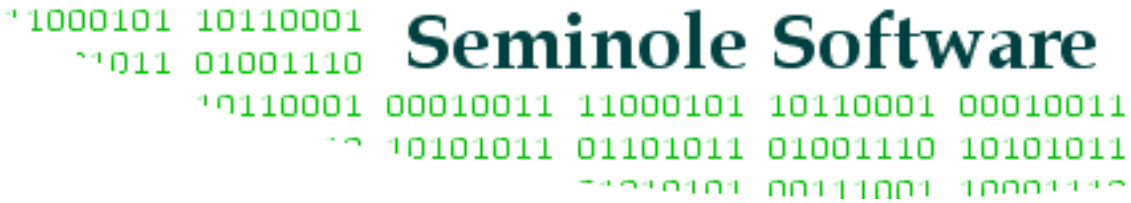


Electronic Stamp Project

Software Design Specification

Version: 1.2

Date Created: 2003.10.24

Signatures

Date	Revision	Approved By

List of Contributors

Name	Initials	Organization	E-Mail
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	2003.10.24	Initial Revision
1.1	2003.10.30	Second Revision
1.2	2003.11.07	Third Revision

Preface

This document presents the Software Design Specification for the Electronic Stamp project by Seminole Software. The major sections of the document address the system decomposition by module, concurrent process, and data entity. The system dependencies are also described.

Section 2, Decomposition Description, gives a view of the whole system design including concurrent processes and data entities that are common amongst all system modules. An important discussion of how the E-Stamp modules extend the existing Pooka email client software is included this section. This discussion includes a UML Class Diagram that depicts the entire system.

Section 4, Interface Description, goes into detail about the user interface for each module of the E-Stamp software. This is followed by an important discussion of the processes implemented in logic for each module of the system.

Section 5, Detailed Design, extends the design discussion found in Section 2 and describes the design for each system module in more detail. A UML Class diagram is included for each module design discussion. This is followed by a description of the data requirements for each module and the design of those data elements.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The purpose of the Software Design Specification is to describe the specific design of the Electronic Stamp software by Seminole Software. The design specification includes an overview of the design along with software module decomposition.

This document provides a detailed description of each software module's design. For each module, a user interface design and class diagram design is given. As well, a process description is described for each module. It is in the process description that the details of what logic will need to be implemented are given.

1.2 Scope

It is within the scope of the Software Design Specification to describe the specific system design of the Electronic Stamp project. This would include user interface design, object-oriented class design, process design, and data design. Any specific detail that is needed about the standards or technology used to design the software 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 and Acronyms

Table of Definitions, Acronyms, and Abbreviations

Definition, Acronym, or Abbreviation	**Description**
SDS	Software Design Specification.

1.4 References

Table of References

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

2 Decomposition Description

2.1 Module Decomposition

The E-Mail Client Software has been decomposed into the following modules.

- E-Stamp (Vendor) Configuration Module: This module collects data from the user to be used for establishing communication with the vendor.
- E-Stamp Vendor Module: This module communicates with the vendor over the Internet. It also obtains the stamp from the vendor.
- E-Stamp Authentication Module: This module authenticates an E-Stamp received by the E-Mail client.
- E-Stamp Manager Module: This module collects information (data) regarding stamp purchase and orientation of the stamp (whether single use, multi use or return use). It also allows the user to manage the stamp book.
- Sending E-Mail Module: This module uses SMTP protocol to send e-mail. It also determines if a new stamp is required if a stamp is not available.
- Receiving E-Mail Module: This module receives e-mail, sorts it and places it in separate folders depending on whether they are stamped or unstamped.

2.2 Concurrent Process Decomposition

The E-Stamp Project consists of two major components, the E-Mail Client and the Vendor. This team shall design the E-Mail Client. The design of the vendor software is out of the scope of the current team's task.

A complete view of the project suggests that there are two processes, the vendor process and the E-Mail client process. The E-Mail client process communicates with the vendor process to obtain an E-Stamp. These two processes run concurrently and only exchange information when the E-Mail client process requires a stamp.

2.3 Data Decomposition

The following are the two major data components, the E-Stamp Purchase Information and the Electronic Stamp.

E-Stamp Purchase Information: This is a database that contains the following data items;
Vendor Name: A string-containing name of vendor. Used only for identification of a vendor by the user.

- Vendor URL: The vendors URL.
- Password: A Password to obtain access to the Vendors website.
- Senders E-Mail Address.
- Senders Credit Card Information: For purchase of E-Stamp.
- Stamp Orientation: Whether Single Use, Return Use or Multi Use.
- Receivers E-mail address.

The Electronic Stamp: This is a data structure that is attached with the outgoing email. It contains the following data.

- Vendors name: For identification of vendor so that the receivers E-Mail client can obtain vendors public key if it does not already have one.
- Senders E-Mail address.
- Receivers E-mail address.
- Orientation of the stamp.
- Digital signature of all the above information.

3 Dependency Description

3.1 Inter-module Dependencies

3.1.1 Independent Modules

The following modules are independent and do not rely on any other modules to initiate them or to provide data.

- E-Stamp (Vendor) Configuration Module.
- E-Stamp Manager Module.

3.1.2 Dependent Modules

The following modules are dependent on one another for their functioning.

- E-Stamp Vendor Module: This is the communication module. This module is executed only when the Sending E-Mail Module calls it. For The E-Stamp Vendor Module to complete its function it must interact with the E-Stamp (Vendor) Configuration Module and the E-Stamp Manager Module.
- E-Stamp Authentication Module: This module is executed when the Receiving E-Mail Module calls it.
- Sending E-Mail Module: This module depends on the E-Stamp vendor module for its successful completion.
- Receiving E-Mail Module: This module depends on the E-Stamp Authentication Module for its successful completion.

3.2 Inter-process Dependencies

As described earlier the two main processes are the E-Mail client process and the Vendor process. The email client process depends on the vendor process only for obtaining stamps. This is the only dependency between the two processes. Please reference Appendix A for a full class diagram of the E-Stamp classes.

