INTEGRATION TECHNOLOGIES
INTARSYS

Java Message Manager

Technical Guide

Version 1.00.005a

Revision History

Date	Version	Description	Author
2/07/03	1.00.000	Initial	INT/Navin Barath
2/10/03	1.00.000	Add Page Numbers, Hide Grid Lines, Lock Document	INT/Navin Barath
2/12/03	1.00.001	Update for latest release	INT/Navin Barath
2/26/03	1.00.002	Update with cryptography changes	INT/Navin Barath
2/27/03	1.00.003	Reformat Figures and Tables	INT/Navin Barath
2/28/03	1.00.003	Update with Cryptography and Service creation information	INT/Navin Barath

Table of Contents

Table of Figures and Tables

1 Introduction

This document includes the requirements and steps that are necessary for the installation of the Java Message Manager application and service. It is recommended that you use the administrator account for the operating system or one that has administrator privileges, on a server in which you intend to install this software before you follow the steps in this document.

It is also recommended that you read this entire document before installing the application. It contains information on customizing the application for your use.

If you have any questions on the contents of this document, please contact FCCSC Help Desk at (954) 761-1170 or email **help@fccsc.org**. Once again, we recommend that you read this document thoroughly, before you begin the installation.

2 System Requirements and Recommendations

2.1 Hardware Requirements and Recommendations

CPU Processing

Minimum : PIII 350MHz [1] Processor, 256MB[2] RAM

Recommended : PIII 1GHz[3] Dual Processor, 1GB[4] RAM

Storage

Application 10MB[2]

JDK1.4 126MB[2]

Please note that the application may run on less that the minimum hardware requirements stated above, but may not be as efficient as desired.

2.2 Software Requirements and Recommendations

Table 1 - Software Requirements below shows software used during application testing. This is therefore the software level presently supported.

	Software
Operating System	WinNT Sp6a, Win2000 Server SP2, WinXP Pro.
JDK	JDK1.4

Table 1 - Software Requirements

[1] MHz denotes Megahertz
[2] MB denotes Megabytes
[3] GHz denotes Gigahertz
[4] GB denotes GigaBytes

3 Configuration and Installation

3.1 Configure TCP/IP

Configure your EntireX Broker local TCP/IP settings. Create an entry in the HOSTS and SERVICES files on your operating system for EntireX Broker. These files can be found in your WINDOWS\SYSTEM32\DRIVERS\ETC directory on the Windows Operating System.

Following is an example of the HOSTS and SERVICES configured for EntireX Broker.

HOSTS

Define the IP address of your Broker Node followed by the name for that node. This will be used by the application when making Broker connections.

```
100.100.100.100  etb100                                  # Broker Node
```

Figure 1 - Sample Host entry

SERVICES

Define the port number for the Broker service here. These names will be used later on when you define the properties f each message type.

```
Etb100          3007/tcp #Entire X Broker - Message Manager
```

Figure 2 - Sample Services entry

3.2 Install JDK

Download jdk1.4 from Sun Microsystems website at **www.sun.com.**

Install the jdk as directed.

3.3 Download the Application

Download Java Message Manager software from FCCSC website at **http://www.fccsc.org/fccsc/webservices/software/downloads/messageman/jman-v1.00.005a-deploy.zip** or other alternate location provided to you by FCCSC Support.

3.4 Extract the application

The application is provided as a complete project compressed into one .zip file. Once downloaded extract that .zip file into a location of your choice. The directory structure that will be created is shown in Figure 3 - Application Directory Structure.

```
fccsc
|——  manager
      |——  backups      - backups during development
      |——  classes      - class files created and used
      |——  cryptography - files require to perform encryption and decryption
      |——  jars         - java archives used
      |——  javadocs     - java documentation
      |——  logs         - logs files directory
      |——  models       - UML models in Rational Rose format
      |——  properties   - application properties files
      |——  sources      - source code
      |——  templates    - templates for institutions html response
```

Figure 3 - Application Directory Structure

The root directory of fccsc\manager contains the startup batch files for both the Message Manager and the Message Tester applications. These are "*run_MessageManager.bat*" and "*run_MessageTester.bat*" respectively.

