

Sample 3

Teller Application:

Use Case and Class Diagrams

Section Contents

Section Contents	1
Introduction	2
Overall System	3
Maintain Settings and Definitions	4
Maintain Settings.....	5
Maintain Definitions.....	6
Maintain Users	8
Perform System Utilities	8
Perform Teller Transactions - Summary	9
Perform Teller Utility Transactions.....	10
Perform Teller Daily Processing Transactions	10
Perform Operator and Cash Management Transactions	11
Perform Financial Transactions.....	11
Perform Monetary Financial Transactions	12
Perform Non-Monetary Financial Transactions	13
Perform Off-Lan Transactions.....	14
Common Transaction Features	15
Setting Class Diagrams.....	16
Definition Class Diagrams	17

Introduction

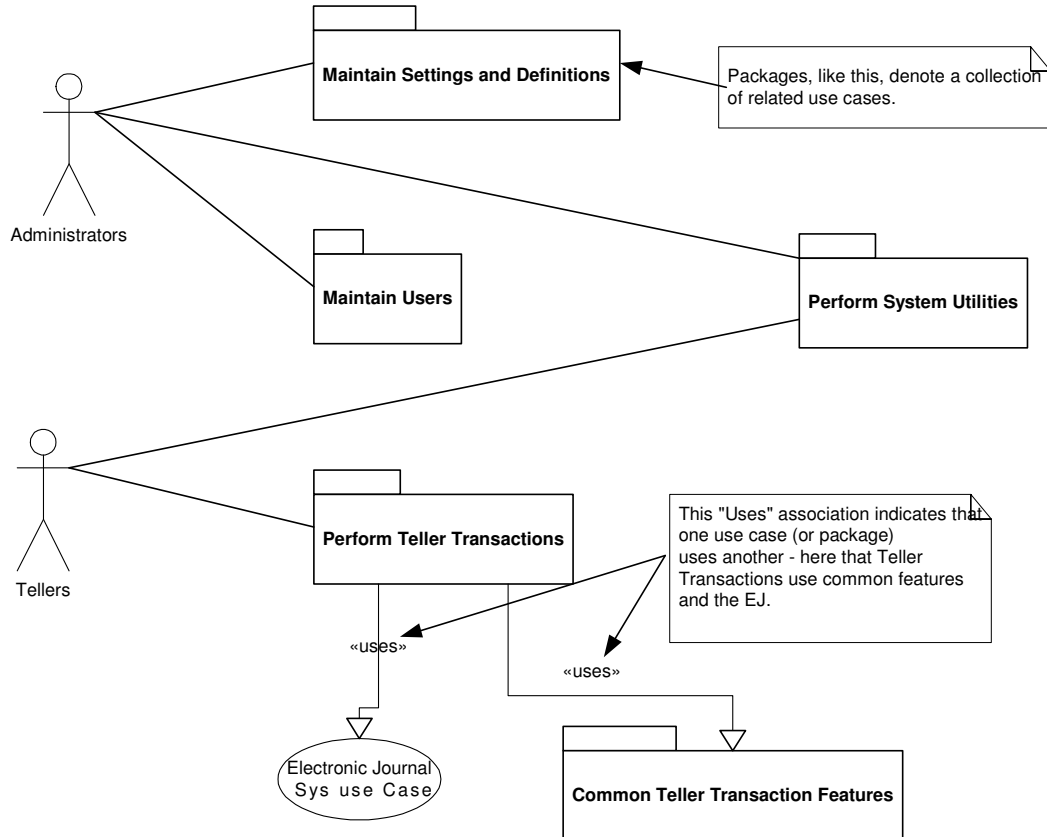
This is an analysis of use cases performed by the proposed ACME teller application. This document is one of a series of documents, beginning with the requirements, “ACME Software: Teller Requirements” and ending with detail specifications. Each document adds detail, taking the view of the system to progressively lower levels. The documents tend to be iterative. Higher-level documents are revised when lower-level documents reveal new aspects of the system.

This use-case analysis is presented as a set of UML (Unified Modeling Language) diagrams. The diagrams show...

1. **Use cases.** Represented by oval “bubbles”, use cases correspond generally to functions or transactions that are defined in ACME Portal Management System. Use cases are the most granular level of the system presented in this high-level, summary document. Knowing use cases helps understand the system requirements. This information is used eventually to identify the categories or classes of information used by the system and to create database schemas and coding specifications.
2. **Use case packages.** Represented by file folders, these are collections of use cases or of other packages. Packages correspond, generally, to the Portal Management System menu structures that are defined for each job authority. Packages provide a higher level of abstraction.
3. **Actors.** Represented by stick figure, these are the main roles performed by system users (human and programmatic). A person can perform one or more roles. Knowing who uses the system helps ensure that the design is appropriate for those users. It also helps to understand the function granularity required. In other words, if there were only one actor, the system could be packaged as a single function (which is the case with the ACME CRM system). If there are multiple actors (which is the case with Teller), the system needs to be broken into functions which can be packaged for the various users.

Note: In order to help understand the Maintain Settings and Definitions use cases and to provide a flavor of the next level of detail, Settings and Definitions Class diagrams have been included at the end of this document.

