# Use Case Details - Structured

Final versions of the requirements documents used for walk-throughs with the main stakeholders.

IDs can be anything from consecutive numbers to meaningful acronyms about what kind of use case this is and what system it is a part of.

## UC1 Withdraw Cash

### General info

#### Description:

The actor uses bank card to access menu and choose withdraw cash. They request cash amount and ATM dispenses it with a receipt.

#### Actors: Customer

#### Supporting roles/systems: Bank

#### Type: System

#### Pre-conditions:

Rules for beginning this use case: state of system prevents usage, must be testable. Or in a business use case, this must be the current state that has met a goal through another use case that this use case can now follow.

* System has more than or equal to ATM\*/ maximum withdrawal per transaction.

### Scope info

#### Level: Goal

#### Includes:

The use cases that are extracted out of this use case and given a special name, so they can be reused. They are required to be a part of this use case. This use case can be considered a grouped use case if it includes one of a group of partial goal use cases.

#### Included in:

The use case(s) that uses this one as a necessary part of it.

#### Use cases grouped by this ID:

If it doesn’t have an included group above, then it will be a category for several use cases.

#### Grouped by:

The group that has others like this one.

### Tracking info

#### Author: BA class of 5/10/19

#### Date created: 5/10/19

#### Date revised: 5/10/19

### Project info

#### Design constraints: ATM chassis Safeco XH23895-3

#### Priority: 3

#### Value to sponsor:

* Cost savings – overhead
* Customer satisfaction
* Competitive advantage

#### Sponsor:

Who is accountable for this use case being delivered successfully?

### Course of Events

The sequence of tasks in conversation format between actor and system. For best linking to other steps, start each number with a system task except for the trigger. Combine actor responses to system events when well. Rules are placed under the task unless they can be reused and then they are referenced and placed in a separate file.

The number of tasks per number is usually small and starts with the system or the role. Tasks are individually stated so the system/role can do multiple things but in separate sentences. The last task will prepare the state of the system so that this use case can be performed again. There will be no condition statements to branch into two separate use cases. There may be a section that is removed to a named partial use case and called an <<include>> to shorten the detailed use case.

References that can be used here to document anything other than a functional requirement are:

* **T#** - Text file item number – used for error messages and small prompts mostly on forms.
* **D#** - Design file item number – used for web pages, full screen menus, etc.
* **R#** - Report file item number – used for printed or on-screen report formats
* **\* -**  a Data Dictionary item – used to refer to data description and validation so that the detail doesn’t have to be specified here. Also bolded and colored is good.
* **Rule#** - Rule file item number – used to refer to process rules. Generally, this will follow one path only and another use case will pick up any other options. Some data validation rules find their way here but should be collected under the Data Dictionary. Unnumbered rules are not reusable and will just be defined below their functional requirement.

1. The use case starts when the actor inserts their card.
2. The system reads the card. The system prompts for PIN (SD#1). The actor enters PIN and press enter.
3. The system requests the bank to validate PIN. The bank confirms good PIN and returns **Accounts**\*. The system makes a log entry.
4. The system prompts for transaction type (SD#2). The actor selects Withdraw Cash.
5. The system prompts with associated account types (SD#3). The actor selects account.
6. The system prompts for amount to withdraw (SD#4). The actor enters amount to withdraw.
7. The system validates amount.
   1. **RULE**: Available funds – Amount requested is less than or equal to available funds.
   2. **RULE**: $20 increments – Amount requested must be in multiples of 20.
   3. **RULE**: Maximum withdrawal – Amount requested must be less than or equal to **ATM**\* maximum withdrawal amount.
   4. **RULE**: Maximum per business day – Amount request must be less than or equal to Account\* amount withdrawn in current business day.
   5. **RULE**: R# <name of rule in separate file>.
8. The system requests the bank to record the transaction. The bank confirms. The system logs the communication.
9. The system dispenses the cash. The system logs cash dispensing. The system deducts amount from ATM\*/cash on hand. The system prompts actor to take cash (SD#6). The actor takes the cash.
10. The system prints a receipt (RD#1). The system prompts actor to take receipt (SD#7). The actor takes the receipt.
11. The system prompts the actor for another transaction (SD#8). The actor declines.
12. The system returns the card. The system prompts actor to take card (SD#9). The actor takes the card.
13. The system displays welcome screen (SD#5).
    1. **RULE**: Time period lag for welcome screen – none.

### .Alternate flows (errors, exceptions)

The error flows are where a rule is broken, or something interrupts the normal “happy path” of the course of events. This often is during communication or other type of I/O.

* <Name> (<number(s) in course of events where this could occur>) – <description of what to do and where to return in the course of events>
* <Name> (<number(s) in course of events where this could occur>) – <description of what to do and where to return in the course of events>

### Alternate flows (extension points)

An exception to branching is when there is an optional <<extends>> of a partial use case. But the use case returns to where the option was taken.

* <Name> (<number(s) in course of events where this could occur>) – <description of what to do and where to return in the course of events>
* <Name> (<number(s) in course of events where this could occur>) – <description of what to do and where to return in the course of events>

### Post-conditions

What are your tests that tell you that this is a successful completion of a use case? It may be a repetition of one of the tasks or a file or document that has been completed. But there are minimal ways to complete the goal and there are very excellent ways to complete it. Put both down. Some people use MoSCow – must have, should have, could have instead of min and max conditions.

* <Name of goal>
  + <minimum set of qualifications>
  + <maximum set of what could be>

### Notes/ Special Requirements

Any kind of quality, capacity, security, availability, disaster recovery information that is because of this use case. Maybe you also have ideas about design, or people who need to be checked with, etc.

* Audio prompts for this use case is helpful. Hearing impaired.

# Screen Design

1. Enter PIN   
   

# Data dictionary

## Account

name – required, maximum number of characters 50, upper and lower case,

type – one of checking, savings, or maintenance.

current balance –

available balance –

amount withdrawn in current business day - 500

Card\*

Transaction\*

## ATM

maximum withdrawal per transaction - 300

## Card

ID

PIN – number is never recorded in a log entry

Account\*

## Transaction