 3 Components

The SFTP Service is implemented using both components developed specifically for the service, and existing frameworks within the Serviceplatform.

## 3.1 Serviceplatform components

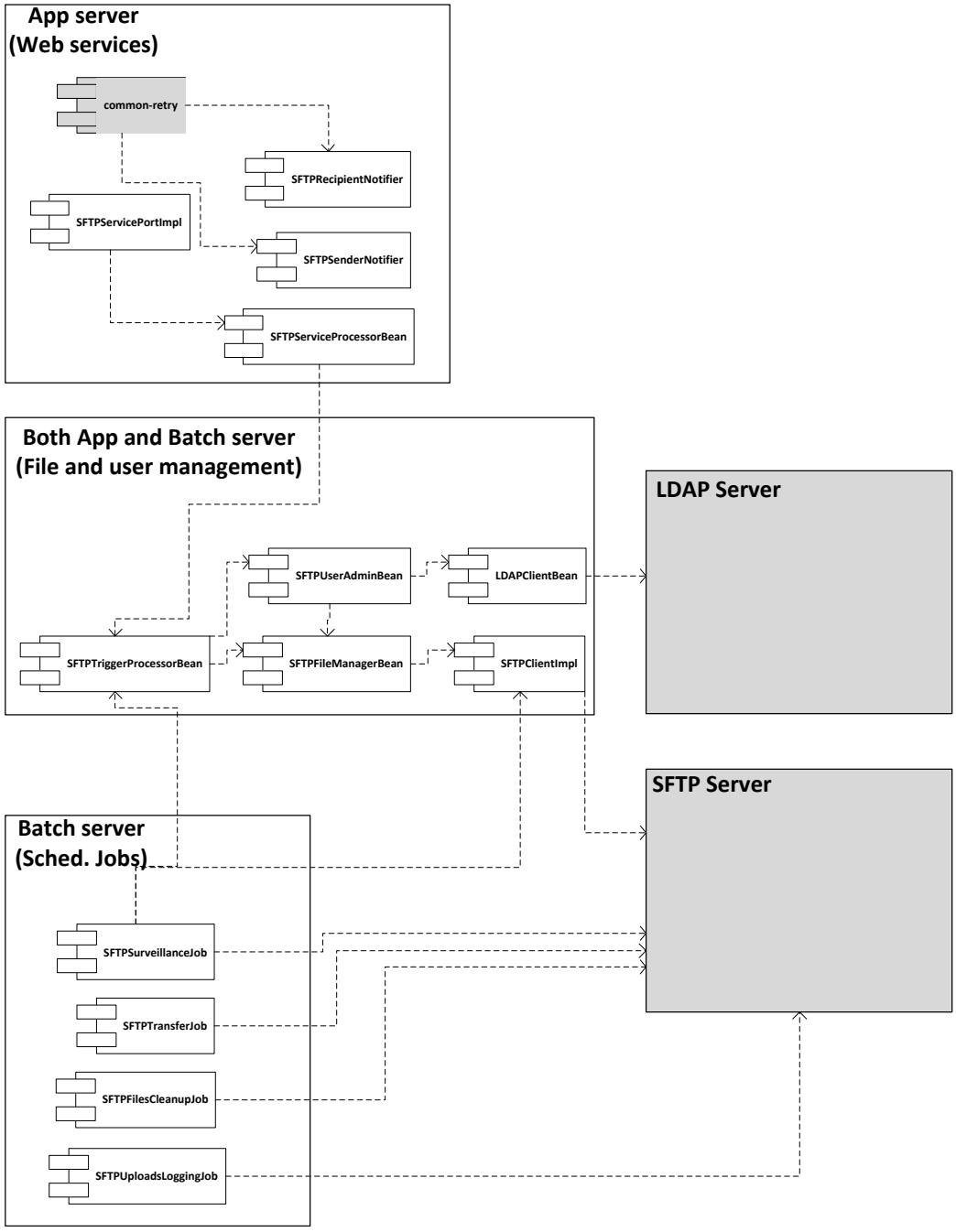
The SFTP Service uses the retry framework to perform callouts and callbacks.

### Retry framework

For UC02 the retry framework developed for the distribution service is used to perform callouts and callbacks to the sending and recipient system. The retry framework ensures that multiple attempts are made to call the recipient and sending system in

## 3.2 SFTP-components

The SFTP Service consists of a set of components distributed across the App servers and the Batch server, with some components being deployed on both types of server. Which components are deployed on which server types are illustrated on the following figure.




**SFTPServicePortImpl**

A Stateless EJB that exposes the webservice for SFTP Webservice. It is implemented as all other webservices on the Serviceplatform as outlined in DDD0007. It exposes the methods *transferFile* and *DeliverBusinessReciept.* It forwards the incoming requests to the SFTPServiceProcessorBean which generates the business response returned to the caller.

**SFTPServiceProcessorBean**

A stateless EJB that contains the business logic for the SFTP webservice. When processing a transferFile request it uses the SFTPTriggerProcessorBean to validate the TriggerObject and return a technical receipt.