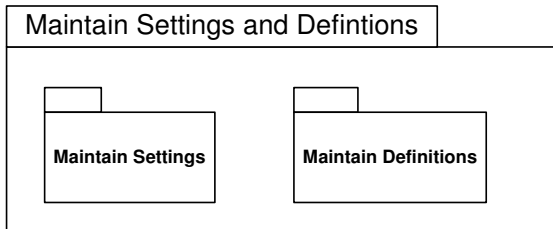
Overall System



Rules/Notes

1. Various levels of Administrator can be defined. Some administrators might be ACME analysts; others might be bank analysts. Some bank analysts might work at a bank's operation center; some might work at branches. Given the variety of administrators that might be defined, a high degree of granularity will be required in the functions used by these people – so that functions can be packaged in a variety of authorities.
2. Tellers include regular tellers, trainees, supervisors, etc. The distinctions between tellers will be based on job authorities and on privileges. Given the variety of teller roles, the functions and transactions used by these actors will also require a high degree of granularity.
3. System utilities are functions and features (such as Help and Message Functions) which are available to all users and employees - not based on authorities. The Teller Transaction package has its own set of utilities.
4. The Electronic Journal (EJ) is associated, at least potentially, with all teller transactions. Every teller transaction can create a record in the EJ. Part of the definition of each transaction (sub-function) will be a flag indicating whether the transaction creates an EJ record and what optional fields are included in the record.

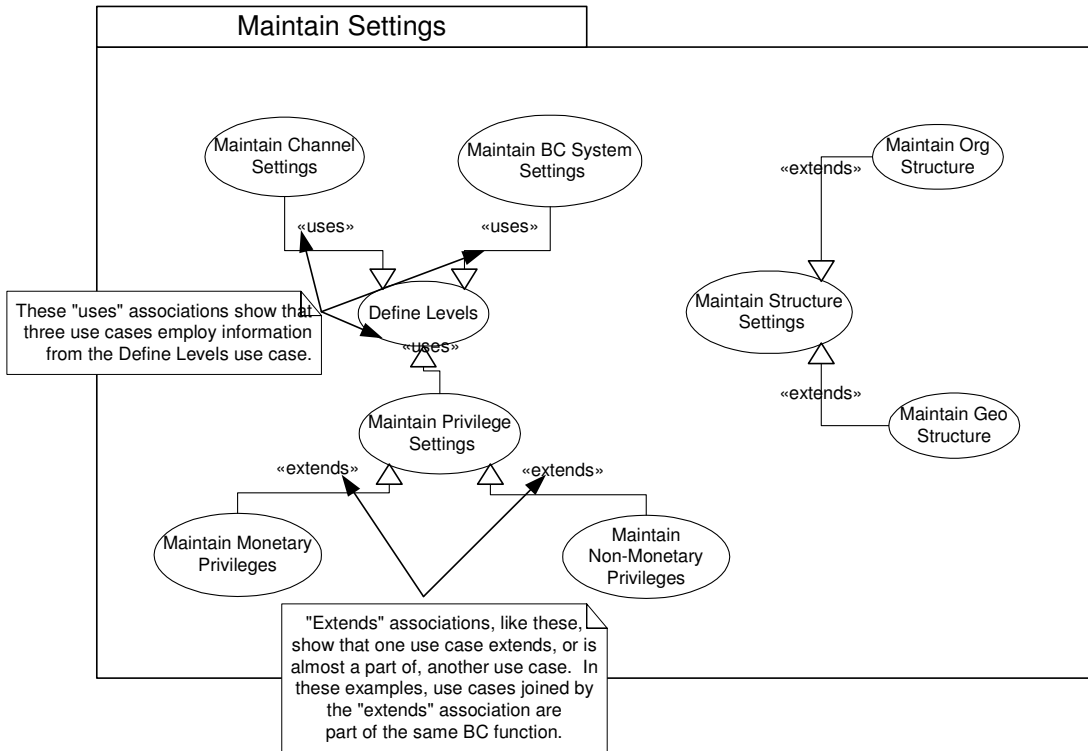
Maintain Settings and Definitions



Rules/Notes

1. In order to make the Teller functions and transactions useable for a wide variety of banks, the system will allow various parameters to be set. This will minimize the amount of re-coding that will be required for each bank. Instead of having programmers write code for each new installation, administrators will set parameters.
2. Parameters are used by functions and transactions. How functions and transactions use parameters is discussed under “Maintain Definitions”.
3. By convention, settings are single parameters. Examples are Page Timeout, Host Timeout, CTR Threshold, etc.
4. By convention, definitions are collective parameters. Examples are definitions for multiple functions, authorities, cash cans, fees, branches, workstations, cost centers, etc.
5. Some parameters might need to be defined then redefined – first as a default value then as a particular value for a given use. For example...
 - Default privileges are currently defined at the enterprise level then potentially redefined for each user
 - Default fees might need to be defined at an enterprise level then redefined for various geographic regions.
 - Default branch definition data might need to be supplied at the enterprise level then redefined at each branch.
 - Default language might need to be set an enterprise level and then reset at other levels. (A level could be “region” – in Canada for instance, or “branch”, in Miami or Los Angeles. Further, language might need to be set at an enterprise level, reset at a regional level and then reset again at the branch level.)
6. Because of the need to define parameters at various levels, “Define Levels” use cases have been included in both the Maintain Settings and the Maintain Definitions packages. These use cases define, for each parameter (setting and definition) which structure level sets the default value and which structure level(s) can override the default. If the bank specifies that a parameter can be defined at multiple levels, the values defined at higher levels become defaults for lower levels.
7. As noted previously, in order to help understand the Maintain Settings and Definitions use cases and to provide a flavor of the next level of detail, see the Settings and Definitions Class diagrams at the end of this document.

