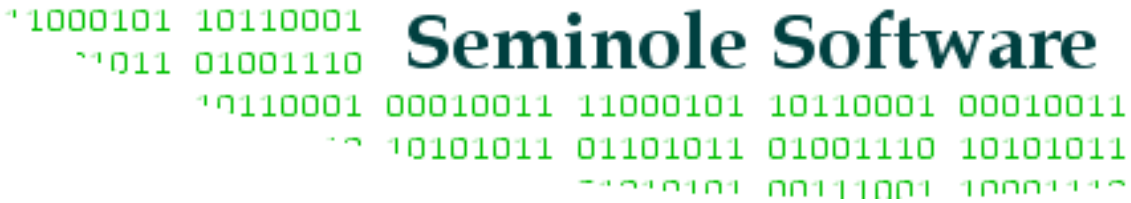


Electronic Stamp Project

Software Requirements Specification

Version: 1.2

Date Created: 2003.09.16

Signatures

Date	Revision	Approved By
2003.10.03	1.0	Nikhil Bandodkar
2003.10.03	1.0	Jidong Long
2003.10.03	1.0	Chuck Weddle

List of Contributors

Name	Initials	Organization	Email
Nikhil Bandodkar	NB	Seminole Software	bandodka@cs.fsu.edu
Jidong Long	JL	Seminole Software	jidolong@cs.fsu.edu
Chuck Weddle	CW	Seminole Software	weddle@cs.fsu.edu

Change History

Revision	Date	Description
1.0	2003.09.16	Initial Revision
1.1	2003.09.27	Second Revision
1.2	2003.10.03	First Final Revision

Preface

This document represents the Software Requirements Specification for the Electronic Stamp project by Seminole Software. The document begins with an Introduction section that describes the purpose of the document and what is considered to be in the scope of this document as well as what is outside the scope of this document.

The next section is an Overall Description of the requirements and functions. This section includes the overall constraints that the project is working within as well as the assumptions made by the project as far as the defining the requirements is concerned. Lastly, the project dependencies are also listed in this section.

The Specific Requirements section comes next and is the most important section of this document. This section goes into detail about each specific requirement of the Electronic Stamp project. A description, use case with sequence of events, and any related requirements is given for each requirement. This section also gives a detailed description of the External Interfaces for the project including a description of the user interface for the software.

The Specific Requirements section also describes the Performance Requirements that are to met by the Electronic Stamp solution. Design Constraints and Standards Compliance are also considered in this section. Lastly, various System Attributes are discussed including Maintainability, Security, and Portability.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The purpose of the Software Requirements Specification is to describe the specific requirements of the Electronic Stamp project that are to be met by the implementation effort of Seminole software. Included with the description of the requirements is a description of any constraints or assumptions that the project is working within.

This document also provides a description of any project dependencies that need to be explicitly expressed. Along with the requirements descriptions, it is also the purpose of this document to describe any performance requirements that need to be met. If there are any standards that need to be considered when developing the software are also listed.

Lastly, the purpose of this document is to communicate the system attributes of the Electronic Stamp software. These system attributes include reliability, availability, scalability, maintainability, and portability.

1.2 Scope

It is within the scope of the Software Requirements Specification to describe the specific system requirements of the Electronic Stamp project. This would include performance requirements, system constraints, and project assumptions. Any specific detail that is needed about the standards or technology used to define these requirements, constraints, and assumptions are within the scope of this document.

It is outside the scope of this document to describe electronic mail systems and technology or the general problem with unwanted electronic mail. It is also outside the scope of this document to describe in any detail at all how certain mentioned standards or technologies work and operate.

1.3 Definitions, Acronyms, and Abbreviations

Table of Definitions, Acronyms, and Abbreviations

Definition, Acronym, or Abbreviation	Description
SRS	Software Requirements Specification.
SMTP	Simple Mail Transport Protocol
POP3	Post Office Protocol 3

1.4 References

Table of References

References	Description
Software Development Plan	The Software Development Plan from the Electronic Stamp project was referenced.

2 Overall Description

2.1 Product Perspective

The Electronic Stamp software that is to be developed by Seminole Software is not a complete email system itself. The Electronic Stamp software and its requirements are only pertaining to the functionality needed to implement the Electronic Stamp system.

Since the Electronic Stamp software is not a complete email system by itself, the software is being developed as part of a previously implemented email software system. Specifically, the Electronic Stamp software extends the Pooka email software. The Pooka email software is an open source application and the Electronic Stamp software will be built into the Pooka email software by changing and extended the Pooka source code.

By doing this, the Electronic Stamp software will provide the standard email functionality provided by the Pooka email software along with the new functionality defined by the Electronic Stamp project.

2.2 Product Functions

The follow is a table of the requirements that the system SHALL meet. The list of requirements was produced from the initial project documentation provided by the requirements expert.

Table of Shall Requirements

ID	Origin	Shall Requirement
19	*Project Description Document*	The Sender SHALL be able to purchase an electronic stamp from a vendor.
20	*Project Description Document*	The Vendor SHALL be able to create an electronic stamp using a public-private key pair with the information sent in the request.
21	*Project Description Document*	The Electronic Stamp SHALL contain a unique ID.
22	*Project Description Document*	A Sender SHALL be able to purchase an electronic stamp from a intermediary between the Sender and the Vendor.
23	*Project Description Document*	The Sender SHALL be able to send an email with an electronic stamp to a Recipient.
24	*Project Description Document*	The Sender SHALL be able to “log” the stamp as valid for return use if the Sender would like to allow the Recipient to respond using the same electronic stamp.
25	*Project Description Document*	The Sender SHALL be able to “log” the stamp as revoked for further use if the stamp was found to be used inappropriately.
26	*Project Description Document*	The Recipients email client software SHALL be able to identify email that have an electronic stamp and SHALL be able to decrypt the electronic stamp using the public key of the vendor from whom the stamp was purchased.