**SFTPRecipientNotifer**

A stateless EJB that contains functionality for performing a callout to the recipient system, with a notification that a file has been transferred to it as required when UC02 is used. It is invoked by the retry framework.

**SFTPSenderNotifier**

A stateless EJB that contains functionality for performing a callback to the sending system, with a business receipt delivered by the recipient system as required when UC02 is used. It is invoked by the retry framework.

**SFTPTriggerProcessorBean**

A stateless EJB that contains the business logic for validating a TriggerObject. It uses both the SFTPUserAdminBean and the SFTPFileManagerBean to perform the validation, and the required actions if the validation is successful. This involves creating a transfer task in the DB. The process of validating a triggerobject is explained in greater detail in Section 5.

**SFTPUserAdminBean**

A stateless EJB that contains functionality for managing users of the SFTP Service. This includes e.g. functionality for creating and removing a user's folders on the SFTP server and creating the users on the LDAP server. It uses the SFTPFileManagerBean for performing operations on the SFTP Server, and the LDAPClientBean for performing operations on the LDAP server

**SFTPFileManagerBean**

A stateless EJB that contains high level methods for performing operations on the SFTP server. This includes e.g. locking a file and creating a user's IN and OUT folders on the SFTP server. It uses the SFTPClientImpl class to perform actions on the SFTP Server.

**LDAPClientBean**

A stateless EJB that contains functionality for e.g. creating and removing a user on the LDAP server.

**SFTPClientImpl**

Contains low level operations for an SFTP server. This includes functionality such a removing a specific directory, or changing the permissions on a specific file.

**SFTPSurveillanceJob**

A scheduled job that periodically checks the OUT folder for all users on the SFTP server for trigger files. If it finds a triggerfile, it validates its content using the SFTPTriggerProcessorBean and uploads a technical receipt to the IN folder using the SFTPClientImpl.

**SFTPTransferJob**

A scheduled job that periodically reads all transfer tasks from the DB. It uses the SFTPClientImpl to move the specified file from the Senders OUT folder to the recipients IN Folder.

It checks if the file should be added to the files watchlist based on the configured filters. If a file matches multiple filters one manual treatment task is created.

**SFTPFilesCleanupJob**

A scheduled job that checks all IN and/or OUT folders on the SFTP server for files that have been on the SFTP server for more than 30 days without being modified. For each such file that is found, a notification is sent to the owner of the file. If it is still present 10 days later the file is deleted. The job uses the SFTPClientImpl to perform file operations on the SFTP Server.

However if file is mentioned in a watchlist notification and deletion is not performed. Jobs examines file according the watchlist thresholds and if these have been exceeded notification is sent and/or a manual treatment task is created.

**SFTPUploadsLoggingJob**

A scheduled job that daily parses the logfile for the last 24 hours from the SFTP server, and adds log entries for every file upload and download, as well as everytime a user logged in and out of the SFTP server.

case the first attempt failed.

# 4 Dynamic Routing

Dynamic routing refers to the case where the recipient specified in the trigger file is a virtual user. When this is the case routing is done based on the xml structure SFTPDynamicRoutingInfo which is sent along in the trigger file. The SFTPDynamicRoutingInfo structure is defined in the file SFTPDynamicRoutingInfo.xsd which can be found in the SFTP Types module in the code base.

An example of a trigger file with the SFTPDynamicRoutingInfo structure can be seen below:

```
<ns:Trigger
xmlns:ns="http://serviceplatformen.dk/xml/wsd
l/soap11/SFTP/1/types">
    <FileDescriptor>
        <FileName>file1.txt</FileName>
        <SizeInBytes>100</SizeInBytes>
        <Sender>sender</Sender>

<SendersFileId>sendersFileId</SendersFileId>
        <Recipients>%recipient%</Recipients>
    </FileDescriptor>
    <FileContentDescriptor>
        <SFTPDynamicRoutingInfo>
            <InfRef>%infref%</InfRef>

<SenderAuthority>%senderauth%</SenderAuthorit
y>
```

```
            <TransactionId>ca02695f-497f-
496d-86b1-4e9ecbd800a9</TransactionId>
            <SenderTimestamp>2016-06-
27T08:48:00+02:00</SenderTimestamp>

<RecipientAuthority>%recipientauth%</Recipien
tAuthority>
        </SFTPDynamicRoutingInfo>
    </FileContentDescriptor>
</ns:Trigger>
```

The different fields in the SFTP Dynamic info structure are used for the following:

- Infref: Specifies which integration the file being sent is related to
- SenderAuthority: The authority that the sending system is sending the file on behalf of
- RecipientAuthority: The authority that the recipient system is acting on behalf of.
- SenderIt-system: The system sending the file on behalf of the authority. Should match an actual usersystem on Serviceplatformen.
- RecipientIt-system: The intended recipient system the file should be delivered to. Should match an actual usersystem on Serviceplatformen.

For SFTP Dynamic routing, routing rules must be configured on Serviceplatformen, that specifies which which authorities are allowed to send files to each other using SFTP Dynamic Routing for a given integration. Along with the Routing information it is stored which actual recipient system files should be transferred to if they match a specific routing rule.