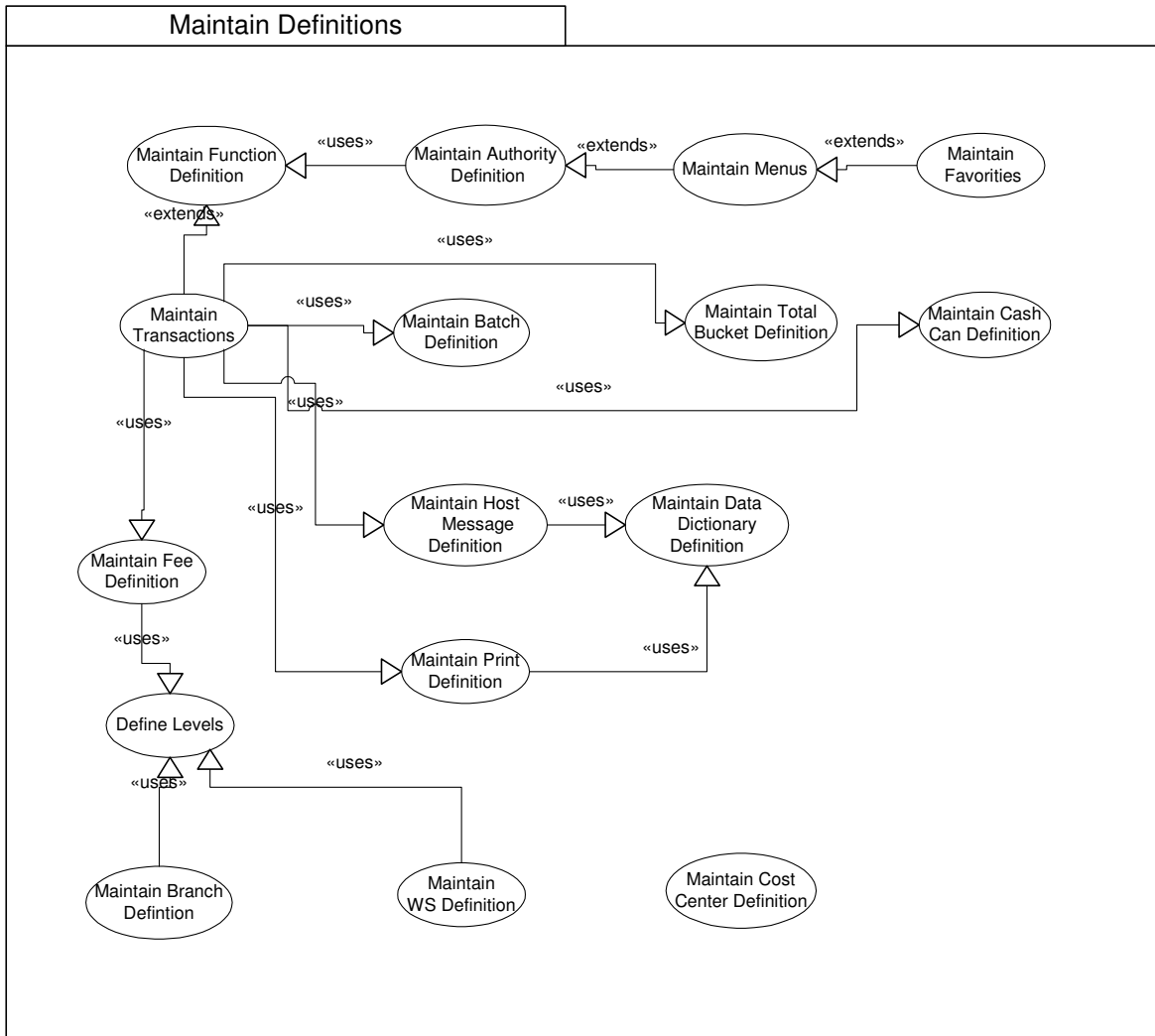
Maintain Settings



Rules/Notes

1. When defining functions/transaction definitions, the channel settings required by the function are specified. When channel settings are presented for maintenance, only the settings that are needed for the functions that have been installed will appear.
2. New uses for structures include branch settings and the Locator function (a System utility). Structures are used to define branches. The Locator function provides information about structure entity types (ATMs, Branches, Offices). The current Structures might be to be enhanced so it can capture additional information.

Maintain Definitions



Rules/Notes

1. A single function page can be associated with multiple transactions or sub-functions. For example, a teller Deposit function could include DDA, Savings and IRA transactions. A teller Withdrawal function could include On-U's DDA, Other Bank, Savings and IRA transaction. This feature is provided so that transactions and sub-functions can be conveniently grouped. Means (such as a dropdown) will be provided on the function page so that the end user can select the particular transaction that displays on a function page.
2. Many of the above definitions relate to transactions – they specify parameters which become attributes of transactions. For example, various host messages are defined and then a particular host messages is used by a transaction.
3. Ultimately, transaction-level code must call these definitions to get the right host message, print format, fee, etc. How this happens and where the transaction attributes are defined can happen at one of two levels. The calls could be hard-coded in the transaction code by the developer based on the specification for the transaction. Alternatively, the attributes could be included in the definition of the transaction and then automatically called as the transaction is processed. Because of the number and complexity of transaction attributes and because of the need to ensure that attributes are processed in a uniform way, the latter course is assumed.

4. Each transaction will have a number of attributes (whether stored in the database as part of the definition of the transaction and automatically called or hard-coded in the transaction page code). Transaction attributes include, but are not limited to:

- EJ flag
- Fields updated in EJ
- Host message used
- Print definition used
- Channel settings required
- Fees required
- Total buckets updated (and how)
- Batch counts updated
- Hooks performed
- Fields used in the transaction

Note: Each field will specify where the field displays on the page and the order in which it appears. If included as part of the transaction definition, this would allow transaction display to be “generated” based on this definition. Alternatively, these attributes could be hard-coded by the developer based on the specification for the transaction.