3.3 Data Dependencies

The following Data Flow Diagram shows the data dependencies between the various entities and modules.

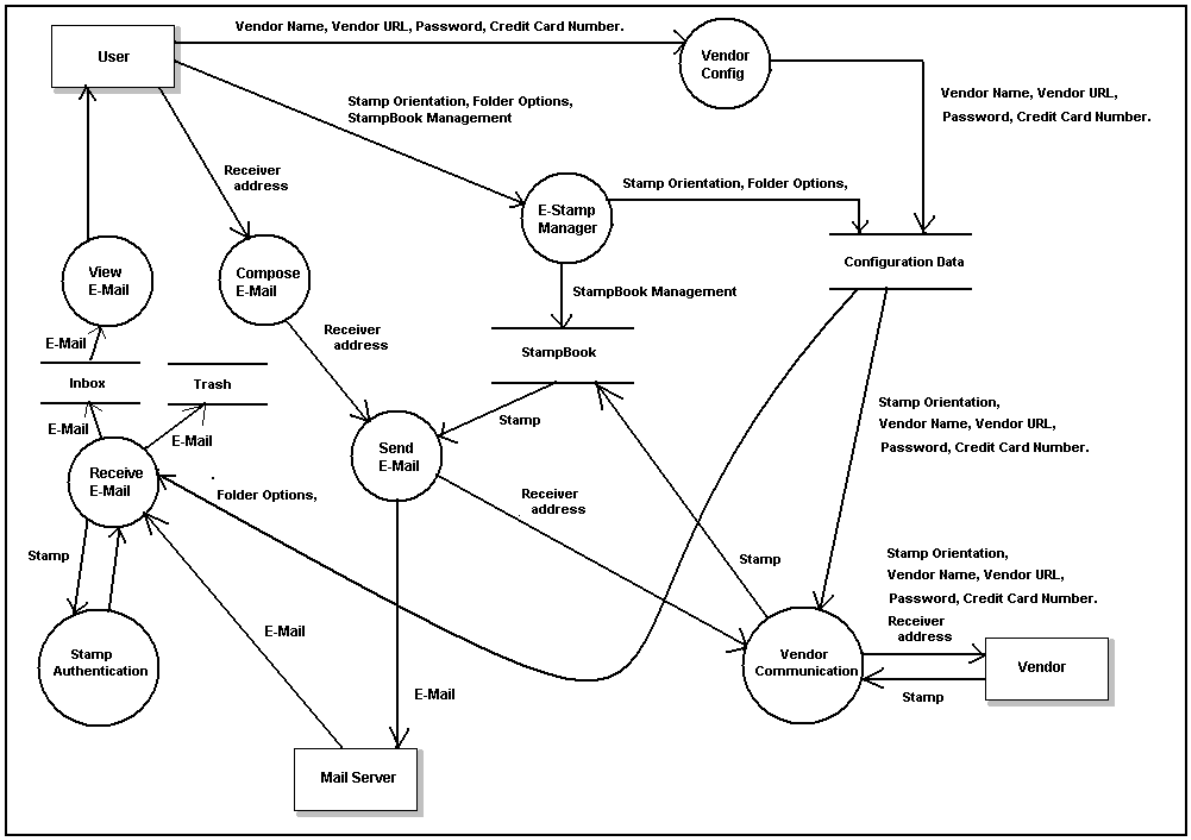


Figure 1, Data Flow Diagram

4 Interface Description

4.1 Module Interface

4.1.1 E-Stamp Configuration Module Description

4.1.1.1 User Interface Design

Figure 2, Configuration Module, E-Stamp Config Menu

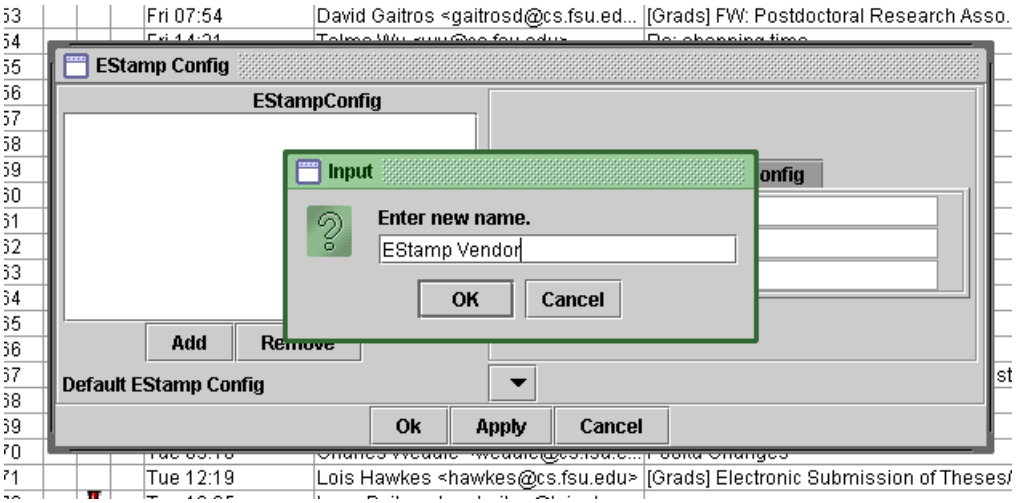


Figure 3, Configuration Module, E-Stamp Vendor Name

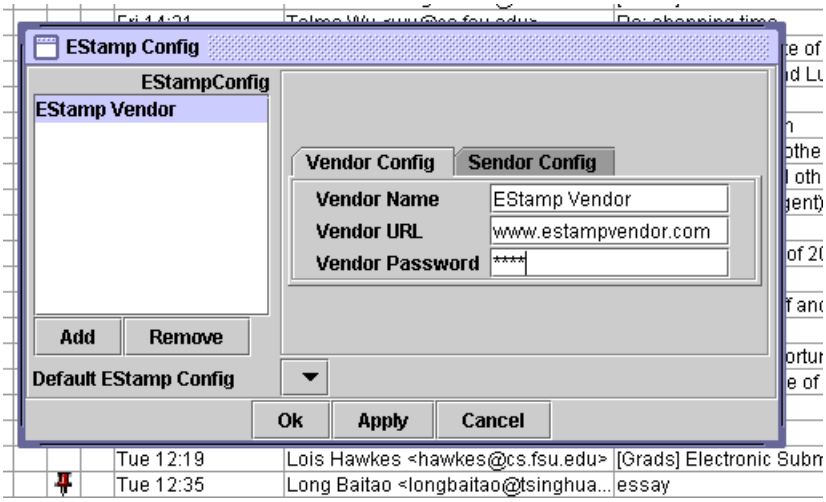


Figure 4, Configuration Module, E-Stamp Vendor Information

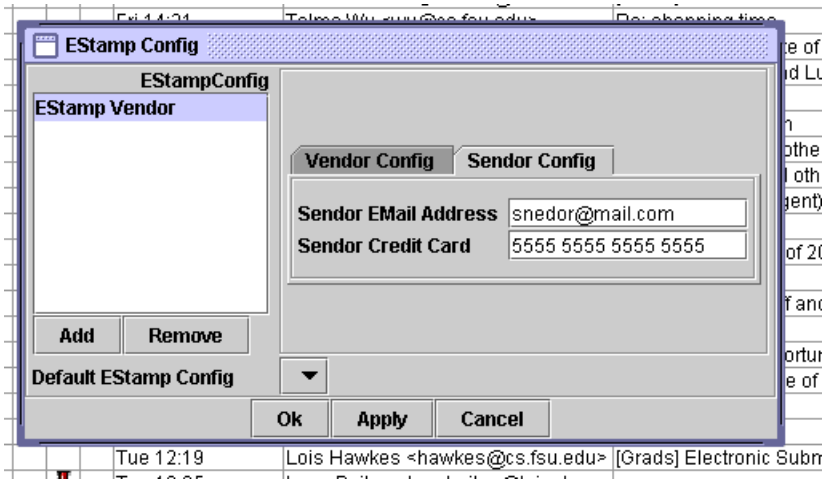


Figure 5, Configuration Module, E-Stamp Sender Information

4.1.1.2 Description