Dynamic routing rules are placed in a table that looks as follows:

| sftp_dynamic_routing_rules | |
|---|---|
| | |
| | **Infef**<br>**SenderAuthority**<br>**SenderIt-system**<br>**RecipientAuthority**<br>**RecipientIt-system**<br>**SFTPUsername** |

For flexibility each of these fields are represented in the database as a string, and the SFTP service does not require the values to follow any specific format.

There exist two means of searching for a dynamic routing rule, called implicit and explicit.

*Implicit* routing is done in the following manner:

- For *implicit* routing rules stored in the database the following fields must be filled out: Infref, SenderAuthority, SenderIt-system, RecipientAuthority and SFTPUsername. RecipientIt-system must be empty.
- The sender must provide the following fields in the SFTPDynamicRoutingInfo structure: Infref, SenderAuthority, SenderIt-system, and RecipientAuthority.
- Serviceplatformen identifies the SFTPUsername the file should be delivered to based on the 4 required fields: Infref, SenderAuthority, SenderIt-system, and RecipientAuthority.
- If a SFTPUsername to deliver the file to cannot be identified based on the provided information an error is returned to the sender in a technical receipt.
- If the field RecipientIt-system was provided in the SFTPDynamicRoutingInfo structure an error is returned in the technical receipt indicated that the provided in infref uses *implicit* routing.

*Explicit* routing is done in the following manner:

- For *explicit* routing rules stored in the database the following fields must be filled out: Infref, SenderAuthority, SenderIt-system, RecipientAuthority and RecipientIt-system. SFTPUsername must be empty.
- The sender must provide the following fields in the SFTPDynamicRoutingInfo structure: Infref, SenderAuthority, SenderIt-system, RecipientIt-system and RecipientAuthority.
- Serviceplatformen identifies the SFTPUsername the file should be delivered to based on the 5 required fields: Infref, SenderAuthority, SenderIt-system, RecipientIt-system and RecipientAuthority.
- If a routing rule cannot be identified based on the provided information an error is returned to the sender in a technical receipt.
- If the field RecipientIt-system was not provided in the SFTPDynamicRoutingInfo structure an error is returned in the technical receipt indicated that the provided in infref uses *explicit* routing and that the fields is required.

A database table has been created that contains a mapping between infref values and if *implicit* or *explicit* routing should be used. So for each file transfer using dynamic routing a lookup is made in the database to determine how the file should be routed.

# 5 Trigger object validation

In both UC01 and UC02 the addressing of a file is delivered to the SFTP Service in a trigger object. The validation process of the trigger object is similar for both use cases, and done using the SFTPTriggerProcessorBean.

A trigger object is validated through the following steps.

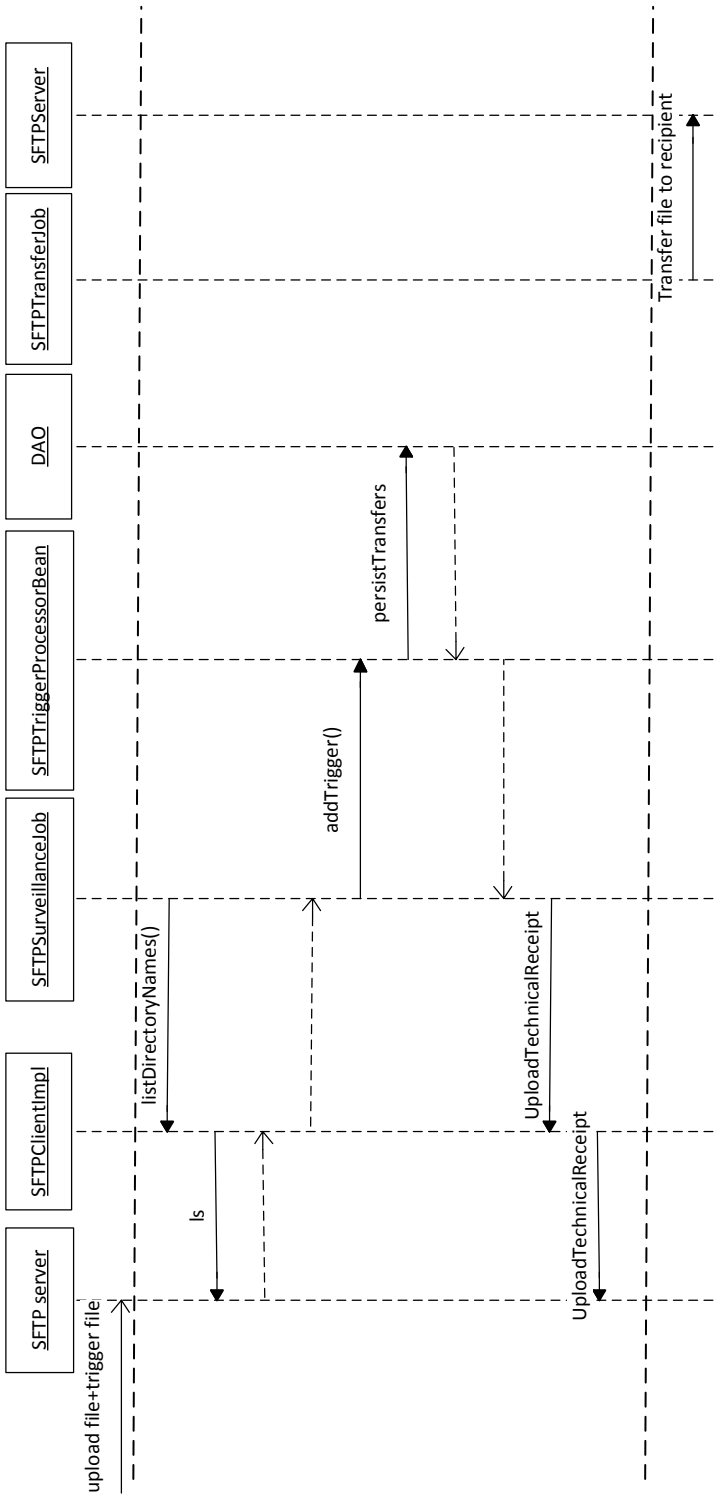
1. It is validated that the provided trigger object contains valid xml.