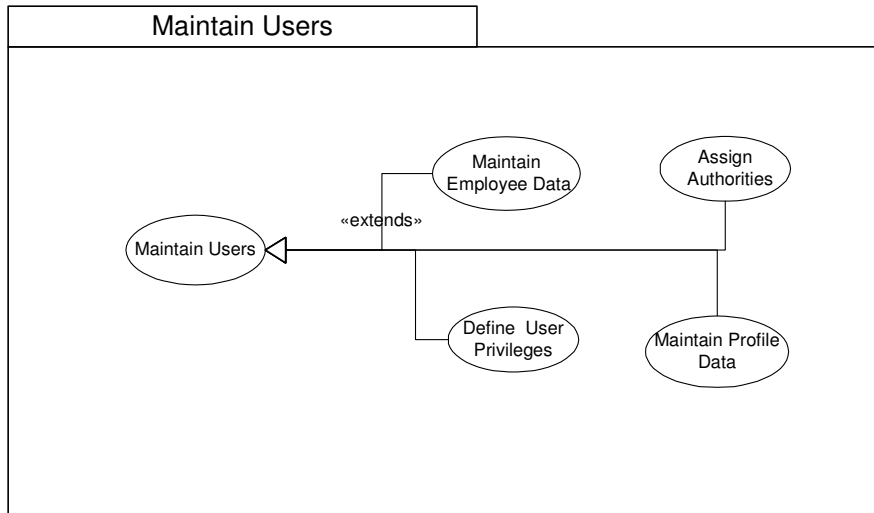
- Field prefill values
- Inter-field processing rules
- Function ID and name

Note: The ID will allow the transaction to be assigned to authorities – including the menus and the favorites associated with authorities. When the user selects the transaction from a menu or favorite the base function will appear, defaulted to the transaction selected. All the other transactions that have been associated with the function can still be selected. The function ID will allow the transaction to be selected from the Page ID field on the status bar.

5. If the transaction attributes are included as part of the transaction definition, then a transaction runtime component (or its functional equivalent) must be employed. (**Note:** If a runtime component is not included then each ASP developer would have to code the actions of the component for each transaction.) The runtime component would build and process transactions by reading transaction attributes and performs various steps in a particular order. These steps might include:

- Read fees required
- Determine fields used and build display
- Read field prefills and display
- Process data entry
- Process mag stripe input
- Execute field edits and inter-field processing rules
- Read host message and send it
- Receive host response
- Execute any rules associated with host response
- Display host response
- Read print message(s) and print
- Update total bucket(s)
- Update batch(es)
- Update EJ

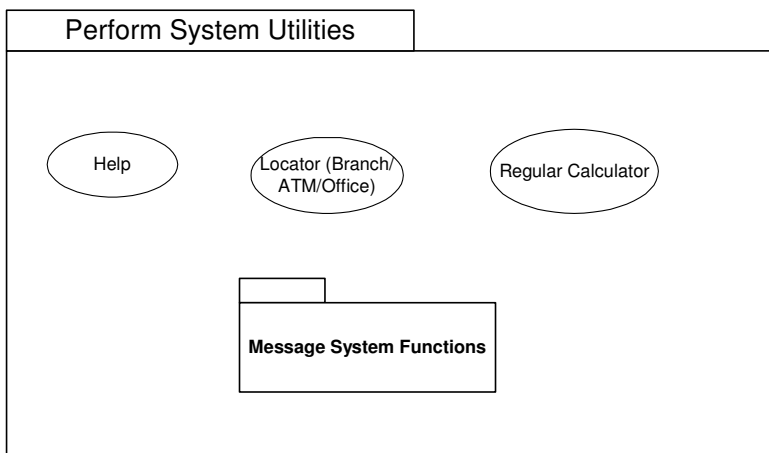
Maintain Users



Rules/Notes

1. No channel-specific user maintenance functions will be created. Channel-specific user attributes (such as floor limit and cash can assigned) can be set for all users.
2. Certain user requirements might derive from specific authorities. For example, all users of teller functions/transactions must be employees. These requirements will be checked when the user is assigned authorities. The requirements for the authority will derive from the functions/transactions which comprise the authority.