The E-Stamp Configuration User Interface can be accessed via the main menu under the heading ‘Electronic Stamp’. The UI in figure 4 allows the user to provide information regarding a new E-Stamp vendor. In figure 4 the user provides the Vendors URL and password. Finally, in figure 5 the user provides his primary e-mail address and credit card information for the E-Stamp transaction.

4.1.2 E-Stamp Vendor Module Description

4.1.2.1 User Interface Design

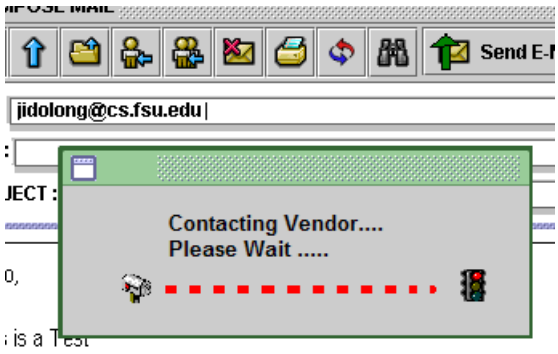


Figure 6, Vendor Module, Contacting Vendor

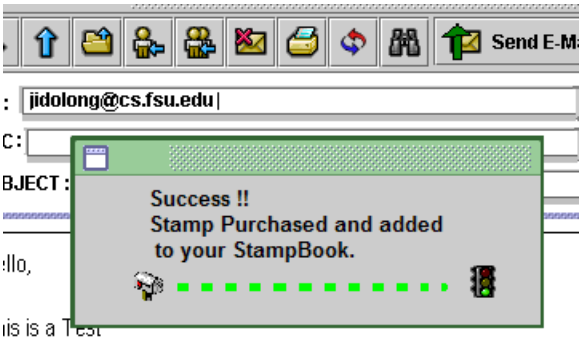


Figure 7, Vendor Module, Stamp Purchase Success

4.1.2.2 Description

The following figures show the user interfaces for this module. The interfaces are non interactive and only show the status of the stamp purchase process from the vendor.