ID	Origin	Shall Requirement
27	*Project Description Document*	The Recipients email client software SHALL reject electronic stamps that do not have a valid Recipient email address with the following exceptions; • The stamp has been "logged" as valid for return or multiple use and the email client software recognizes the stamp. • The stamp has been "logged" as valid for return or multiple use but has been revoked because it appeared on spam. The software can send a reply message stating that a valid electronic stamp is required. • The stamp ID is checked against the email client software's list of used stamp ID's and rejects the email if the stamp is found to have been used without being logged for further use. • The time window of validity of the stamp has expired.
28	*Project Description Document*	The Recipient SHALL be able to "log" an electronic stamp as valid for multiple use. This would be to allow the Sender to respond to the Recipient's responses.
29	*Project Description Document*	The system SHALL allow a user to discard all unstamped mail -or- Reply to unstamped mail with a "challenge" message on how to obtain a stamp and send a stamped message. -or- Selectively filter unstamped mail.

2.3 Constraints

The follow is a table of the design constraints that the system SHALL meet. The list of constraints was produced from the initial project documentation provided by the requirements expert.

Table of Design Constraints

ID	Origin	***Shall Requirement***
1	*Project Description Document*	*A sender not know to the recipient SHALL be able to send a "first class" email message to the recipient with the assurance that it will not be filtered out as spam.*
2	*Project Description Document*	*The cost of an electronic stamp SHALL be competitive to the cost of a hard mail stamp from the USPS.*
3	*Project Description Document*	*The cost of an electronic stamp SHALL be significant enough to make bulk first-class email more expensive then sending regular bulk hard mail via USPS.*
4	*Project Description Document*	*The Email Client Software SHALL be able to be used by recipients who must publish their email addresses. The system does not rely on hiding one's email address.*
5	*Project Description Document*	*The Email Client Software SHALL not require a key distribution infrastructure. It does not require the sender or recipient to have a public or private key.*
6	*Project Description Document*	*The system SHALL use the existing Internet SMTP mail transport protocol.*

ID	Origin	***Shall Requirement***
7	*Project Description Document*	*The system SHALL be backward compatible with previous email systems. It should not inhibit someone from getting an email if their email system does not support the use of electronic stamps.*
8	*Project Description Document*	*The email messages SHALL not be encrypted.*
9	*Project Description Document*	*The system SHALL not rely on the honesty of the origin address in the email header.*
10	*Project Description Document*	*The system SHALL not restrict the content of the email message.*
11	*Project Description Document*	*The system SHALL allow a user to discard all unstamped mail* *-or-* *Reply to unstamped mail with a "challenge" message on how to obtain a stamp and send a stamped message.* *-or-* *Selectively filter unstamped mail.*
12	*Project Description Document*	*The system SHALL allow the recipient to receive email from "blacklisted" domains if the email has a stamp.*
13	*Project Description Document*	*The system SHALL allow the recipient to respond to the sender using the same stamp the sender purchased.*
14	*Project Description Document*	*The system SHALL allow the sender and recipient to have a indeterminate exchange of emails related to an email originally sent from the sender using the original stamp.*
15	*Project Description Document*	*The system SHALL not allow the reuse of stolen or intercepted stamps.*
16	*Project Description Document*	*The system SHALL automate the purchasing and affixing of an electronic stamp to an email.*
17	*Project Description Document*	*The system SHALL protect against stamp forgery using public-key encryption.*
18	*Project Description Document*	*The system SHALL not provide email security. This means that the system will not hide the content of the message or certify that a message has not been forged.*

2.4 User Characteristics

The following table identifies and describes the different users of the Electronic Stamp software. The information gathered about the different users of the system helped define what the software needs to do. Also, these users are referenced in the requirements and diagrams.