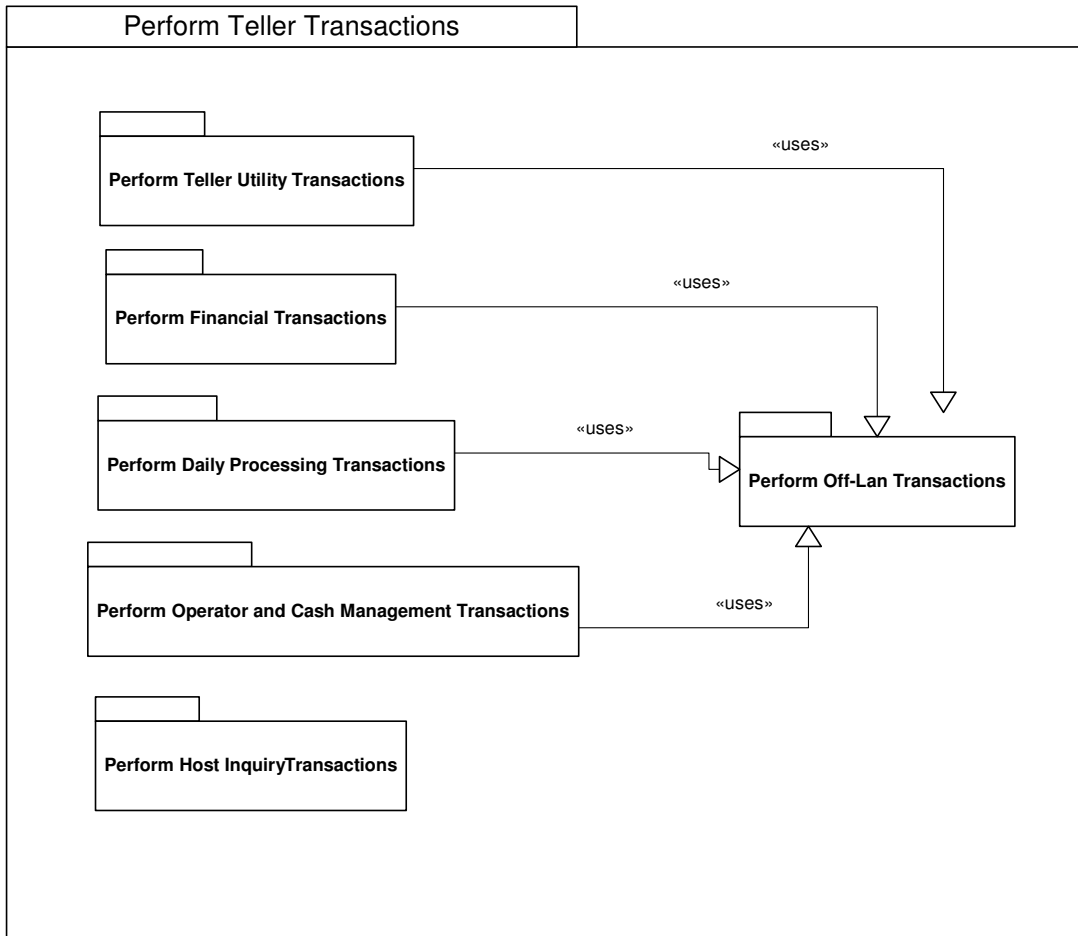
Perform System Utilities



Rules/Notes

1. As noted previously, these are functions and features that are available to all Portal System users – they are not based on authorities (although use of particular Message System functions might be restricted – perhaps by privileges).

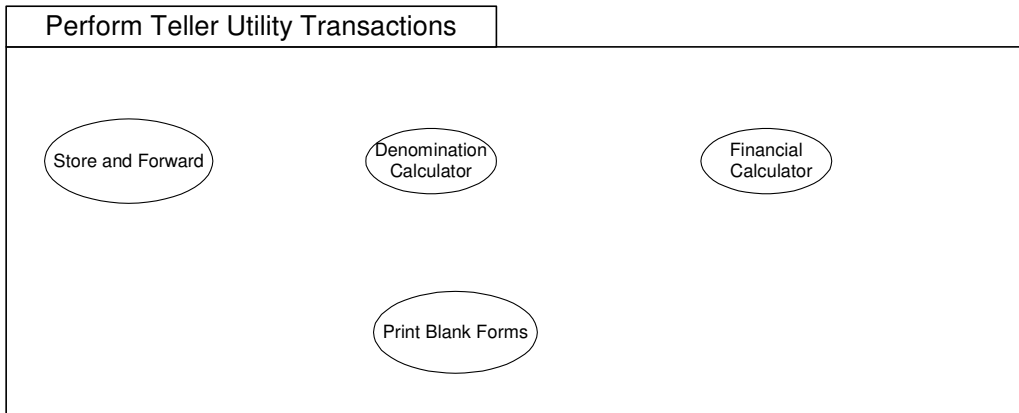
Perform Teller Transactions - Summary



Rules/Notes

1. The basic criteria for a teller transaction are that it can be defined to write a record to the electronic journal.

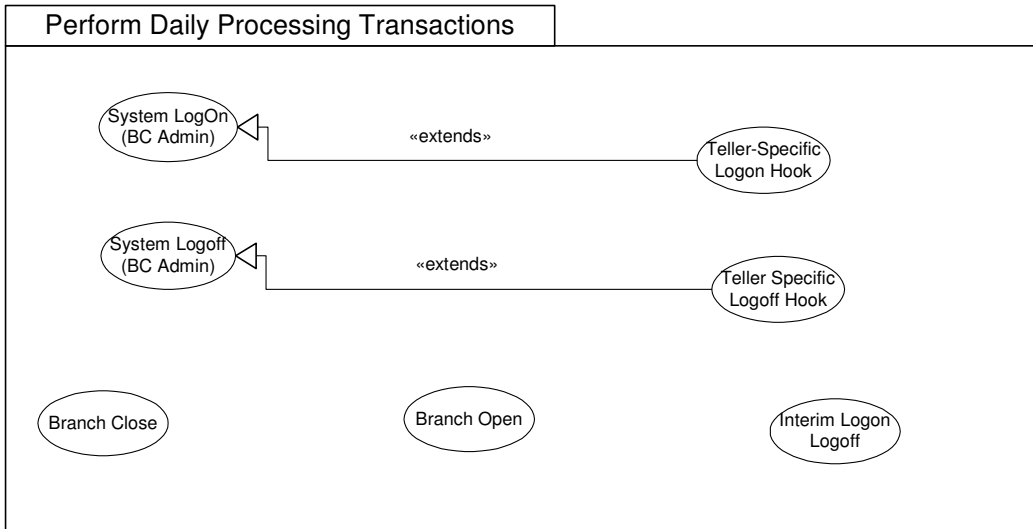
Perform Teller Utility Transactions



Rules/Notes

1. Teller utility transactions are functions and features that are only appropriate to the teller application. These transactions are available to all users who have an authority that contains a teller transaction.