2. It is checked that the specified sender username is correct
3. It is checked that the specified file exists on the SFTP Server, and that it has the correct size
4. If the recipient is a virtual user then trigger file will pass dynamic routing validation. This validation consists of the following:
   a. The trigger file should contain the SFTPDynamicInfoStructure inside FileContentDescriptor structure in the trigger object.
   b. The trigger file should contain only one recipient, which is the virtual user.
   c. The InfRef value in the SFTPDynamicInfoStructure is present and not empty.
   d. The SenderAuthority value in the SFTPDynamicInfoStructure is present and not empty.
   e. The RecipientAuthority value in the SFTPDynamicInfoStructure is present and not empty.
   f. The SenderIt-system value in the SFTPDynamicRoutingInfoStructure is present and not empty.
   g. Verify there exists a dynamic routing rule in the database using the approach described in Section 4.
   h. Verify all extra validations that might exist for the matched routing rule. Right now only validations related to maximum file size are supported.
   i. If all validations pass a record is added into the table sftp_dynamic_routing_transfers with appropriate status, recipient name, trigger content, sender name, routing username.

      If one of the rule will be failed, then a TriggerProcessorException will be thrown with appropriate error message.
5. It is validated that each of the specified recipients can receive the file. This involves checking that they have the same use case as the sender, they do not have too many files in their IN folder, or that they do not have a file with the same name present in their IN folder.
6. If all the previous steps are successful, then the file to be sent is locked on the SFTP server. In this context "locked" means that the senders write rights are removed from the file. Based on the result of the validation a technical receipt is generated, and a transfer task is persisted into the db.

# 6 UC01 flow

The flow for file transfer using UC01 is illustrated in the following figure:




It should be noted that the figure has been simplified for the sake of clarity. The dotted lines indicates that the following steps of the flow happens asynchronously from the previous step.

1. The sending system uploads the file to be transferred along with a triggerfile to their out folder on the SFTP Server.

2. The SFTPSurveillanceJob periodically checks all OUT folders on the SFTP server for triggerfiles using the SFTPClientImpl class to get a list of the complete content of a folder.

3. The SFTPClientImpl class runs a "ls" command on the specified out folders and returns the content to the SFTPSurveillanceJob

4. The SFTPSurveillanceJob checks the content of each folder returned by the SFTPClientImpl. For each trigger file found, it sends it to the SFTPTriggerProcessorBean for processing.

5. The SFTPTriggerProcessorBean validates the triggerfile and generates a technical receipt that is returned to the SFTPSurveillanceJob. The SFTPTriggerProcessorBean furthermore performs the necessary operations that must be performed based on the result of the validation, e.g. locking the file to be transferred, and adding a transfer task to the "sftp_transfers" table.

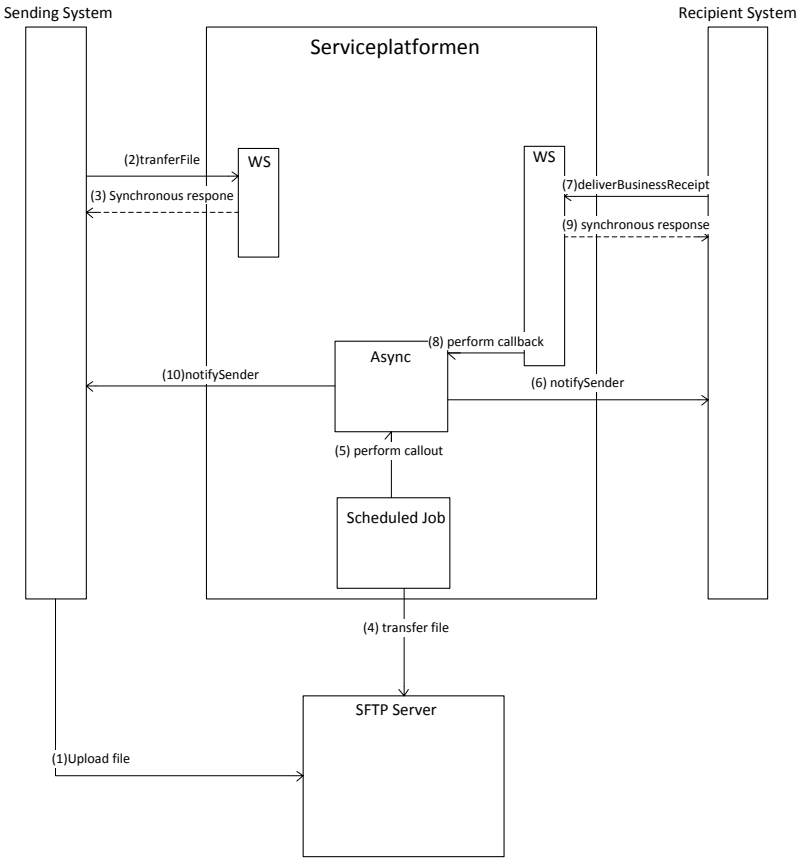
6. The SFTPSurveillanceJob uploads the technical receipt to the IN folder of the sending system