Table of User Characteristics

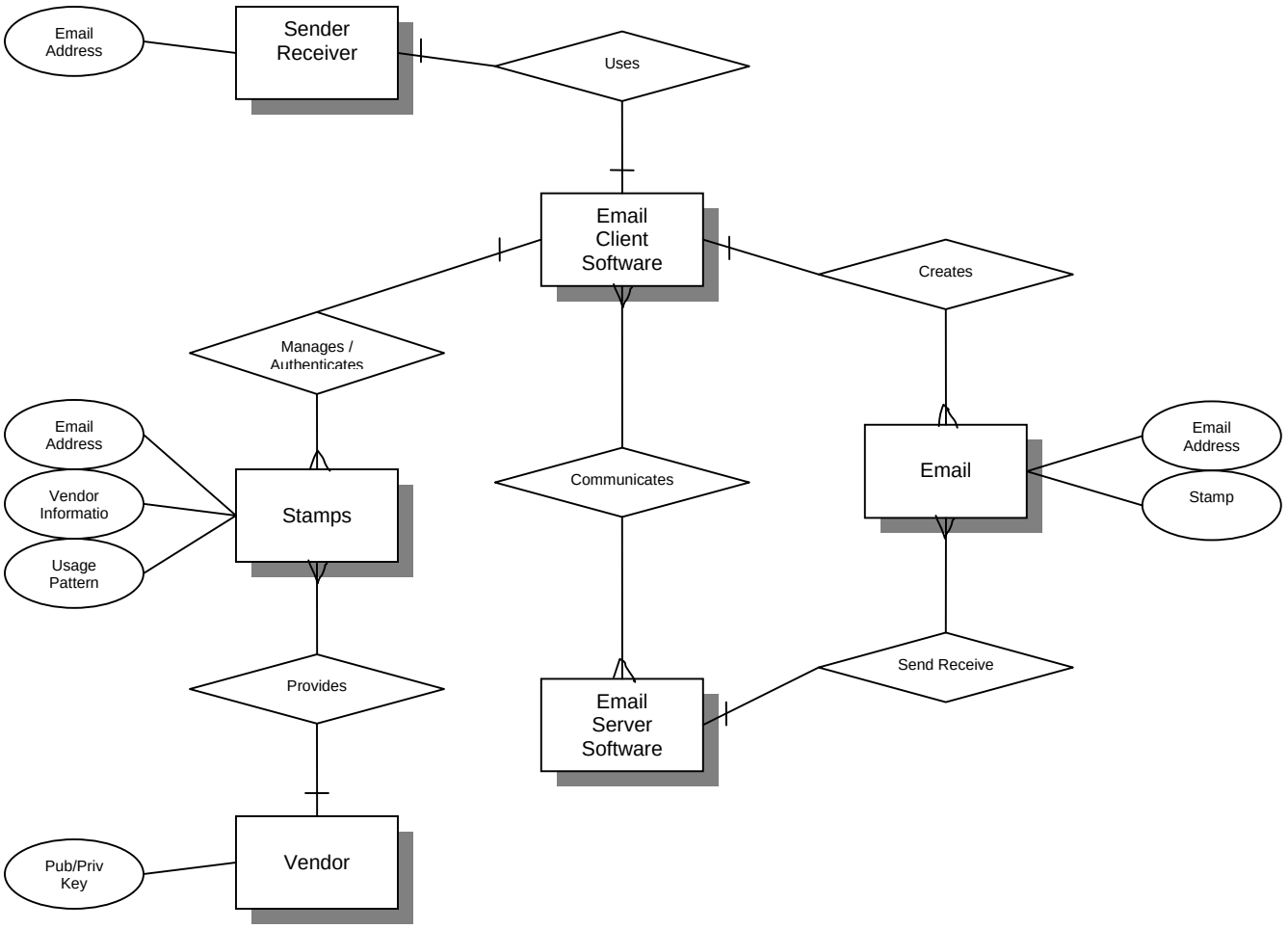
User	Description
Sender	The sender is anyone sending email. This is a vary large group of users from all different backgrounds. Because of this, the system

	should be easy to use and conform to commonly understood user interface styles for wide acceptance.
Recipient	Like the Sender, the Recipient is anyone receiving email. Again, this is a vary large group of users from all different backgrounds. Along with having an commonly understood interface as stated above but the system should be able to be used on a wide range of popular system platforms to be able to meet the wide range of potential users.
Electronic Stamp Vendor	The Electronic Stamp Vendor is most likely a business user that can provide public and private key encryption of a stamp. These users will have to be computer savvy at the very least. A more sophisticated interface, user or otherwise, can be built to accommodate the more advance needs of the electronic stamp vendor.
Administrator	The Administrator user will be computer literate and technically competent in performing administration on computer systems.

2.5 Entity Relationships

Figure 1 shows the entity relationships for the Electronic Stamp project.