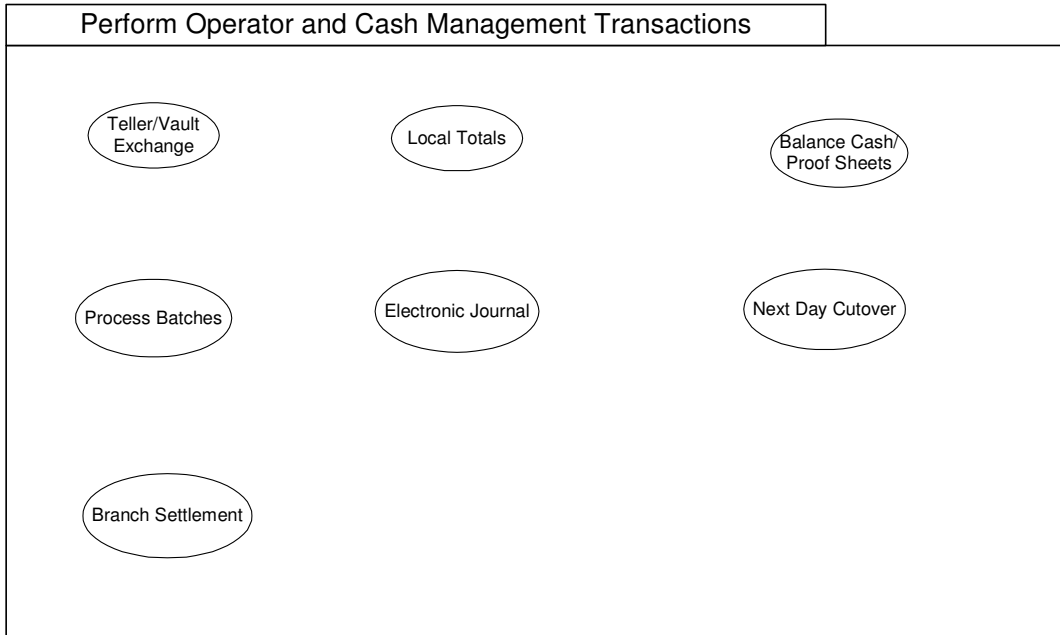
Perform Teller Daily Processing Transactions



Rules/Notes

1. Special logon/logoff requirements ("hooks") will be captured as part of the definition of these functions and will be passed on to authorities which include functions. During Portal System Logon, (which defaults to the last-used authority) any hooks associated with authority will get executed. The same thing happens if user switches to a new authority. Any logon hooks associated with that authority will be executed.

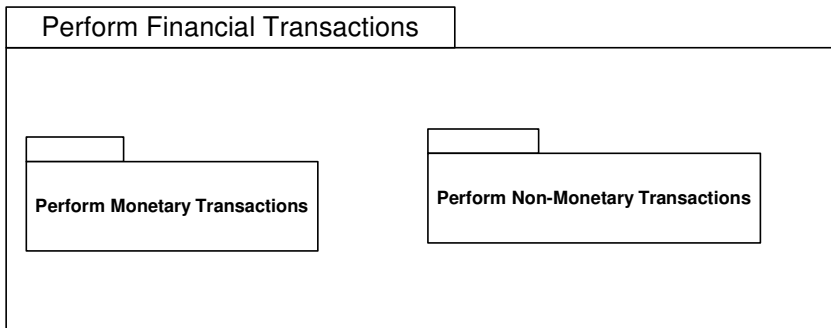
Perform Operator and Cash Management Transactions