7. The SFTPTransferJob periodically checks for new files to be transferred. If it detects a file that must be transferred, it transfers the file to the recipient systems IN folder, along with a recipient notification.

When the SFTPSurveillanceJob parses a trigger file, all recipients will be checked. If at least one of them is a virtual user, then the job will check the SFTP Dynamic Routing Info data using the rules described in section 4. When the SFTP Service encounters trigger files with a "virtual" recipient specified, Serviceplatformen will use dybanic routing to attempt to route the file.

# 7 UC02 flow

For the UC02 flow communication between the sending- and recipient system and the Serviceplatform is done through webservice calls. A simplified version of the flow for UC02 is illustrated in the following figure:




1. The sending system uploads the file it wishes to send to the SFTP server
2. The Sending system calls the *transferFile* method on the SFTP webservice with a request for transfer of a file
3. The Serviceplatform validates the file transfer request, if it is valid a transfer task is created and a synchronous response is returned to the sending system
4. A scheduled job detects the newly created transfer task, and transfers the file to the recipient system on the SFTP server
5. If the file successfully transferred to the recipient, then an asynchronous task for performing a callout to the recipient system is created.
6. The retry framework attempts to call the recipient system with a notification that a file has been transferred to the system.
7. The recipient system calls the *deliverBusinessReceipt* method on the SFTP webservice with a business receipt for the transferred file.
8. An asynchronous task for performing a callback to the sending system is created
9. A synchronous response is returned to the recipient system
10. The Serviceplatform performs a callback to the sending systems.

The flow for transferring files using UC02 is separated into three different sub flows, which are handled independently within the Serviceplatform.

These three flows are:

- The initial call to the transferFile method on the SFTP webservice
- Transfering of the file and callout to the receiving system
- The recipient system calls the deliverBusinessReceipt and a callback is made to the sending system.

For UC02 the trigger object is delivered to the Serviceplatform through a webservice call to the SFTP webservice. Furthermore the recipient system also delivers a business receipt by calling the SFTP webservice.

## 7.1 SFTP webservice

The SFTP webservice exposes 2 methods:

### 7.1.1 transferFile

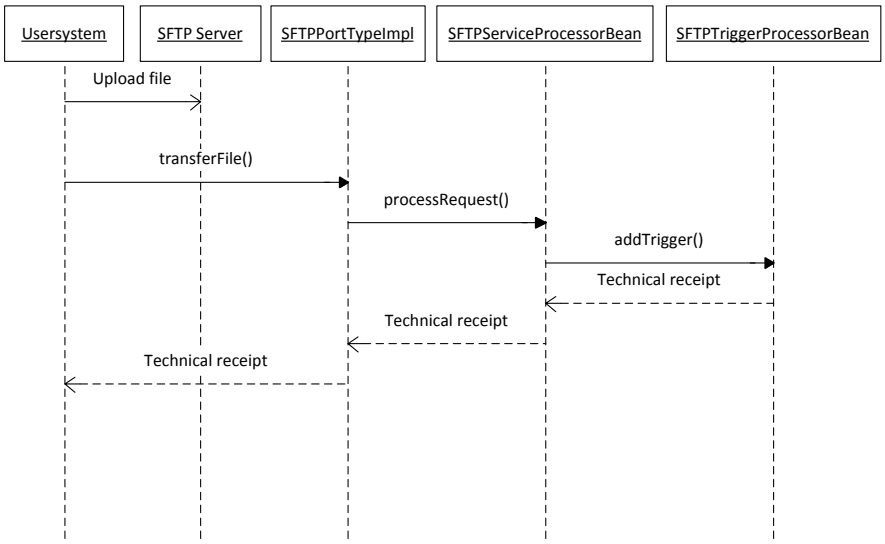
This webservice is called by the sending system with a trigger object as input. The trigger object is validated and a technical receipt is returned to the caller.

### 7.1.2 deliverBusinessReceipt

This webservice is called by the recipient system with a business receipt indicating whether or not the recipient system accepted the file. The business receipt is validated and if it contains errors a response with an error message is returned to caller.