3.5 Configure Properties

All properties files are found in the `fccsc\manager\properties` directory. There are properties files for Message Manager, Mail, Log4j, cryptography and one for each message type handled by Java Message Manager. A property may be commented using "#" symbol. Each properties file's properties are described in detail after illustrating the properties file.

3.5.1 Message Manager properties

File : manager.properties

Defines the ports and maximum values for concurrent connections to/from Java Message Manager and the Tester applications.

```
# (mm) message manager configuration file

manager.listener.port    = 3007
manager.listener.backlog = 1000

tester.listener.port    = 5200
tester.listener.backlog = 1000
```

Figure 4 - Message Manager Properties Sample

manager.listener.port	The local TCP/IP port that the Java Message Manager listens on for requests. Modify this port if 3007 is being used by other services on your computer and if this has been approved by the FACTS administration run by the Florida Center of Academic Advising for Student (FCAAS).
manager.listener.backlog	The maximum length of the server socket queue. This value defines how many concurrent connections can be maintained by the Java Message Manager.
tester.listener.port	The local TCP/IP port that the Java Message Manager Tester listens on for replies to requests. Modify this port if 5200 is being used by other services on your computer.
tester.listener.backlog	The maximum length of the server socket queue. This value defines how many concurrent connections can be maintained by the Java Message Manager Tester.

3.5.2 Mail properties

File : mail.properties

Defines the default email properties. The Admissions process is the only one that utilizes properties defined here.

```
# mail notification configurations
#
mail.transport.protocol = smtp
mail.host               = smtp.yourinstitution.edu
mail.user               = admissions@yourinstitution.edu
mail.password           = bingo
mail.from               = factsadmissions@yourinstitution.edu
mail.address.to.1       = name1@yourinstitution.edu
# mail.address.to.2       = name2@yourinstitution.edu
# mail.address.cc.1       = cc1@yourinstitution.edu
# mail.address.cc.2       = cc2@yourinstitution.edu
# mail.address.bcc.1      = bcc1@yourinstitution.edu
# mail.address.bcc.2      = bcc2@yourinstitution.edu
```

Figure 5 - Mail Properties Sample

mail.transport.protocol	The mail protocol being used for sending email.
mail.host	The SMTP host name or IP.
mail.user	The user account name to the SMTP server for authentication.
mail.password	The password for the user account name to the SMTP server for authentication.
mail.from	The default from email address or sender email address.
mail.address.to.1	The default to email address or recipient email address. To add more email addresses for the to, use enumeration E.G. mail.address.to.1 and mail.address.to.2 etc.

mail.address.cc.1	The default carbon copy email address or carbon copy recipient email address. To add more email addresses for the cc, use enumeration E.G. mail.address.cc.1 and mail.address.cc.2 etc.
mail.address.bcc.1	The default blind carbon copy email address or recipient blind carbon copy email address. To add more email addresses for the bcc, use enumeration E.G. mail.address.bcc.1 and mail.address.bcc.2 etc.

3.5.3 Log4J properties

File : log4j.properties

Defines the applications logging properties.

```
# attach appender MM to root
log4j.rootLogger=ERROR, MM

# set logging level for package/s
log4j.logger.fccsc=ERROR

# configure the file appender's information
log4j.appender.MM.File=logs/jman.logs
log4j.appender.MM.Append=true
log4j.appender.MM=org.apache.log4j.DailyRollingFileAppender
log4j.appender.MM.DatePattern='.'yyyy-MM-dd
log4j.appender.MM.layout=org.apache.log4j.PatternLayout
log4j.appender.MM.layout.ConversionPattern=%d [%-8t] %-5p %c - %m%n
```

Figure 6 - Log4J Properties Sample

log4j.rootLogger	Define the default logging level. Possible values are DEBUG, INFO, WARN, ERROR and FATAL. Also define an output destination name for logging by attaching an appender to the logger.