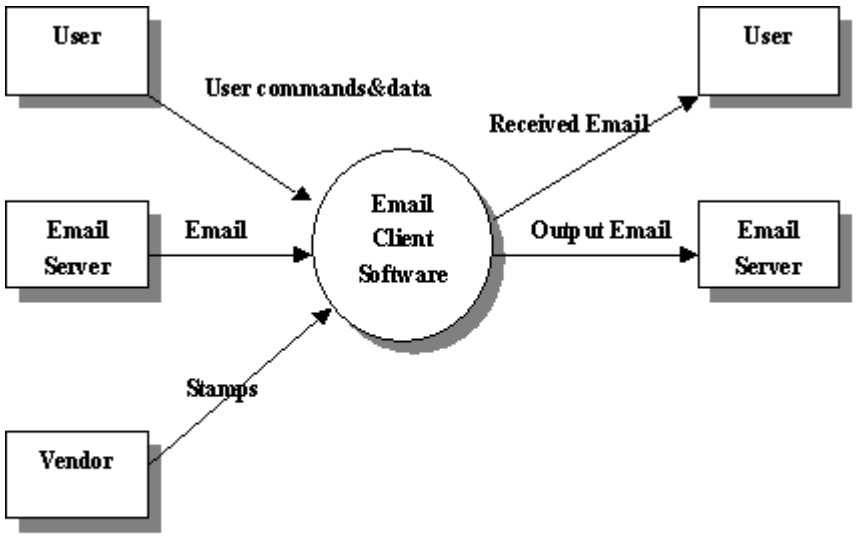
Figure 1 Entity Relationship Diagram

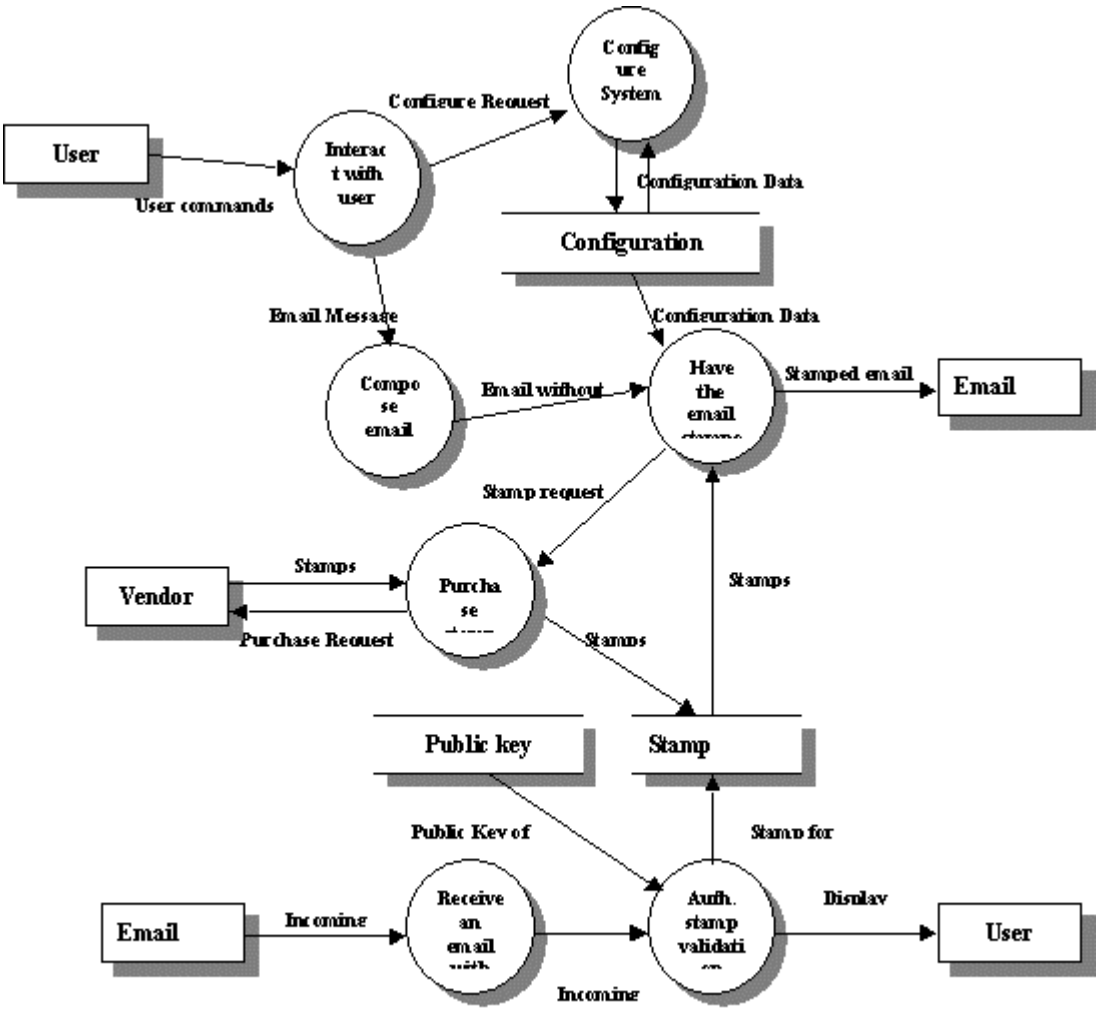


2.6 Data Flows

The following figures represent the data flow diagrams of the Electronic Stamp software. The first data flow diagram, figure 2, is the top level data flow. This is followed by the more detail data flow of the Email Client Software.

Figure 2 Data Flow Diagram





Data Flow Diagram

2.7 Data Dictionary

The follow tables in this section make up the data dictionary for the Electronic Stamp project. Using the Data Flow Diagrams, the following Data Dictionary elements were defined.

- Stamp
- Configuration Data
- Plain Email
- Email with Stamp
- Public Key
- Purchase Request
- User Commands and Data

Table of Data Dictionary

Data Dictionary Attribute	**Detail**
Name	Stamp
Aliases	E. Stamp/Stamp for multi-use
Where Used / How Used	Purchase stamp (input/output) Have the email stamped (input) Authenticate email stamp(output) Stored in the stamp repository
Description	Stamp=Source address + identification of sender + destination address + identification of receiver + serial number + usage pattern + expire data + signature Identification of sender = *any string that is the unique name of the sender* Identification of receiver = *any string that is the unique name of the receiver* Source address =[user + '@'+ server | anonymous] Destination address =[user + '@'+ server | anonymous] User = *any string of ASCII characters* Server = *Domain name of the email server* Serial number = *any string of numbers* Usage pattern = *one byte character* Expire data = month/ + day/ + year Month = * integer from 1 to 12* Day = * integer from 1 to 31* Year = *any four number string* Signature = *128 bits of string*
Supplementary Information	Usage pattern only takes one byte and only three bits of the byte is used which stand for one-to-one usage, reply use and multi-use respectively. Signature is obtained by performing signature algorithm on the content of combination of all the items before signature in the stamp. The output of the signature algorithm is a 128 bit of string (different signature algorithm will have different size of

	output, in this application all signature is supposed to be at 128 bit large.

Data Dictionary Attribute	**Detail**
Name	Configuration data
Aliases	NO
Where Used / How Used	Interact with user (output) Configure system (input/output) Have the email stamped (input)
Description	Configuration Data: Vendor information + authenticator information + Stamp usage Information + information about operations on unstamped mail. Vendor Information = vendor name + IP address of vendor + service port Vendor name =* any string that stands for the unique name of the vendor * IP address of vendor = * any 4 bytes of string* Service port = * any 16 bits long integer * Authenticator information = authenticator name + IP address where the public key of authenticator is available + service port Authenticator name = " any string that stands for the unique name of the authenticator. IP address of the authenticator = * any 4 bytes of string * Stamp usage = * one 32 bit long integer * Information about operations on unstamped email = * one 32 bit long integer *
Supplementary Information	Stamp usage and information about operations on unstamped email will take one 32 bit long integer each. Every bit in the integer is an indicator for an operation or usage, the detailed definition for those bits haven't finished yet!