4.1.3 E-Stamp Manager Module Description

4.1.3.1 User Interface Design

Figure 8, Manager Module, Manager Menu

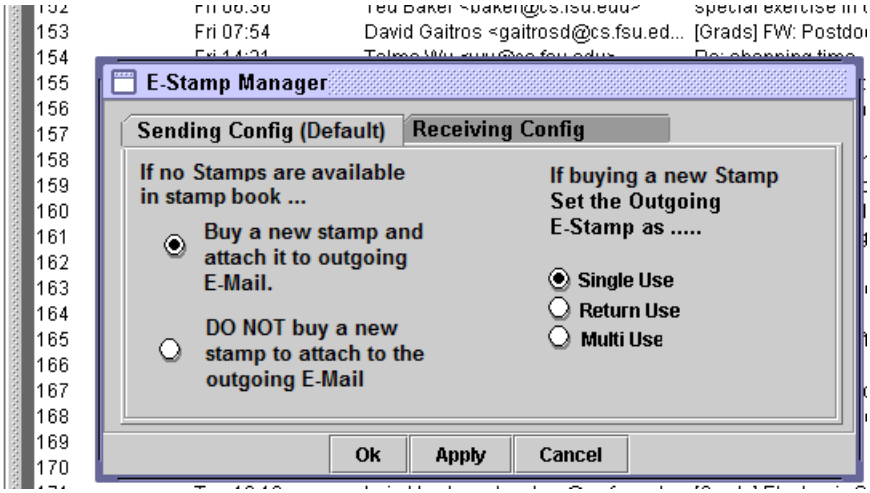


Figure 9, Manager Module, Sending Configuration

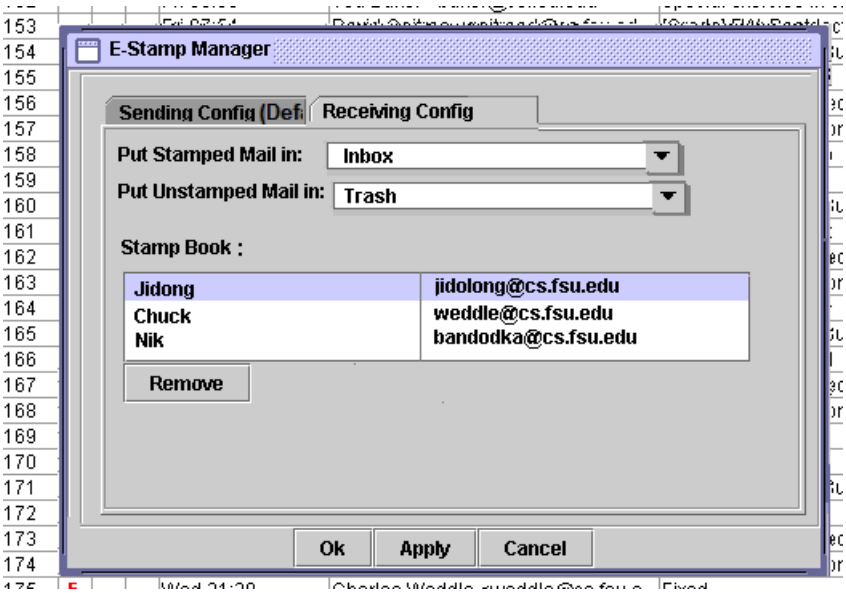


Figure 10, Manager Module, Receiving Configuration

4.1.3.2 Description

The User Interfaces allow the user to define actions to be performed by the software to automate the sending of stamped e-mail and receiving a stamped e-mail. The UI in figure 9 asks for actions to be performed by the system while sending out an e-mail. This UI provides the user four options. Based on the option selected by the user the system sends an e-mail with a single use stamp, a return use stamp, a multi use stamp or no stamp at all.

The UI in figure 10 deals with actions to be performed upon receiving a stamped e-mail. The stamped e-mail may be placed in a particular folder and unstamped mail may be placed in the same or different folder specified by the user. The stamp book is a collection of multi use or return

use stamps from people, from whom the user has approved receipt of e-mail. The user may revoke stamps by removing the incumbent stamps from his stamp book.

4.1.4 E-Stamp Sending E-Mail Module Description

4.1.4.1 User Interface Design

Figure 11, Sending E-Mail Module, E-Stamp Menu

4.1.4.2 Description

If the user has not already configured the E-Stamp Manager, the circled area will allow the user to select various stamp options. Alternatively it also allows the user to send unstamped email if a stamp is not already available. However if the E-Stamp Manager has been configured then the circled list box will highlight the default options.

4.1.5 E-Stamp Receiving E-Mail Module Description

4.1.5.1 User Interface Design

Figure 12, Receiving E-Mail, E-Mail with E-Stamps Indicator

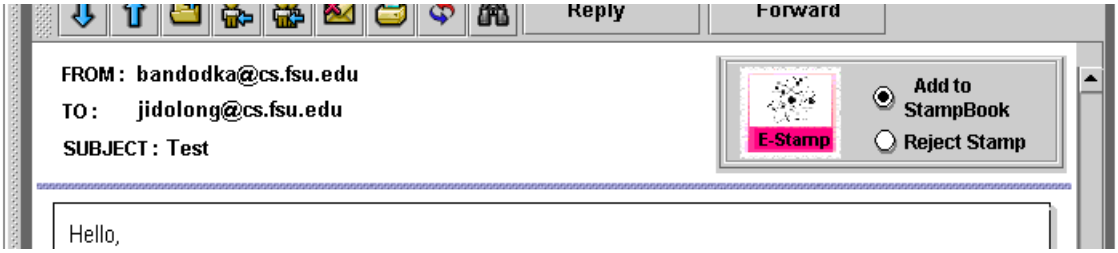


Figure 13, Receiving E-Mail, E-Stamp Options

4.1.5.2 Description

The UI for this module is limited to allow the user to reject the stamp or add it to his stamp book as shown in figure 13. By default the stamp will be added to the stamp book.

4.2 Process Interface

4.2.1 E-Stamp Configuration Process Description

The primary objective of this module is to obtain Vendor contact information and access password. It is assumed that the user has created an account with the vendor via the vendor's website. The user also provides information required to purchase stamp such as his e-mail address and credit card information. The software stores all this information in a database.

The information entered in the present module is used by the E-Stamp vendor module when it has to purchase an E-Stamp from the vendor via the internet.

4.2.2 E-Stamp Vendor Process Description

The Stamp Vendor Module is the software's communication module. It uses http to communicate with the Vendor over the internet. This module uses the vendor contact information from the E-Stamp Config module. It also takes data from the E-Stamp Manager Module regarding the reusability of the E-Stamp. The module then communicates with the vendor over the internet and procures a stamp. This stamp is stored in the Stamp Book to be attached with outgoing E-mail.

4.2.3 E-Stamp Authentication Process Description

The E-Stamp Authentication Module gets a stamp from the Receiving E-Mail module. This module obtains the vendor public key from its public key database. The module then verifies the stamps digital signature. If the signature is authentic then the stamp is deemed as Authentic. If the public key for the vendor is not available in the database, the module proceeds to obtain the vendors public key from a public directory. The authentication data is passed on to the receiving e-mail module.

4.2.4 E-Stamp Manager Process Description

The primary objective of the E-Stamp Manager module is to collect information to automate the process of e-stamp attachment to outgoing e-mail without persistent user interaction and to save received e-mail at a user convenient location. Options selected in the Sending configuration are passed on to the Sending E-Mail module along with persisting data from the E-Stamp Config module.

While receiving e-mail the software must sort stamped and unstamped e-mail into different folders or if the user chooses, place both in the same folder. This information is obtained from the current module and passed on to the receiving e-mail module. The user may also manage his stamp book by revoking stamps from certain senders or alternatively electing to keep those stamps. The revoked stamps are then logged in the revoked stamp database. This information is used by the receiving e-mail module and forms the basis upon which a stamped email whose e-stamp has been revoked is put in a folder separate from the valid stamp e-mail.

