# 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.

## S1 - Withdraw Cash

### General info

#### Description:

A customer wants to receive cash from their account. They use an ATM card, receive cash and a receipt.

#### Actors: Customer

#### Supporting roles/systems: Bank

#### Type: System

#### Pre-conditions: ATM must have $300 (RULE max withdrawal per 24 hour period) in it.

### Scope info

#### Level: Goal

#### Includes:

#### Grouped by: SGr1 Do Transaction (to do)

### Tracking info

#### Author: BA class of 9/27/18

#### Date created: 9/27/18

#### Date revised:

### Project info

#### Design constraints:

* ATM SafeChassis URE857d89
* Bank card with mag stripe

#### Priority: 9

#### Value to sponsor: Decrease overhead expenses and increase customer satisfaction by allowing transactions 24 hours a day.

#### Sponsor: Bank CEO

### 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
* **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
* **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 bank card.
2. The system asks for **Account** **PIN** (D1). The actor enters **Account** **PIN**.
3. The system requests Bank to validate **Account** **PIN**. The Bank returns **Account** information. The system logs the message.
4. The system displays transaction menu (D2). The actor selects Withdraw Cash option.
5. The system prompts for account selection (D4). The actor selects account.
6. The system prompts for withdrawal amount (D3). The actor enters amount to withdraw and confirms.
7. The system validates amount requested.
   1. **RULE sufficient funds** –**Account** **balance** must be greater or equal to amount requested.
   2. **RULE increments for dispensing** – Amount requested must be in increments of $20.
8. The system requests Bank to record transaction. The system records transaction. The Bank returns the new **Account** **balance.** The system logs the message.
9. The system dispenses cash, logs the cash dispensing, updates **ATM cash on hand**, prompts actor to take cash (D8), and starts a cash removal timer. The actor takes the cash.
   1. **RULE Cash removal timeout** – 30 seconds.
10. The system prompts whether to print a receipt or not (D7). The actor accepts.4
11. The system prints a receipt (D10). The actor takes receipt.
12. The system prompts for another transaction (D5). The actor declines.
13. The system prompts actor to remove card (D6) and starts a card removal timer. The actor removes card.
    1. **RULE Card removal timeout** – 30 seconds and card is retained if timer expires.
14. The system returns to idle screen (D9).

### Alternate flows (errors, exceptions) – bad interruptions

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.

* **Bank card not read** (1) -
* **Wrong type of card** (1) -
* **Insufficient funds** (7a) – Prompt actor for amount. Validate amount. Return to use case at step #?
* **Invalid PIN (3)** – Reenter PIN up to three times. After that, retain card.
* **Cash not taken (8a)** – cash is retained, cash on hand is updated.
* **Cash at threshold (8)** – if cash on hand is below $2000, send message to bank.
* **Cancel key pressed** (anytime before 6) – transaction is stopped, use case continues at #12.

### Alternate flows (extension points) – optional pause to do something

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.

* Buy stamps (?#)
* Check balance on screen (?#)

### 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.

* Bank has recorded transaction.
* Must have –
* Should have –
* Could have -

### 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.

* Fewer keystrokes are better for the user.