Data Dictionary Attribute	**Detail**
Name	Plain email
Aliases	email message/display information
Where Used / How Used	Transform the input into an email (input) Authenticate email stamp (output)
Description	Plain email = email header + email message + email attachments
Supplementary Information	More detail of email structure can be found in the specification of SMTP.

Data Dictionary Attribute	**Detail**
Name	Email with stamp
Aliases	incoming email/outgoing email

Where Used / How Used	Have the email stamped (output) Receive an email with stamp (input) Authenticate email stamp (input)
Description	Email = plain email + stamp Plain email = email header + email message + email attachments Email message = *any string of ASCII characters* Email attachments =*one or more files of any type*
Supplementary Information	Plain email is an email without a stamp sent out by this software. The clear description of data type of email header, email message and email attachment is not provided here for they are not going to be used by this implementation of this software. The third party of the email package will be used to marshal / unmarshal email automatically. More detail can be found in the specification of SMTP. The description of stamp is already provided in the document.

Data Dictionary Attribute	**Detail**
Name	Public key
Aliases	public key of vendor
Where Used / How Used	authenticate email stamp (input)
Description	Public key = * n times 64 bits of string, n is from 1 to 40*
Supplementary Information	

Data Dictionary Attribute	**Detail**
Name	Purchase Request
Aliases	NO
Where Used / How Used	Purchase stamp (output)
Description	Purchase Request: identification of sender + sender IP + identification of receiver + receiver IP + number of stamps + credit card information Identification of sender = *any string that is the unique name of the sender* Identification of receiver = *any string that is the unique name of the receiver* Sender IP =[user + '@'+ server | anonymous] Receiver IP =[user + '@'+ server | anonymous] Number of stamps =* any integer from 1 to 1024 * Credit card information = card type + card number + billing address Card type=[Visa | MasterCard | American Express | Discovery] Card number = * any string of numbers *

	Billing address = address + city + state + country + zip code
Supplementary Information	

Data Dictionary Attribute	**Detail**
Name	User Commands and Data
Aliases	NO
Where Used / How Used	Interactive with user (input)
Description	User Commands and Data = commands + data Commands = [send email | reply email | save email | delete email | create folder | receive email |….|] Data = * any email message and attachment data * + configuration data
Supplementary Information	Commands originate from the user by clicking specific button or menu items. Commands will be converted into events in the operations system automatically and then the email client software captures those events and responds them accordingly. The data could be any that could be used to construct an email or provide the configuration information.

2.8 State Transitions

Please refer to Appendix A for the State Transition Diagrams. The first state transition diagram shows the state transitions when a Sender sends an email. The second diagram in Appendix A shows the state transitions when a Receiver receives an email or when a Sender receives a reply email. The last state transition diagram shows the state transitions when a Receiver sends a reply message.

2.9 Assumptions

The following table lists the assumptions made by the requirements that define the Electronic Stamp software.

Assumption	**Description**
Secure Channel from Sender to Vendor	The defined requirements assume that there is a method for secure transaction between the Sender and the Vendor.

3 Specific Requirements

3.1 System Features

3.1.1 Sending Email

3.1.1.1 Introduction

The Electronic Stamp software shall allow a user to send email to a recipient via a mail server, with or without a stamp. The user must first compose and edit the message using Pooka's email editor. The process of purchasing the stamp can be performed automatically when sending an email.

3.1.1.2 Functional Requirements

Purpose: Sending the composed email to the email server with or without a stamp.

Input: Saved Text Message, Recipients Email Address, Subject of the Message (optional), and Whether a Stamp is required or not.

Processing: If stamp is required, obtaining a digitally signed Stamp from Vendor. Appending the Stamp to the Outgoing Email. Logging the message.

Output: The email with a recipient email address, with or without the stamp sent out to the Mail server. A message displaying the status of the email transaction.

3.1.1.3 Stimulus Response

A) User Sends Email with stamp and with Automated Stamp Purchase.

User Actions	**System Actions**
(1) Compose/Edit Email	
(2) Initiate the Send Email Function	
	(3) Close the Graphical Mail Composition Window
	(4) Query if stamp is required
(5) Reply to this query: Stamp is Required	
	(6) System Retrieves Information from the automated purchase stamp configuration.
	(7) Contact Vendor
	(8) Obtain stamps
	(9) Display Information regarding Purchase of stamps.
(10) Confirm Purchase	
	(11) Query for Stamp configuration (Return use or single use)
(12) Reply to this query	
	(13) Append the stamp to the Email

	(14) Send the Email to the mail server
	(15) Log the Email
	(16) Display a text message regarding status of the Email transaction.

B) User Sends Email with stamp and with Manual Stamp Purchase.

User Actions	System Actions
(1) Compose/Edit Email	
(2) Initiate the Send Email Function	
	(3) Close the Graphical Mail Composition Window
	(4) Query if stamp is required
(5) Reply to this query: Stamp is Required	
	(6) Open Manual Stamp Purchase Function if Automated Stamp Purchase is not Configured. (See 3.2.5.2 below)
	(7) Query for Stamp configuration (Return use or single use)
(8) Reply to this query	
	(9) Append the stamp to the Email
	(10) Send the Email to the mail server
	(11) Log the Email
	(12) Display a text message regarding status of the Email transaction.

C) User Sends Email without Stamp.

User Actions	System Actions
(1) Compose/Edit Email	
(2) Initiate the Send Email Function	
	(3) Close the Graphical Mail Composition Window
	(4) Query if stamp is required
(5) Reply to this query: Stamp is Not Required	
	(6) Send the Email to the mail server
	(7) Log the Email
	(8) Display a text message regarding status of the Email transaction.