While sending an e-mail, the user enters the receivers e-mail address. The software checks the stamp book for available stamps for the receiver. The information contained in this stamp book is edited using the UI shown in figure 4.3.

4.2.5 E-Stamp Sending E-Mail Process Description

This module follows the following sequence of actions.

1. When either compose or reply action is performed the compose/edit e-mail interface comes up.
2. The module checks the settings in the Stamp Manager Module and accordingly sets the “Send E-mail with ….” Option.
3. When user performs the send action, the module checks stamp book for available stamps.
4. If no stamps are available and e-mail is selected to be stamped, the module passes receiver e-mail address to the stamp purchase module.
5. The module obtains stamp from the stamp purchase module attaches it to the e-mail and sends it.

4.2.6 E-Stamp Receiving E-Mail Process Description

The Receive E-Mail Module controls the receipt of E-Mail. It follows the following sequence of actions.

1. An E-mail is received. Check if E-Mail is Stamped or Unstamped.
2. If unstamped, use data provided in the Stamp Manager Module to place the e-mail in a particular folder.
3. If stamped, check if the stamp is present in the stamp book or has been revoked. If present in stamp book or revoked stamp database, place the e-mail in the folder specified in the Stamp Manager Module.
4. If stamp is not present in either, obtain stamp and pass this data to the authenticate stamp module.
5. If stamp is authentic place the e-mail in the folder specified in the Stamp Manager Module.
6. If the stamp is not authentic send message to the sender informing him/her that the message was not delivered.

7. For all e-stamped messages display the alphabet ‘E’ along with the e-mail in the inbox.

When the user opens the e-mail, he has the option of rejecting the stamp or adding the stamp to stamp book. After exiting this module the stamp will either be in the stamp book or logged as revoked.

5 Detailed Design

5.1 Module Detailed Design

5.1.1 E-Stamp Provider

5.1.1.1 Design

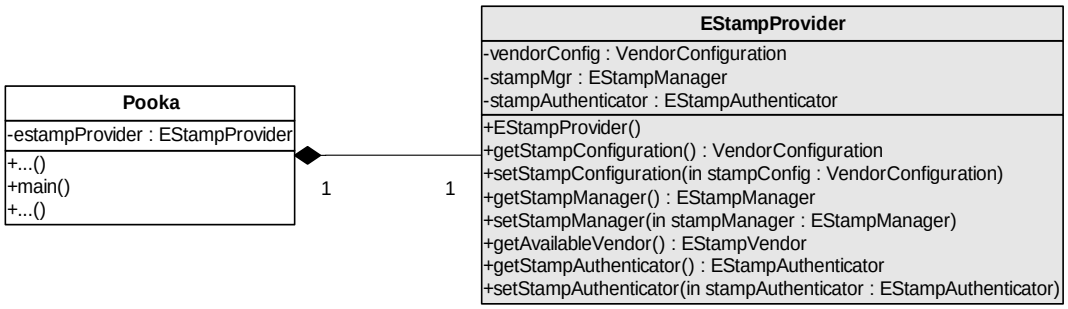


Figure 14, E-Stamp Provider Class Diagram

5.1.1.2 Design Description

The EStampProvider class deserves special attention because it is the central class to all of the other EStamp classes. As shown in the UML in the figure above, the Pooka class, which contains the main() method that is immediately called on Pooka application startup, instantiates the one and only EStampProvider class used in the system.

The EStampProvider class has a containment relationship with three other important EStamp Classes; VendorConfiguration, EStampAuthenticator, and EStampManger. A description of each one of these classes follows. Singleton instances of these three classes make up the most important attributes to the EStampProvider class. Please notice that these are shown as attributes in the EStampProvider UML class definition in the figure above.

One particular public method exposed on EStampProvider that needs explanation is getAvailableVendor(). This method is used to return to the caller an EStampVendor, which is described below, that will use the VendorConfiguration object to either get the vendor that the user has specified as a default or possibly a vendor of a specific name.

5.1.2 E-Stamp Configuration Module Detailed Design

5.1.2.1 Design

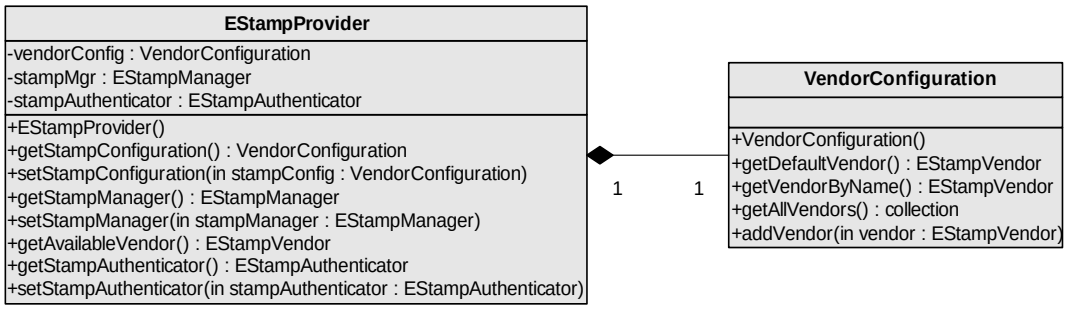


Figure 15, E-Stamp Configuration Module Class Diagram

5.1.2.2 Design Description

The EStampProvider constructor will instantiate one instance of the VendorConfiguration class. The VendorConfiguration class is a manager of EStampVendor objects. Internally, VendorConfiguration handles persisting EStampVendors and retrieving already configured EStampVendors in persistent storage.

Some of the more important methods exposed by VendorConfiguration are getDefaultVendor() and addVendor(). The method getDefaultVendor() will return to the caller an EStampVendor that is the default vendor specified by the user in the EStamp Config UI dialog. addVendor() takes as an argument a new EStampVendor that is to be persisted. The main caller of addVendor() is the EStamp Config UI dialog but it just as easily be called from any other part of the system if other ways to configure vendors became available.