Rules/Notes

1. The Electronic Journal use case “bubble” above represents EJ user activity – e.g., view, print, forward, EJ records.

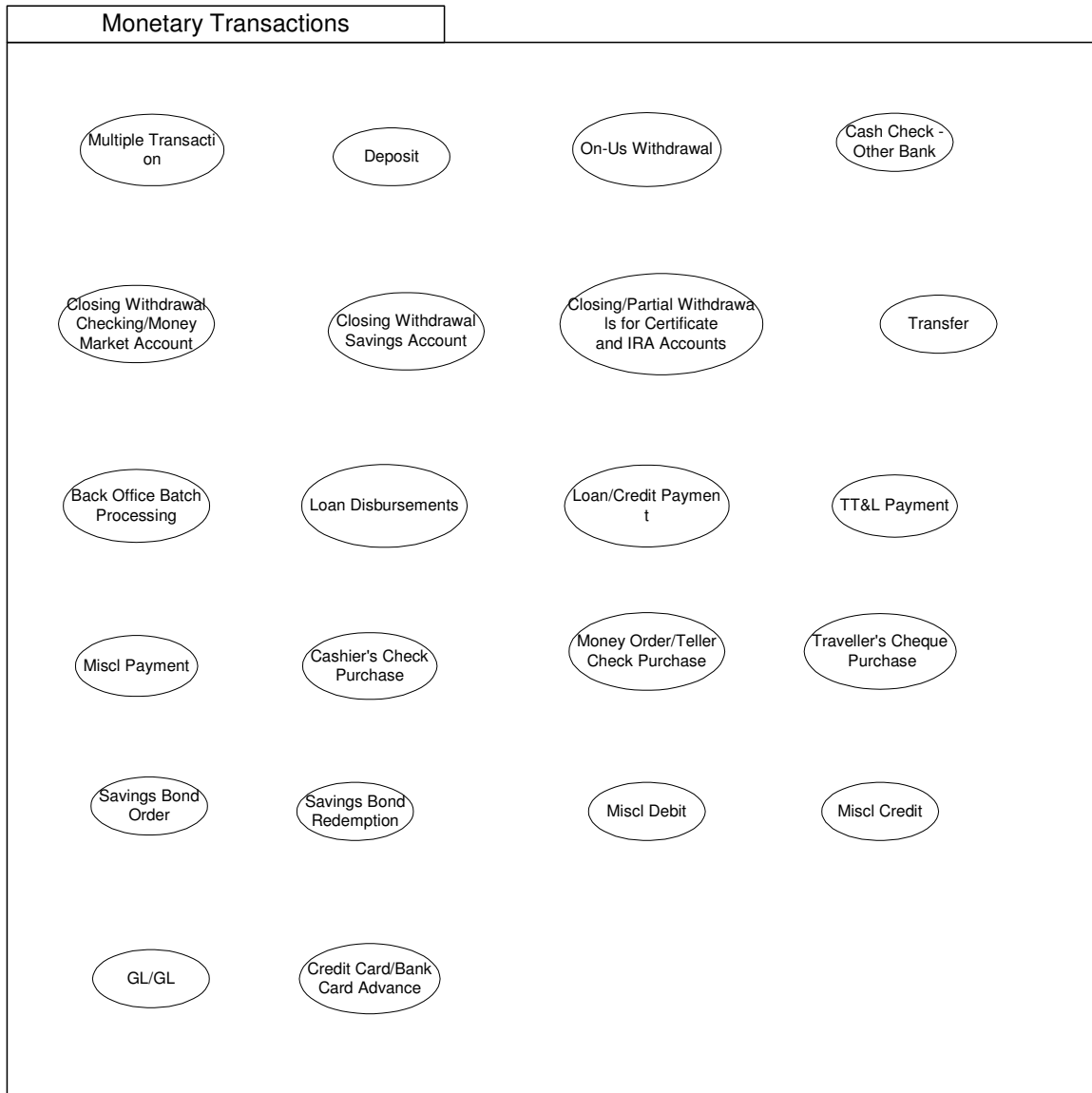
Perform Financial Transactions



Rules/Notes

1. As indicated in the diagram, there are two broad classes of financial transactions – monetary and non-monetary.

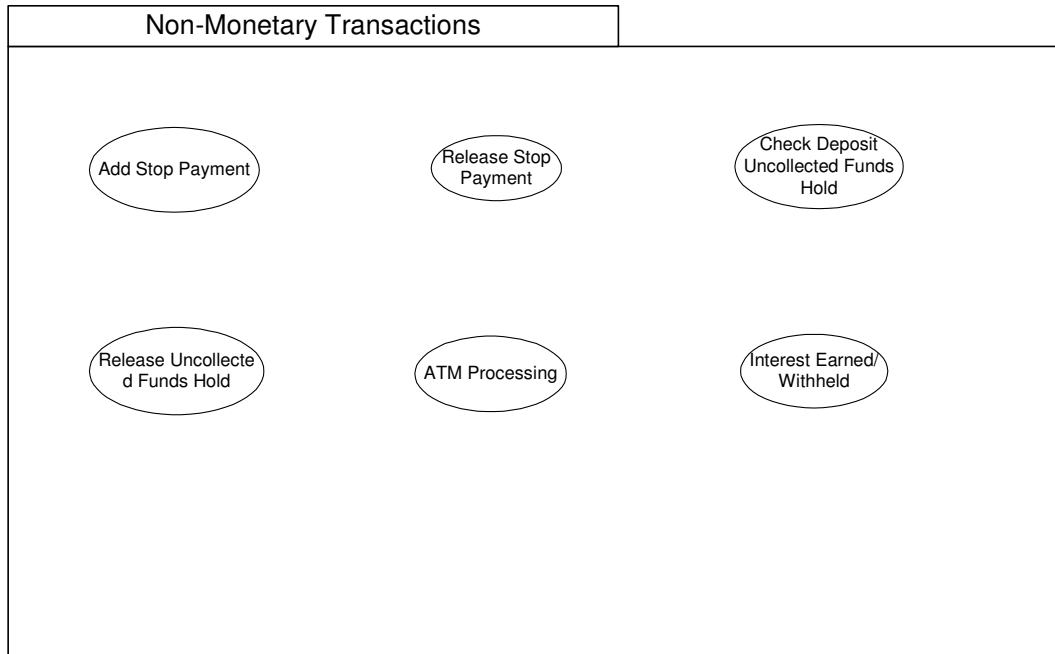
Perform Monetary Financial Transactions



Rules/Notes

1. The particular transactions included in this package will be determined later. The package might include other packages.

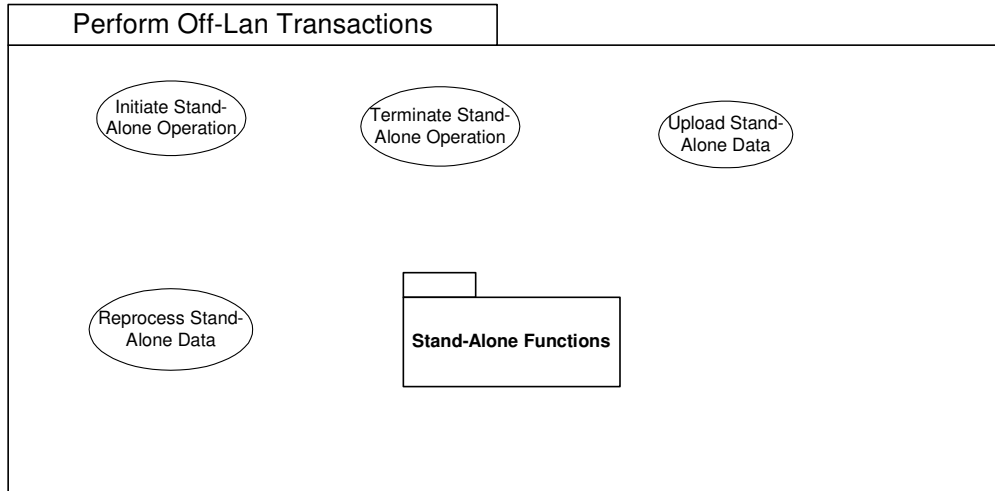
Perform Non-Monetary Financial Transactions



Rules/Notes

1. The particular transactions included in this package will be determined later. The package might include other packages.