3.1.2 Receiving Email

3.1.2.1 Introduction

The software regulates spam messages by putting all unstamped messages in a separate folder. Only stamped messages go into the Inbox.

3.1.2.2 Functional Requirements

Purpose: Receiving Email from the email server and screening as much of the spam as possible.

Input: Email, Recipients Email address, Stamp with unique stamp ID (optional), Senders E- Mail Address (optional).

Processing: If stamped, obtaining the vendors public key and verifying the signature, checking the unique Stamp ID. If Not Stamped, applying selective filter to senders Email address.

Output: Mail from desirable source in Inbox. Spam in the Spam Folder. Challenge message to spammers.

3.1.2.3 Stimulus Response

User Actions	System Actions
	(1) Receive email.
	(2) Check if Stamp is present.
	(3) Gets public key in case the public key is compromised or hasn't been obtained by the ECS.
	(4) Verify if Stamp is Valid.
	(5) If stamp is valid put Email in Inbox.
	(6) If Stamp is not present System applies selective Filter and puts Email from previously selected Source in Inbox.
	(7) All undesirable mail Sources are sent a Challenge message and the Email is categorized as Spam.
(8) User views the email in Inbox or any other Folder.	
(9) User deletes Email or marks Stamp as Valid or Invalid.	
	(10) Saves the users preference for

3.1.3 Replying to Email

3.1.3.1 Introduction

This feature allows the user to reply to Email that he receives from a sender whose Email address is available to the user. This is similar to the Reply function typically found in all Email clients.

3.1.3.2 Functional Requirements

Purpose: Replying to Email with or without a stamp.

Input: Text Message, Recipients Email Address, Subject of the Message (optional), and Whether a Stamp is required or not.

Processing: If stamp is required, obtaining a digitally signed Stamp from Vendor. Appending the Stamp to the Outgoing Email Logging the message

Output: The Email with a recipient Email Address, with or without the stamp sent out to the Mail server. A message displaying the status of the Email transaction.

3.1.3.3 Stimulus Response

A) Reply to a Stamped Email (with stamp logged for Return Use.)

User Actions	System Actions
(1) Call reply function to Email in either Inbox or any other folder.	
	(2) System Checks for presence of stamp.
	(3) System checks if the stamp is logged for return use: Stamp is Logged for Return Use.
	(4) Initiate the Compose Email function
(5) Compose/Edit Email	
(6) Initiate Send Email function.	
	(7) Query for Stamp configuration (Approve for Multiple use)
(8) Reply to query.	
	(9) Send the Email to the mail server
	(10) Log the Email
	(11) Display a text message regarding status of the Email transaction.

B) Reply to a Stamped Email (with stamp not logged for Return Use) with a Stamped Email.

User Actions	System Actions
(1) Call reply function to Email in either Inbox or any other folder.	
	(2) System Checks for presence of stamp.
	(3) System checks if the stamp is logged for return use: Stamp is not logged for Return Use.
	(4) Query if Reply with Stamped mail or Unstamped Mail.
(5) Reply to query: Send Stamped Email	
	(6) Initiate the Compose Email function
(7) Compose/Edit Email	
(8) Initiate Send Email function.	
	(9) Close the Graphical Mail Composition Window
	(10) Retrieve Information from the automated purchase stamp configuration or alternatively Manually.
	(11) Contact Vendor
	(12) Obtain stamps
	(13) Display Information regarding Purchase of stamps.
(14) Confirm Purchase	

	(15) Send the Email to the mail server
	(16) Log the Email
	(17) Display a text message regarding status of the Email transaction.

C) Reply to a Stamped Email (with stamp not logged for Return Use) with an Unstamped Email.

User Actions	**System Actions**
(1) Call reply function to Email in either Inbox or any other folder.	
	(2) System Checks for presence of stamp.
	(3) System checks if the stamp is logged for return use: Stamp is not logged for Return Use.
	(4) Query if Reply with Stamped mail or Unstamped Mail.
(5) Reply to query: Send Unstamped Email	
	(6) Initiate the Compose Email function
(7) Compose/Edit Email	
(8) Initiate Send Email function.	
	(9) Close the Graphical Mail Composition Window
	(10) Send the Email to the mail server
	(11) Log the Email
	(12) Display a text message regarding status of the Email transaction.

D) Reply to an Unstamped Email

User Actions	**System Actions**
(1) Call reply function to Email in either Inbox or any other folder.	
	(2) System Checks for presence of stamp.
	(3) Stamp not present.
	(4) Initiate the Compose Email function
(5) Compose/Edit Email	
(6) Initiate Send Email function.	
	(7) Send the Email to the mail server
	(8) Log the Email
	(9) Display a text message regarding status of the Email transaction.

3.1.4 Automated Stamp Purchase

3.1.4.1 Introduction

The Automation of stamp purchase allows the user to send Email without having to perform actions to purchase a stamp from a vendor every time Email is sent. In this feature the user configures the software and provides it all the information that is required to purchase a stamp. All that the software asks of its user is whether a stamp is required or not. If required, the software

automatically proceeds to purchase the stamp from the vendor and attach it to the Email before sending the Email.

3.1.4.2 Functional Requirements

Purpose: Gathering information for automating the purchase of stamps.

Input: Whether or not Automation is required, Vendor Information, Billing Information.

Processing: None

Output: All the gathered information is stored at an appropriate location to be retrieved when E Mail has to be sent.

