

Requirements document for Standing Orders in GnuCash

Table of contents

Document Maintenance.....	1
Versions.....	1
Feature overview.....	2
Current status.....	2
target concept.....	2
Requirements.....	2
Functional requirements.....	2
Non-functional requirements.....	5
Constraints.....	5

Document Maintenance

For Purpose of contact see following information:

Bugreference	642123
Abstract	This document should give a mainly user orientated view to the wished feature of standing orders via HBCI in GnuCash. It will describe the most important requirements.

Versions

Version	Date	Author	Description
0.1	2011-02-11	Simon Harhues	Initial version

Feature overview

Current status

GnuCash uses the program (/library collection) AqBanking to make online banking via HBCI possible. You can transfer from account to account an normal accounting entry. But with GnuCash it is not possible to have standing orders, like for renting a flat. However this feature is already build in into AqBanking. If you want to create, show, edit or delete standing orders you have to use a different software than GnuCash at the moment.

target concept

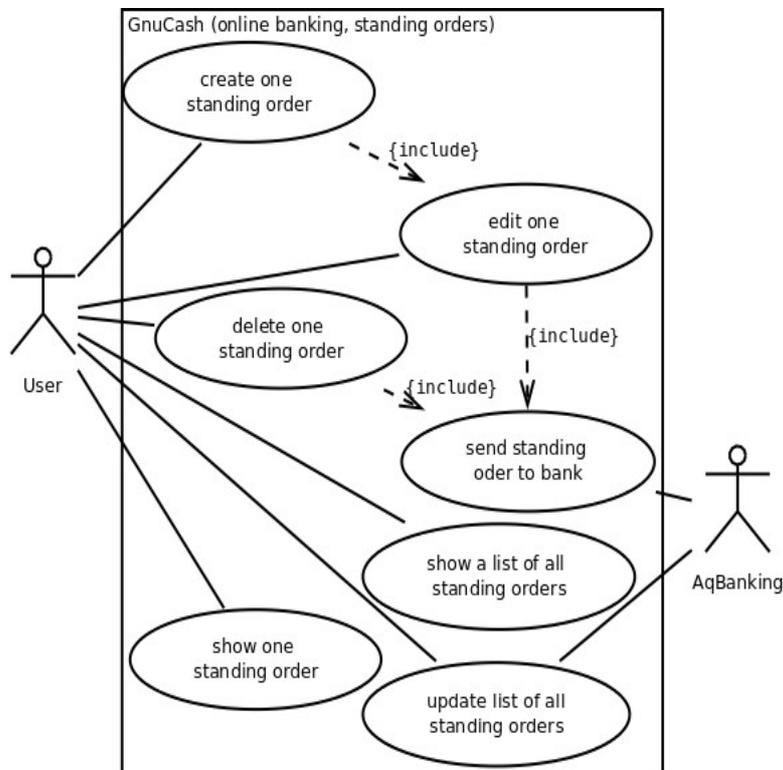
Eventually you will be able to manage your standing orders via HBCI with GnuCash. It should be possible to create, edit and delete standing orders. This feature is mainly useful for German users, as HBCI is a German standard. The requirements to fulfil are explained in detail in the next section.

Requirements

Functional requirements

Use cases

The following UML use case diagram shows the necessary use cases to handle standing orders.



The actor “user” is a end user of GnuCash. “AqBanking” is the external program which GnuCash uses to establish the HBCI connection.

Use case 1: Create a standing order

Actor	User
Goal	Create a new standing order and send it to the bank.
Trigger	Triggered by the user. (out of dialogue from use case 5)
Prerequisite	Configuration of HBCI is done and working.
Regular application flow	<ol style="list-style-type: none">1. Open form to create a new standing order2. fill out the form3. save & send it to bank
Expansions	To send it to the bank use case 4 is used.
Errors	Error handling is done like with normal orders via HBCI. Error handling is done like with normal orders via HBCI.

Use case 2: Edit a standing order

Actor	User
Goal	Edit an existing standing order and send the updated order to the bank
Trigger	Triggered by the user out of use case 5.
Prerequisite	Configuration of HBCI is done and working. There are yet standing orders in the program.
Regular application flow	<ol style="list-style-type: none">1. Execute use case 62. Edit some fields3. save & send to bank
Expansions	To send it to the bank use case 4 is used.
Errors	Error handling is done like with normal orders via HBCI. Only if the altering at the bank succeeded, the updated data can be shown in GnuCash. Otherwise we have to roll back to show the old data.