5.1.3 E-Stamp Vendor Module Detailed Design

5.1.3.1 Design

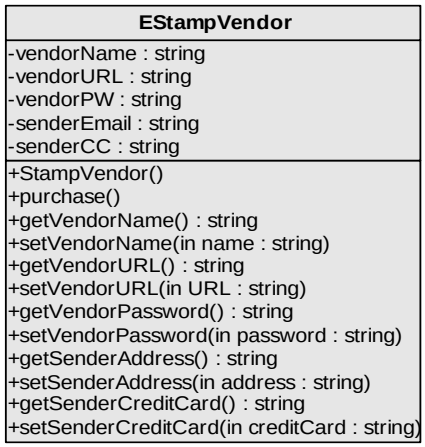


Figure 16, E-Stamp Vendor Module Class Diagram

5.1.3.2 Design Description

The EStampVendor class is instantiated when needed. As mention in the section above, the VendorConfiguration class handles the processing of the EStampVendor objects during system runtime.

The EStampVendor contains the vendor name, URL, and PW as private attributes as well as the sender email and credit card information. With this information, a vendor can be uniquely identified and used for purchasing a stamp. The methods exposed by this class do not warrant explanation, for they are all simply mutators and accessors for the private attributes.

5.1.4 E-Stamp Authentication Module Detailed Design

5.1.4.1 Design

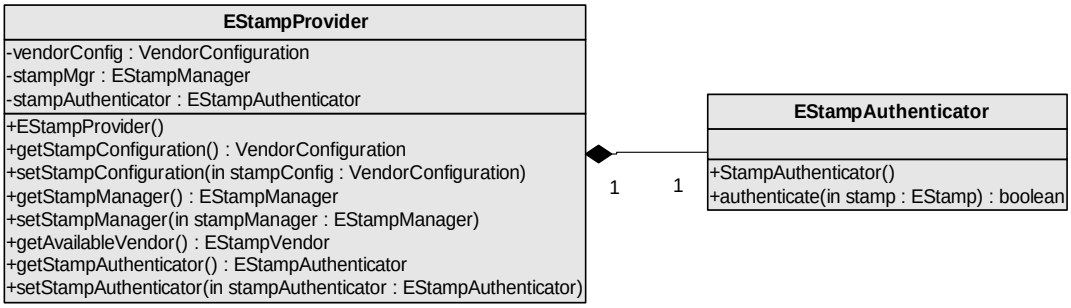


Figure 17, E-Stamp Authentication Module Class Diagram

5.1.4.2 Design Description

The EStampProvider constructor will instantiate one instance of the EStampAuthenticator class. The EStampAuthenticator class authenticates EStamps as the name implies. Internally, EStampAuthenticator handles the decrypting the EStamp with the vendor's public key.

Some of the more important method exposed by EStampAuthenticator is authenticate(). Again, as the name implies, this method with authenticate the EStamp passed in as a method parameter. The caller will examine the Boolean result returned from authenticate() as to whether this EStamp was valid or not.

5.1.5 E-Stamp Manager Module Detailed Design

5.1.5.1 Design

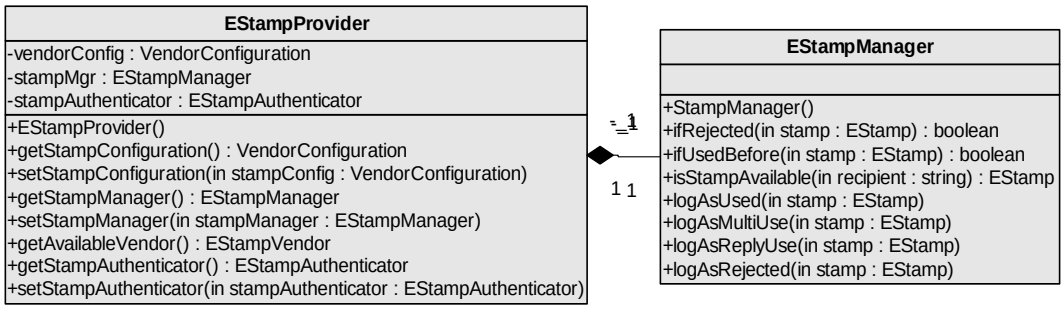


Figure 18, E-Stamp Manager Module Class Diagram

5.1.5.2 Design Description

The EStampProvider constructor will instantiate one instance of the EStampManager class. The EStampManager class is a manager of EStamp objects. Internally, EStampManager handles logging the EStamp with a particular attribute, for example multiuse. This class also takes care of the details of manipulating the EStamp database for such activities as determining if a stamp has been rejected already.

Some of the more important methods exposed by EStampManager are ifRejectedStamp() and isStampAvailable(). The method ifRejectedStamp() will determine if a stamp has already been rejected. The caller will examine the Boolean result returned to determine whether this EStamp has been rejected. isStampAvailable() will be used to determine if a valid EStamp already exists for use for the recipient passed in as a parameter. The call will receive the valid EStamp for that recipient if one is available.