3.1.4.3 Stimulus Response

User Actions	**System Actions**
(1) Initiate Stamp Purchase Configuration	
	(2) Open Stamp Purchase Configuration Dialog
(3) Enter Purchase information including (but not limited to) whether automation is desired, Vendor and Billing Information.	
	(4) Save information for retrieval while sending Email.
	(5) Close Stamp Purchase Configuration Dialog

3.1.5 Manual Stamp Purchase

3.1.5.1 Introduction

A user may choose to purchase a stamp, by providing relevant information at the time of sending the Email. The Manual Stamp Purchase Feature asks the user to input the same information as in automated Stamp purchase, except that it does so just before sending the Email.

3.1.5.2 Functional Requirements

Purpose: Gathering information for purchasing stamps.

Input: Receiver Email Address, Vendor Information, quantity, Billing Information.

Processing: None

Output: The software contacts the Vendor via a secure channel and obtains a digitally signed stamp. The stamp is appended to the Email.

3.1.5.3 Stimulus Response

User Actions	System Actions
(1) Initiate purchase stamp function.	
	(2) Client software begins purchase stamp dialog.
(3) User inputs (Receiver email address, vendor to purchase stamps from, quantity and so on)	
	(4) ECS communicates with vendor software
	(5) Receive stamps
	(6) Append Stamps
	(7) Display Message regarding status of the purchase.
(8) Confirm the actions.	

3.2 Performance Requirements

The following tables list the performance requirements of the Electronic Stamp software.

Table of Performance Requirements

Performance Requirement	**Description**
Email Storage Capacity	The email storage capacity of the Electronic Stamp software will be configurable to the extent that the Pooka software allows and the hardware permits.
Software Runtime Errors	The Electronic Stamp software will handle the runtime errors consistently and as gracefully as possible.

3.3 Design Constraints

The Electronic Stamp project is extending the Pooka email software to implement the required functionality, which has placed certain design constraints on the design of the Electronic Stamp software. The table below lists those design constraints.

Table of Design Constraints

Design Constraint	Description
Pooka Design Paradigm	The source code written to extend Pooka to provide the Electronic Stamp software will comply with the design paradigm used in Pooka.

Design Constraint	Description
Pooka Email Functionality	The Electronic Stamp software will only provide the email functionality that is provided in Pooka. The Electronic Stamp software will extend Pooka only for the features required to implement the electronic stamp solution.
Java	The Pooka source code is written in Java and because of this, the source code written to implement the features of the Electronic Stamp software will also be in Java.
Object Oriented Design and Programming	An Object Oriented Design and Programming method was employed in designing and programming Pooka. This means that the Electronic Stamp software will also employ the Object Oriented Design and Programming methods to best integrate with the Pooka software.

3.4 Standards Compliance

The following table lists the different standards that the Electronic Stamp project is to be in compliance with.

Table of Standards Compliance

Standard	**Description**
SMTP	The Simple Mail Transport Protocol (SMTP) transfers mail reliably and efficiently. The Pooka email software and the Electronic Stamp extensions of the Pooka email software will comply with the SMTP standard.
POP3	The Post Office Protocol 3 (POP3) is the service that stores and serves emails for client machines that are not connected to the Internet 24 hours a day.
Public / Private Key Standard Suite	The electronic stamp vendor will digitally sign the electronic stamp using the vendor's public and private key pair.

3.5 Software System Attributes

3.5.1 Reliability

Reliability in the Electronic Stamp software will be ensured by thorough unit, milestone, and release testing. Comprehensive test scenarios and acceptance criteria will be established to reflect the necessary level reliability required of the Electronic Stamp software. The all delivered

source code will be thoroughly tested using the established test scenarios until the acceptance criteria are satisfied by the Electronic Stamp software.

3.5.2 Security

The Electron Stamp software will utilize Public / Private key encryption. Every stamp purchased from a Vendor will be digitally signed using the vendor's public and private key pair. The email client software will use the public key of the vendor that the stamp was purchased from to decrypt the stamp and verify the validity of the email. This will provide an electronic stamp that is as secure as the public / private key encryption method is secure.

3.5.3 Maintainability

The Electronic Stamp software extends the functionality of the Pooka email client software. The Pooka email client software is written in Java, a portable programming language. Because Pooka is written in Java, the Electronic Stamp software will also be written in Java. Java promotes good design practices due to the inherent structure of a Java program.

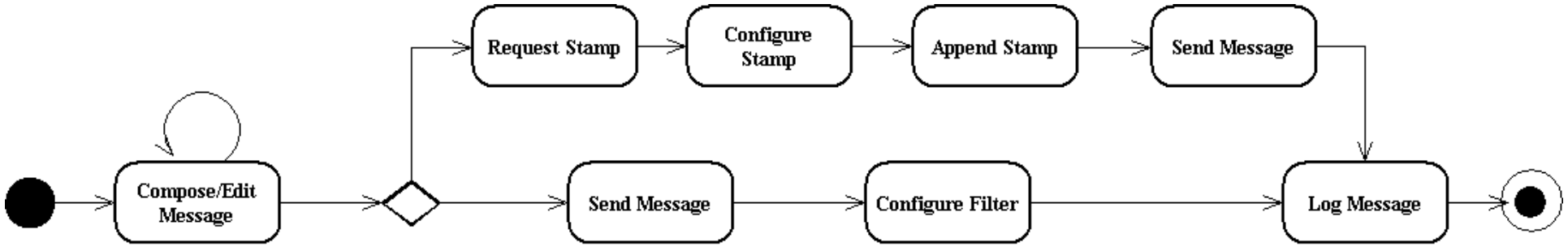
Along with the well-formed programming enforced by Java, best practice development conventions will be enforced for the construction of the Electronic Stamp software. These will include adequate commenting within the source code that complies with and uses the automated Java documentation standard so that source code documentation will be able to be automatically generated. Consistent variable naming conventions will be used by all the programmers. Consistent spacing will be used in the source code by all the programmers. The design of the source code will use the principles of Object-Oriented Design and the source code will be programmed using Object-Oriented Programming. Object-Oriented Design and Object-Oriented programming will make the code easier to understand.