log4j.logger.fccsc	Set the default message logging level for a package. In the case above we are setting the default logging level to the highest (DEBUG) for the top package FCCSC. Possible values are DEBUG, INFO, WARN, ERROR and FATAL.
log4j.appender.MM.File	The actual location and file name of the log file specified as a relative path.
log4j.appender.MM.Append	Append to log file if TRUE. Clears log file before logging if FALSE
log4j.appender.MM	Appender class name.
log4j.appender.MM.DatePattern	Appender pattern to be used for date.
log4j.appender.MM.layout	Appender pattern layout class.
log4j.appender.MM.layout.ConversionPattern	Appender conversion pattern layout.

See your Log4J documentation for more information at **http://jakarta.apache.org/log4j/docs/**.

3.5.4 Message properties

3.5.4.1 Common Message Properties

Each Message Type has its own properties file. The message properties file is named with the message type as part of the name, therefore constructed as follows: "`message`"+ *message type* + "`properties`", delimited by "`.`".

Following are the properties that are *common* to the message properties files. These must be defined within each `message.`*`messagetype`*`.properties` file. *Note* that we use the Verify message file "`message.VERIFY.properties`" in the following illustration to describe common message properties.

```
# message configuration file
message.object        = fccsc.manager.data.process.Verify
entirex.brokerid      = etb100:3007
entirex.serverclass   = UTLI01P1
entirex.servername    = DEVL
entirex.service       = STSI01N0
entirex.userid        = NONE
entirex.uow           = false
entirex.uow.timeout   = 30S
entirex.uow.maxlength = 30000
```

Figure 7 - Common Message Properties Sample

message.object	Message class processor. The class that will process this message type.
entirex.brokerid	EntireX Broker ID and port. This must be the same as those defined when configuring TCP/IP parameters for EntireX Broker in the HOSTS and SERVICES files.
entirex.serverclass	ExtireX Broker Class.
entirex.servername	ExtireX Broker Server.
entirex.service	ExtireX Broker Service.
entirex.userid	ExtireX Broker User Id.
entirex.uow	ExtireX Broker Unit of Work Boolean value (TRUE/FALSE). Defines if this message type is a UOW or not.
entirex.uow.timeout	ExtireX Broker UOW timeout value. See EntireX Broker documentation for more information.
entirex.uow.maxlength	ExtireX Broker UOW maximum length of a message.

3.5.4.2 Specific Message Properties

The following properties are specific to a message type

ADMISSION

mail.notification	This enables or disables email notifications for the admissions response message process. Options: ON or OFF

LOCALSHOP

guest.student.id	Guest Audit - this value will be used when the StudentId contains "GST".

3.5.5 Cryptography properties

```
#
# cryptography configuration
#
crypto.path.certificate = cryptography/javaman.crt
crypto.path.privatekey  = cryptography/javaman.pem
crypto.path.passphrase  = cryptography/passphrase.pph
```

Figure 8 - Cryptography Properties Sample

cryptography.path.certificate	This is the relative path (relative to the application's root directory) and filename for the public key/certificate. It is used to encrypt a message.
cryptography.path.privatekey	This is the relative path (relative to the application's root directory) and filename for the private key. This is used to decrypt a message that was previously encrypted using the relative public key.
cryptography.path.passphrase	This is the relative path (relative to the application's root directory) and the filename for the encrypted pass phrase file.

4 Starting Java Message Manager as an application

To start the Java Message Manager as an application, execute the batch file "`run_MessageManager.bat`" by double-clicking on it or, by using the MS-DOS command prompt. It can be found in your application root directory `fccsc/manager`. If you want to start Java Message Manager as a service, see section 7 Creating the Windows Service.

Note: **You must modify the JAVA_HOME variable to your JDK home directory before you start Java Message Manager**.

This will start a java process that will listen on a port defined in the `message.properties` file described above in a MS-DOS command prompt window as show in Figure 9 - Message Manager Console.