5.1.6 E-Stamp Sending E-Mail Module Detailed Design

5.1.6.1 Design

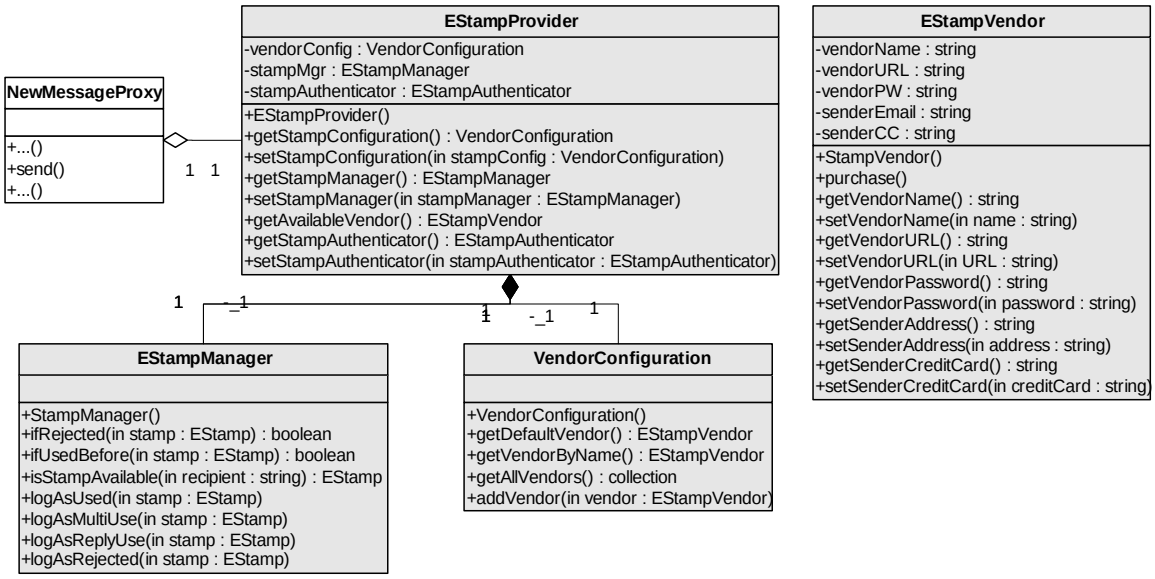


Figure 19, E-Stamp Sending E-Mail Module Class Diagram

5.1.6.2 Design Description

The Pooka class NewMessageProxy has a method called send() that handles the sending of a message. It is here that the EStampProvider will be referenced to retrieve the EStampManager singleton and call isStampAvailable(). If a stamp is available for the recipient specified in the new message, then it is used. If not, then EStampProvider is referenced to retrieve the VendorConfiguration singleton and call getDefaultVendor(). Once the default EStampVendor has been retrieved, then purchase() will be called to actual get a stamp.

Notice that NewMessageProxy only has a shared reference to EStampProvider, which is indicated in the above UML diagram by the white diamond. The EStampProvider is contained in the Pooka class.

5.1.7 E-Stamp Receiving E-Mail Module Detailed Design

5.1.7.1 Design

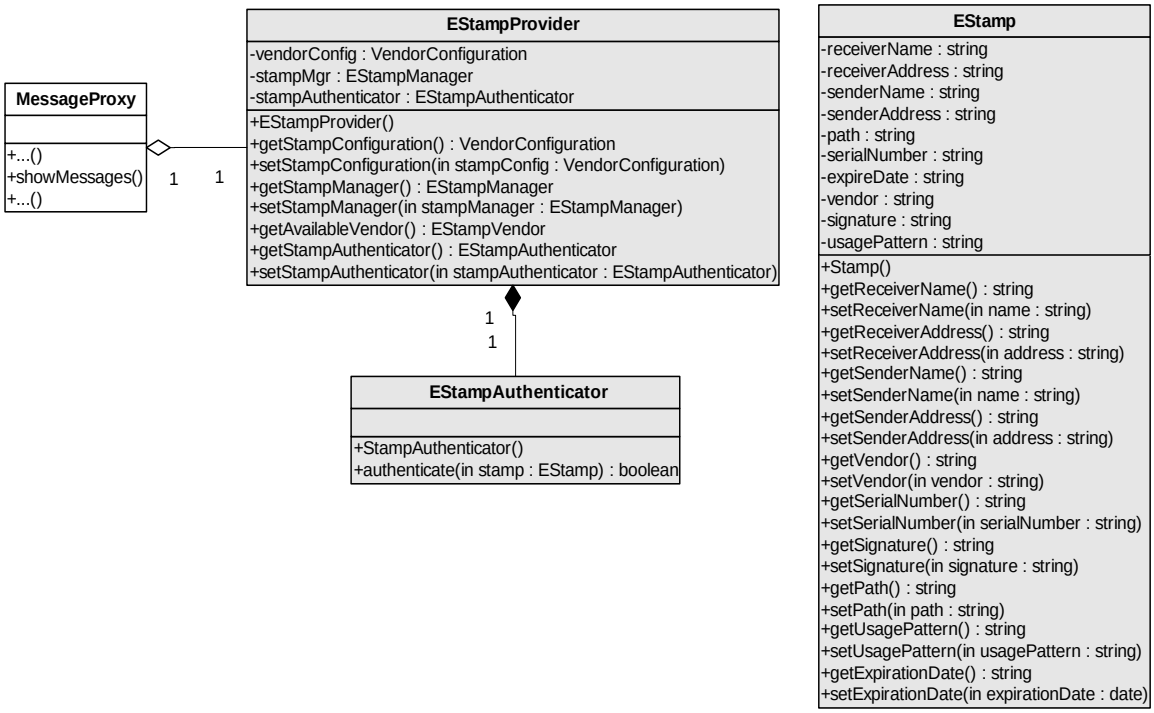
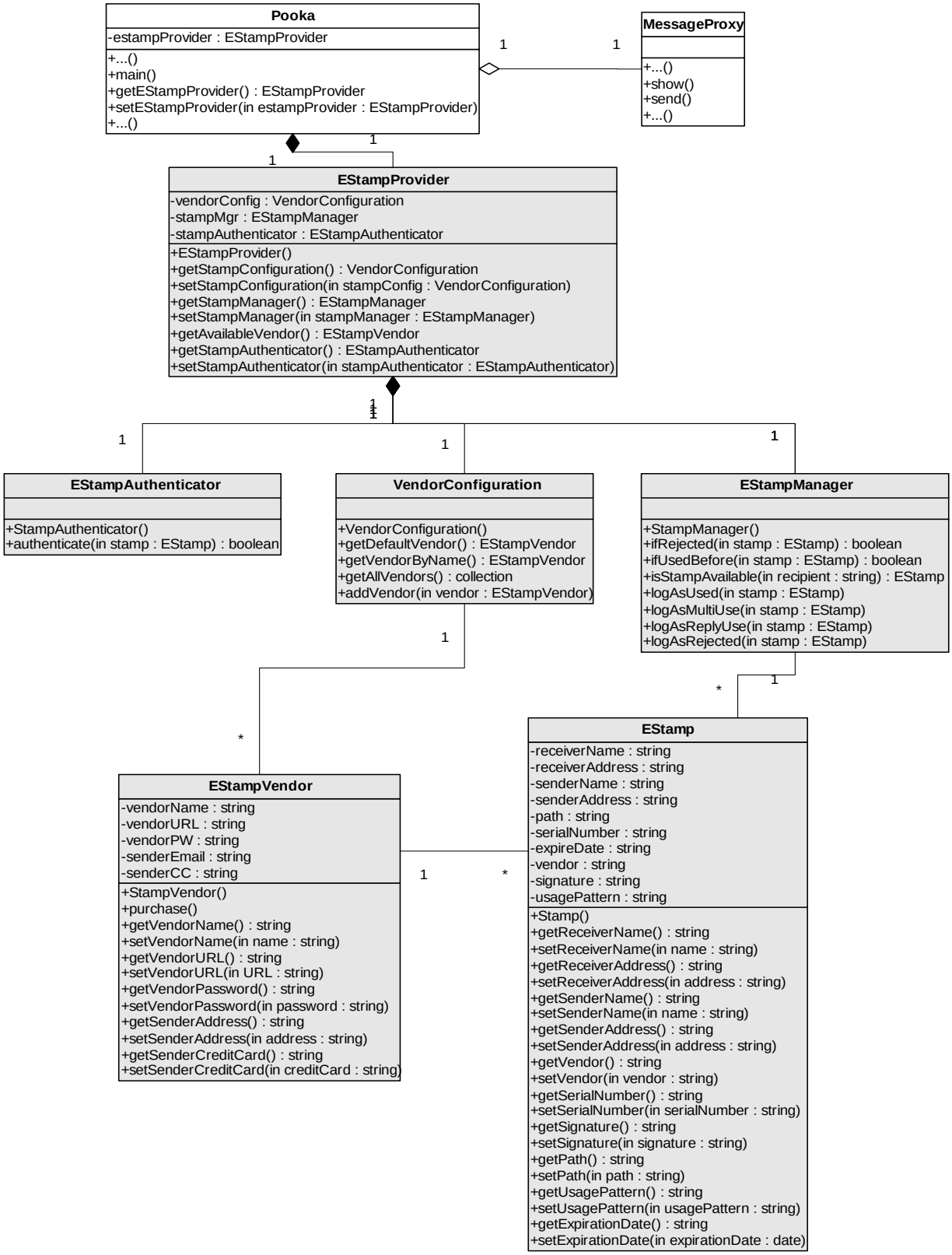