3.5.4 Portability

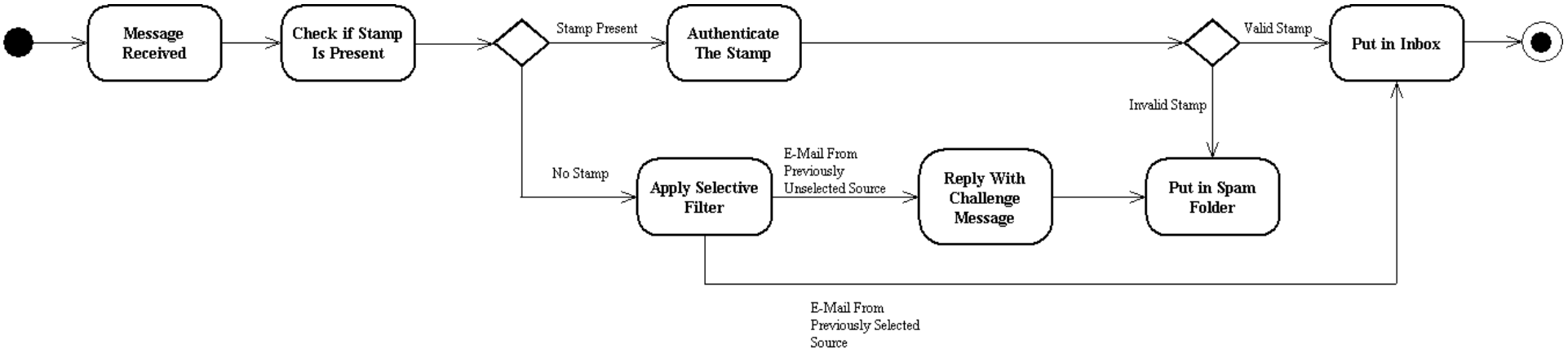
As mentioned above, the Electronic Stamp software extends the functionality of the Pooka email client software. Because Pooka is written in Java, the Electronic Stamp software will also be written in Java and gain the portability providing by that language.

It is safe to say that the implementation of the Electronic Stamp software will be able to be ported to other system platforms that accept Java applications with little to no changes required. It is not safe to say that the Electronic Stamp software will execute properly on the other system platforms with little or no change. Significant changes to the Electronic Stamp software may be required to ensure proper execution on other system platforms.

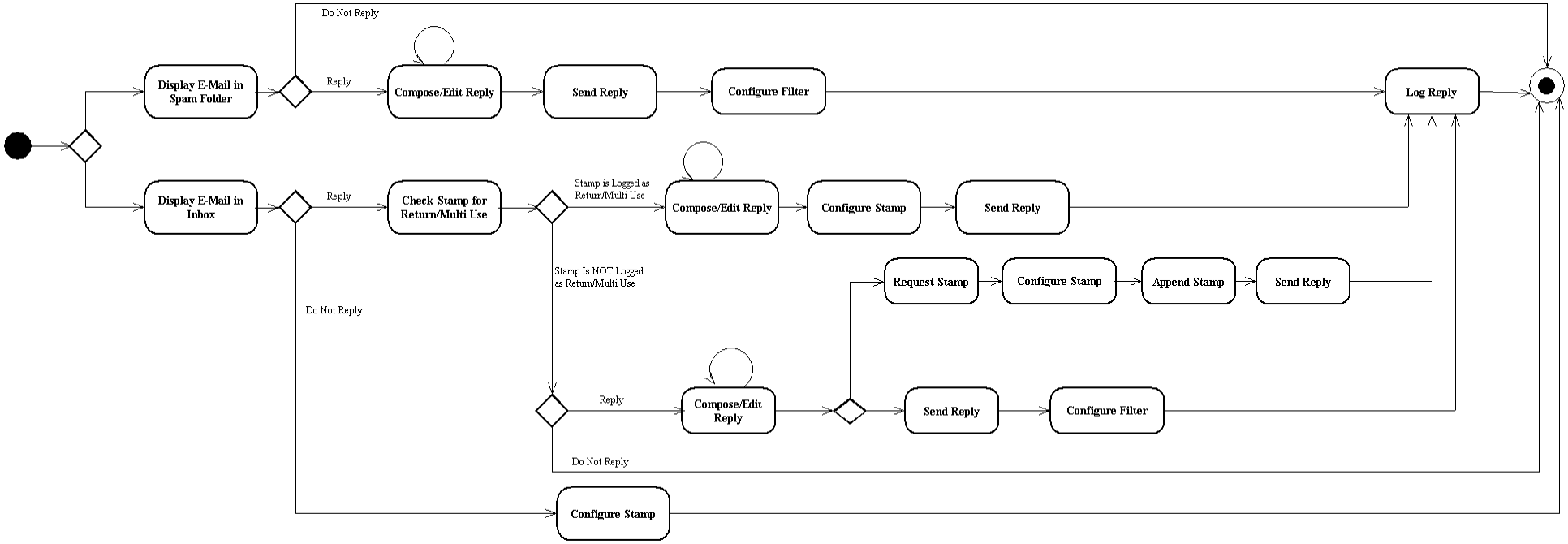
Appendix A – State Transition Diagrams



State Transition Diagram: Sender Sends E-Mail



State Diagram 2: Receiver Receives E-Mail OR Sender Receives Reply



State Diagram 3: Receiver Sends Reply