## 7.2 TransferFile flow

This flow covers the processing of the initial webservice call made by the sending system to the SFTP webservice with a trigger object. The trigger object is validated, and a technical receipt is generated and returned to the caller.

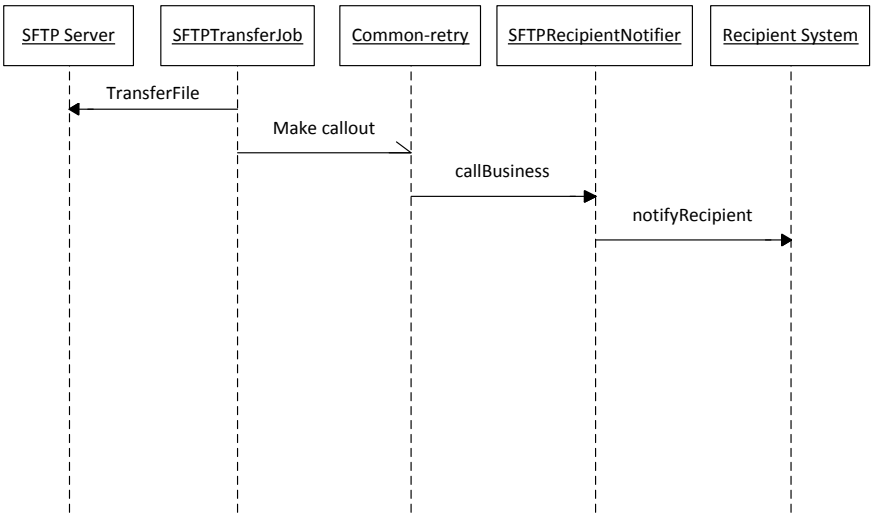



1. The sending system uploads the file to be sent to the SFTP Server

2. The sending system calls the transferFile method on the SFTP Webservice with a triggerobject.
3. The trigger object is send to the SFTPServiceProcessorBean for processing
4. The SFTPServiceProcessorBean forwards the trigger object to the SFTPTriggerProcessorBean
5. The SFTPTriggerProcessorBean validates the trigger object, and generates a technical receipt.
6. The technical receipt is returned synchronously to the sending system through the SFTPServiceProcessorBean and the SFTPWebservice.

### 7.3 Callout flow

The callout flow is initiated when the SFTPTransferJob transfers a file to a recipient that is using UC02.



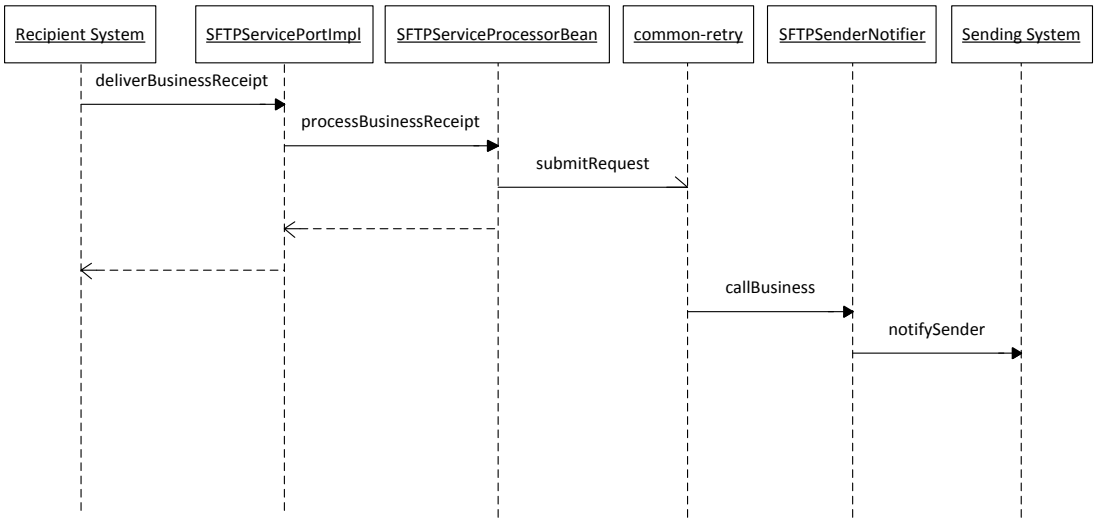

1. The SFTPTransferJob detects that a file needs to be transferred and transfers the file to the recipients IN folder.

2. If the recipient sending and recipient system uses UC02, then a task is added to the common-retry framework for making a callout to the recipient system. This task is processed asynchronously.