Figure 20, E-Stamp Receiving E-Mail Module Class Diagram

5.1.7.2 Design Description

The Pooka class MessasgeProxy has a method called showMessages() that is used to display the messages within the UI. It is here that the EStampProvider will be referenced to retrieve the EStampAuthenticator singleton so that the unauthenticated EStamps contained on new email may be determined if they are valid or not using the authenticate method.

Appendix A – EStamp Class Diagram


Pooka
-estampProvider : EStampProvider
+...()
+main()
+getEStampProvider() : EStampProvider
+setEStampProvider(in estampProvider : EStampProvider)
+...()
MessageProxy
+...()
+show()
+send()
+...()
1
1
1
1
EStampProvider
-vendorConfig : VendorConfiguration
-stampMgr : EStampManager
-stampAuthenticator : EStampAuthenticator
+EStampProvider()
+getStampConfiguration() : VendorConfiguration
+setStampConfiguration(in stampConfig : VendorConfiguration)
+getStampManager() : EStampManager
+setStampManager(in stampManager : EStampManager)
+getAvailableVendor() : EStampVendor
+getStampAuthenticator() : EStampAuthenticator
+setStampAuthenticator(in stampAuthenticator : EStampAuthenticator)
1
1
1
1
1
EStampAuthenticator
+StampAuthenticator()
+authenticate(in stamp : EStamp) : boolean
VendorConfiguration
+VendorConfiguration()
+getDefaultVendor() : EStampVendor
+getVendorByName() : EStampVendor
+getAllVendors() : collection
+addVendor(in vendor : EStampVendor)
EStampManager
+StampManager()
+ifRejected(in stamp : EStamp) : boolean
+ifUsedBefore(in stamp : EStamp) : boolean
+isStampAvailable(in recipient : string) : EStamp
+logAsUsed(in stamp : EStamp)
+logAsMultiUse(in stamp : EStamp)
+logAsReplyUse(in stamp : EStamp)
+logAsRejected(in stamp : EStamp)
1
*
1
*
EStampVendor
-vendorName : string
-vendorURL : string
-vendorPW : string
-senderEmail : string
-senderCC : string
+StampVendor()
+purchase()
+getVendorName() : string
+setVendorName(in name : string)
+getVendorURL() : string
+setVendorURL(in URL : string)
+getVendorPassword() : string
+setVendorPassword(in password : string)
+getSenderAddress() : string
+setSenderAddress(in address : string)
+getSenderCreditCard() : string
+setSenderCreditCard(in creditCard : string)
1
*
EStamp
-receiverName : string
-receiverAddress : string
-senderName : string
-senderAddress : string
-path : string
-serialNumber : string
-expireDate : string
-vendor : string
-signature : string
-usagePattern : string
+Stamp()
+getReceiverName() : string
+setReceiverName(in name : string)
+getReceiverAddress() : string
+setReceiverAddress(in address : string)
+getSenderName() : string
+setSenderName(in name : string)
+getSenderAddress() : string
+setSenderAddress(in address : string)
+getVendor() : string
+setVendor(in vendor : string)
+getSerialNumber() : string
+setSerialNumber(in serialNumber : string)
+getSignature() : string
+setSignature(in signature : string)
+getPath() : string
+setPath(in path : string)
+getUsagePattern() : string
+setUsagePattern(in usagePattern : string)
+getExpirationDate() : string
+setExpirationDate(in expirationDate : date)