Use case 3: delete a standing order

Actor	User
Goal	Delete an existing standing order at the bank
Trigger	Triggered by the user. (out of dialogue from use case 5)
Prerequisite	Configuration of HBCI is done and working. There are yet standing orders in the program.
Regular application flow	<ol style="list-style-type: none">1. Select an order from the list of all standing orders2. delete it (and send the deletion-request to the bank)
Expansions	To send it to the bank use case 4 is used.
Errors	Error handling is done like with normal orders via HBCI. In case that the order can could not be deleted at the bank, it will remain in GnuCash.

Use case 4: send standing order to bank

Actor	AqBanking
Goal	Send the request to create, alter or delete an ongoing order
Trigger	Triggered by use case 1, 2 or 3.
Prerequisite	Configuration of HBCI is done and working. There are yet standing orders in the program.
Regular application flow	This use case is transparent to the user except if e.g. AqBanking is prompting for a password. <ol style="list-style-type: none"> 1. Call AqBanking with the wanted action 2. give feedback to the calling use case
Expansions	
Errors	Error handling is done like with normal orders via HBCI. The calling use case will get feedback if the transaction succeeded.

Use case 5: show a list of all standing orders

Actor	User
Goal	Give the user an overview of all of his standing orders.
Trigger	Triggered by the user.
Prerequisite	Configuration of HBCI is done and working. There are yet standing orders in the program.
Regular application flow	The user should see following fields: <ul style="list-style-type: none"> • labelling of standing order • recurring date of transferring the money • amount
Expansions	From this dialogue out all other use cases except 2 and 4 are started
Errors	

Use case 6: show one standing order

Actor	User
Goal	See the details of one standing order
Trigger	Triggered by the user.
Prerequisite	Configuration of HBCI is done and working. There are yet standing orders in the program.
Regular application flow	<ol style="list-style-type: none"> 1. Select one order out of the list of all standing orders 2. open dialogue with detailed view of standing order
Expansions	The user can edit fields which leads to the use case 2.
Errors	

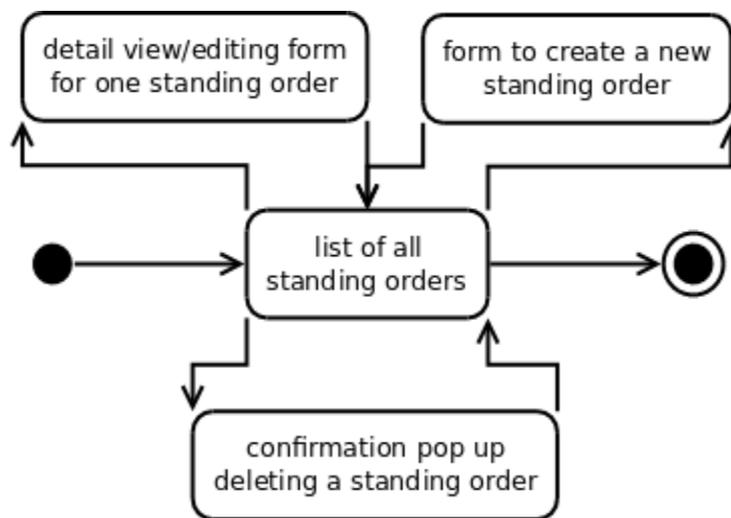
Use case 7: update list of all standing orders

Actor	User
Goal	See the details of one standing order
Trigger	Triggered by the user.
Prerequisite	Configuration of HBCI is done and working. There are yet standing orders in the program.
Regular application flow	3. Select one order out of the list of all standing orders 4. open dialogue with detailed view of standing order
Expansions	The user can edit fields which leads to the use case 2.
Errors	Show a dialogue to user that Update failed.

Non-functional requirements

Dialogues

Following UML state diagram shows the proposed dialogue chain.



The form to create a new dialogue and to view/edit an existing dialogue should look basically the same. The new dialogues should also have the same appearance like the other dialogues dealing with (HBCI) orders.

Constraints

The implementation of the HBCI protocol is not part of this feature, as AqBanking is providing this functionality.

The first implementation should only cover the functionality to do basic things like creating, editing and deleting standing orders. Maybe it is even possible to abandon the possibility of editing an order in the first turn, as it is possible to reach the same goal via deleting and recreating a standing order. In further versions the features can be get improved and completed.