3. The SFTPRecipientNotifier attempts to call the recipient system with a recipient notification object.

### 7.4 Callback flow

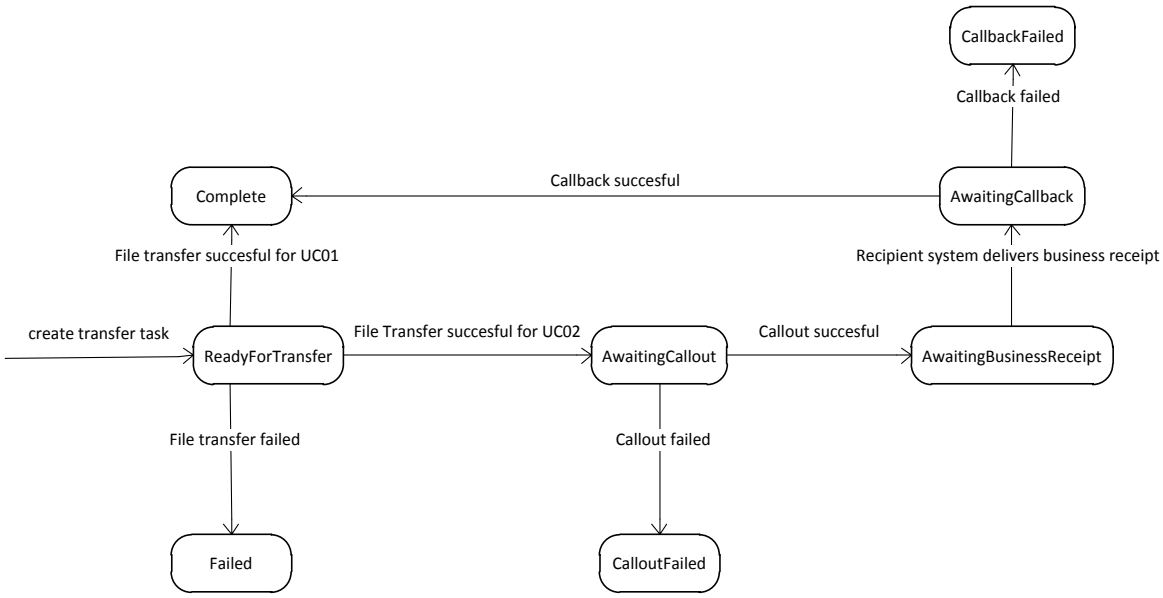
The callback flow handles the final part of UC02. This involves the sending system calling the SFTPServicePortImpl with a business receipt. The SFTP Service validates that the delivered business receipt is associated with a file transfer. If this is the case a callout is done using the retry framework in the Serviceplatform.




1. The recipient system calls the SFTP webservice with a business receipt.
2. The SFTPServicePortImpl forwards the business receipt to the SFTPServiceProcessorBean for handling.
3. The SFTPServiceProcessorBean validates that the file transfer the recipient system is delivering a business receipt for exists. If the file transfer exists then a callout task is send to the retry framework, otherwise a response with an error message is returned to the sending system, and no more is done.
4. If a task was sent to the retry framework, then the retry framework asynchronously makes a call to the callBusiness method on the SFTPSenderNotifier.
5. The SFTPSenderNotifier makes a call to the sending system with a business receipt, containing information that the recipient system has accepted or rejected the transferred file.

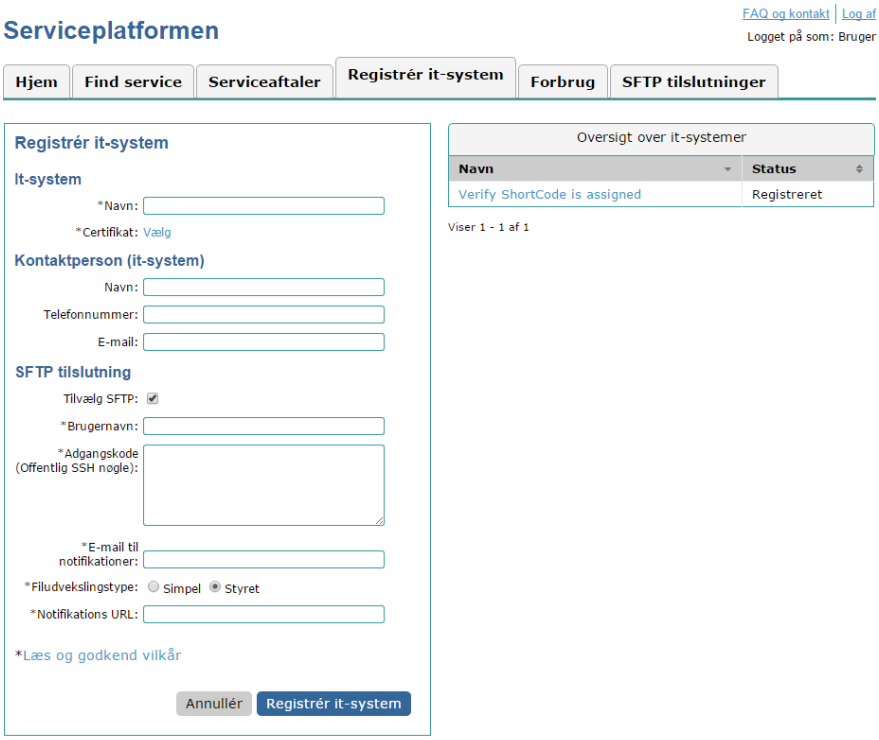
# 8 State of a transfer task

The states a transfer task can be in is different for UC01 and UC02. The following figure gives an overview of the states a transfer can be in, and what brings the transfer task to the given state.