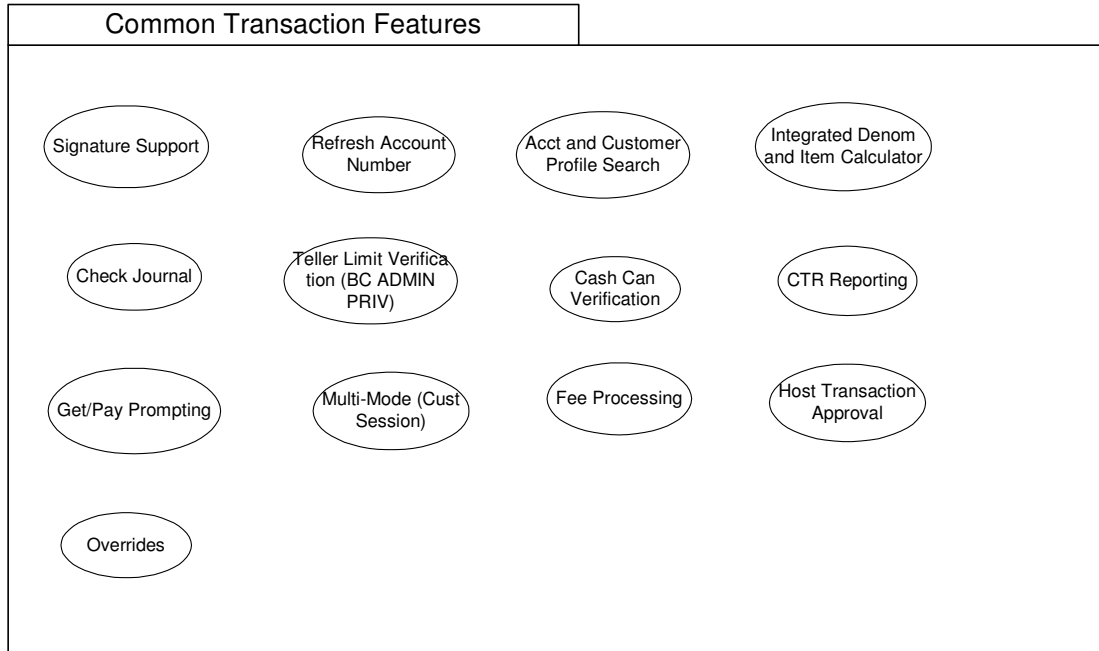
Perform Off-Lan Transactions



Rules/Notes

1. These are “redundant” transactions used to perform basic teller transactions when workstation is in off-lan mode.
2. The basic teller transactions available in off-lan mode are represented by the “Stand-Alone Functions” package – and will be determined at a later time.
3. Printing is restricted to validation and receipts printed at the local printer.

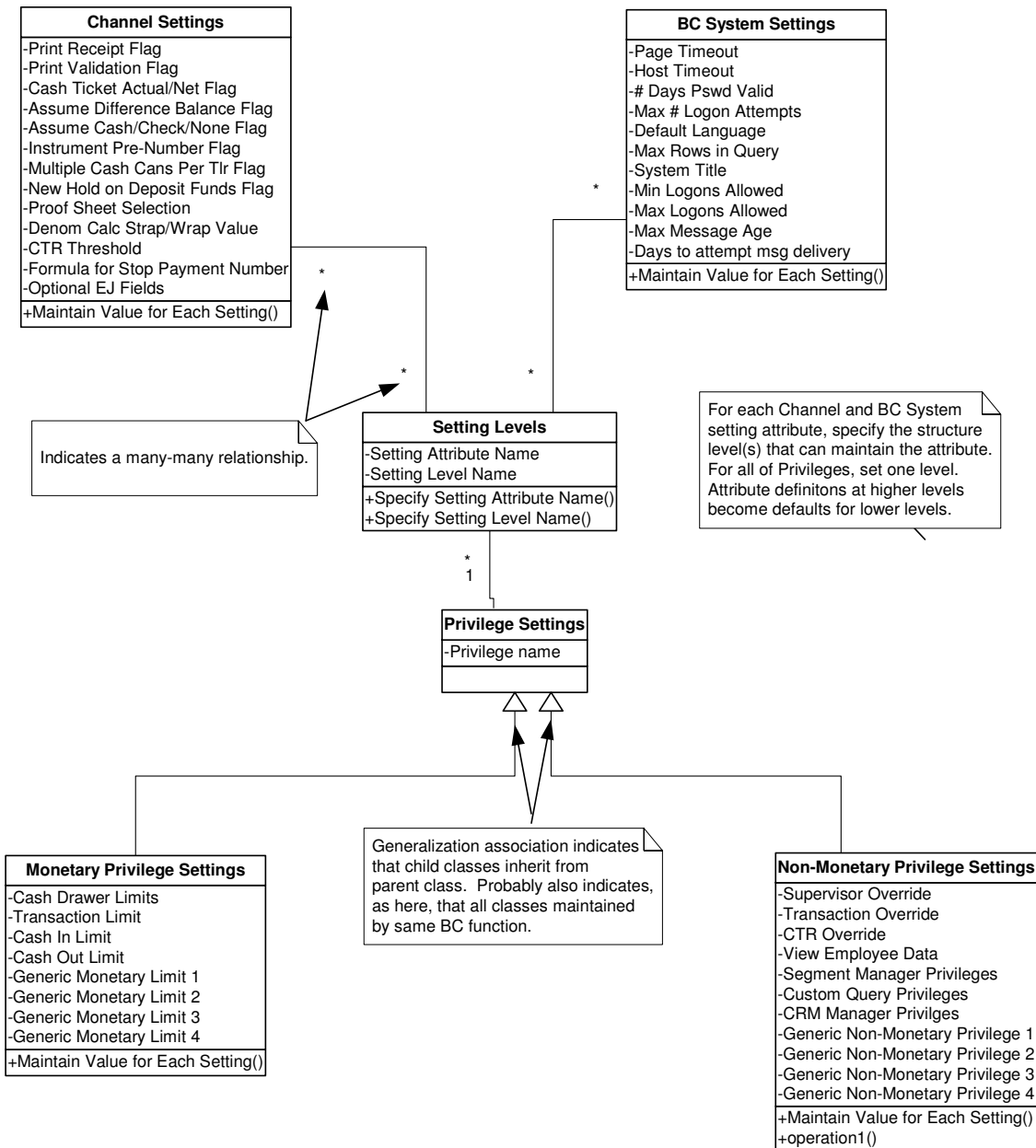
Common Transaction Features



Rules/Notes

1. Common transaction features are used by other teller transactions. These features are not included in authorities and are not directly accessible to tellers.

Setting Class Diagrams



Definition Class Diagrams