# 9 SFTP user administration

Since access to the SFTP Service is granted through a connection agreement, the administration module has been updated. The new GUI for creating a connection agreement is illustrated below:




When a connectionagreement is created with the SFTP option added, then a user is created on the LDAP server with the username provided in the administrationmodule. Furthermore an IN and an OUT folder is created on

the SFTP server, and the provided public key is uploaded to the SFTP server, such that the user is able to log into the SFTP server using their ssh key.

## 10 Database model

Five new database tables have been created for the SFTP Service: *"adm_connectionagreement_sftp"*, *"usersystem_sftp"*, *"sftp_transaction"*, *"sftp_transfer"* and *"sftplog"*, *"UserSystemSFTPFilter"*, *"SFTPWatchlist"*.

The *"sftp_transaction"* table contains a row for each trigger file that has been processed by the SFTP Service. The overall status of the transfers specified in a trigger file is maintained in this table.

The *"sftp_transfer"* table contains a row for each file transfer performed on the SFTP server. The status of the transfer task is maintained in this table.

The *"adm_connectionagreement_sftp"* contains SFTP information related to a connectionagreement.

The *"usersystemsftp"* table contains SFTP information related to the a user system.

The *"sftplog"* table contains an entry for each operation performed on the SFTP server that must be logged.

The *"UserSystemSFTPFilter"* contains information about filters that is used to determine which file should be monitored.

The *"SFTPWatchlist"* contains list of file that should be monitored and cleaned up.

The "sftp_dynamic_routing_rules" table contains the routing rules that has been setup for dynamic routing.

The "sftp_dynamic_routing_transfers" used to log information on files transferred with SFTP Dynamic Routing.

## 11 Logging on the SFTP server

As part of the required reporting for the SFTP server a scheduled job has been created that daily parses the raw log file from the SFTP server. This log file contains entries for every operation performed on the SFTP server. The scheduled job parses this log file and adds an entry to the database for every time a user: logged in, logged out, uploaded a file, and downloaded a file. When a user uploads and downloads a file, the size of the file, along with the time it took to upload a file is logged. Furthermore every action performed by the SFTP service using its "master" user is logged in similar detail. This includes e.g. login, upload of technical receipts and transferring of files to recipients.

## 12 Cleanup on the SFTP server

A scheduled job has been created that periodically scans all configured in usersystemsftp table (in and/or out) folders on the SFTP server.

If it finds a file that has been on the SFTP server for more than 30 days without being modified, it will send a notification to the user system that owns the file, with a message stating that the file will be deleted in 10 days.

It will only send one mail to each user system per day, meaning that if a user system has 10 files that has been on the SFTP server in an unmodified state for 30 days, it will receive one mail listing the 10 files that will be deleted in the future.

A scheduled job daily goes through a watchlist. Files on the watchlist will not be deleted. By default the folders of usersystems are not being monitored and files in the folders will therefore added to the watchlist

Files are added to the watchlist if they match a certain configured filter and removed if they are manually deleted form the folder. If a file on the watchlist has not been deleted within a configured number of days after being uploaded a notification mail is send to the usersystem/manual treatment task is created. The number of days before a notification is send to the usersystem, after which administaters of the usersystem are expected to take action, and the number of days before a manual treatment task is created can be configured in a SFTP that matches a specific file.

SYSTEMATIC


**Denmark**

Søren Frichs Vej 39
8000 Aarhus C, DK
Tel.: +45 8943 2000
more.info@systematic.com

Landgreven 3, 2.sal
1301 Copenhagen K, DK

**Australia**

Tower A, Level 5,7 London Circuit
Canberra ACT 2600, AU
Tel.: +61 (0)2 6169 4088
more.info.au@systematic.com

**Finland**

Finlaysoninkuja 19
33210 Tampere, FI
Tel.: +358 207 463 870
more.info.fi@systematic.com

**France**

5 Place de la Bastille
75004 Paris, FR
Tel.: +45 8943 2000
(HQ in Denmark)
more.info.fr@systematic.com

**Germany**

Im Zollhafen 24
50678 Köln, DE
Tel.: +49 221 650 783 71
more.info.de@systematic.com

**New Zealand**

15 Level, 171 Featherston Street
Wellington 6011, NZ
Tel.: +64 04 894 8571
more.info.nz@systematic.com

**Singapore**

15 Hoe Chiang Road
#12-02 Tower Fifteen
Singapore 089316
Tel.: +65 6653 7492
more.info.sg@systematic.com

**Sweden**

Ostermalmstorg 1, 4th Floor
Stockholm 114 42, SE
Tel.: +46 770 770109
more.info.se@systematic.com

**UAE**

World Trade Centre, Level 17, Suite 56
Abu Dhabi, UAE
Tel.: +971 2 654 4675
Fax: +45 8943 2020
more.info.uae@systematic.com

**United Kingdom**

Meadow Gate, Farnborough Airport
Farnborough, Hampshire
GU14 6XA, UK
Tel.: +44 1276 675533
more.info.uk@systematic.com

**United States of America**

5875 Trinity Parkway, Suite 130
Centreville, Virginia 20120-1971, USA
Tel.: +1 703 385 7522
more.info.us@systematic.com