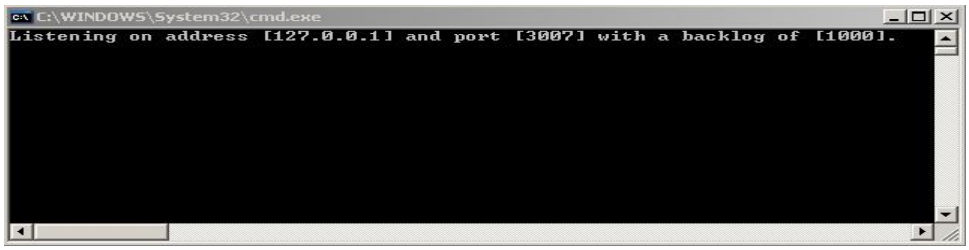


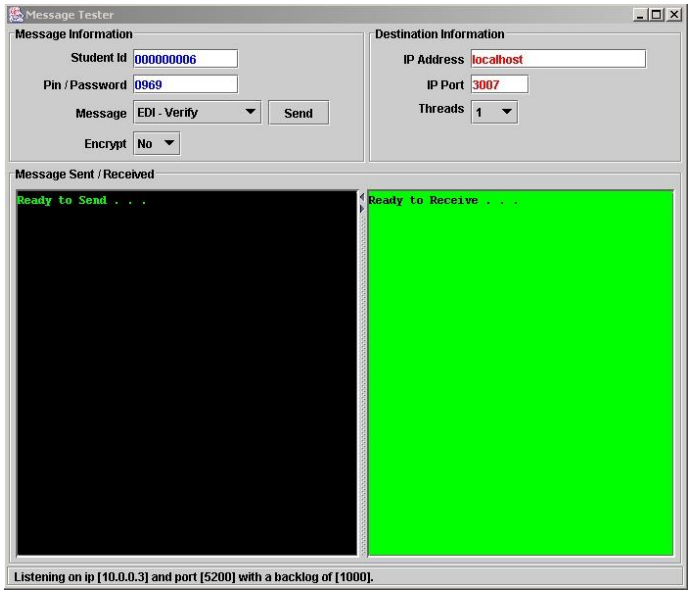
Figure 9 - Message Manager Console

5 Testing the Application

In the application root directory fccsc\manager, you will find the batch file "`run_MessageTester.bat`"

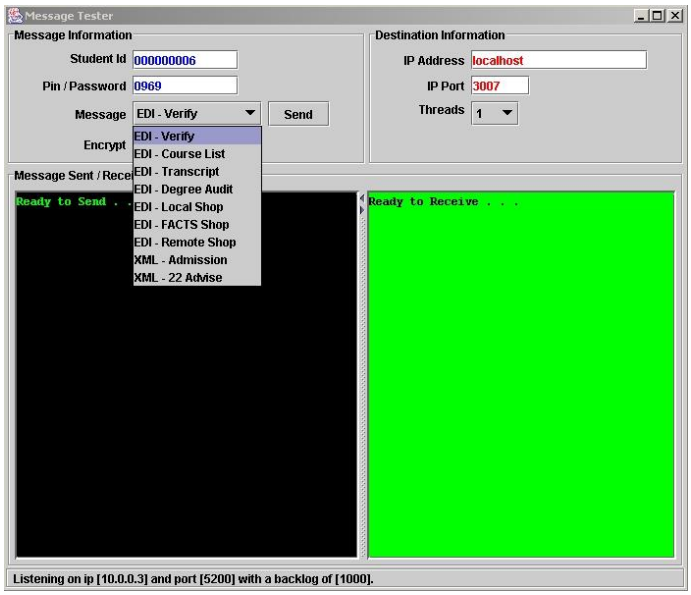
Note: **You must modify the JAVA_HOME variable to your JDK home directory before you start Message Tester**.

To test the Java Message Manager, you are provided with a "tester" also in the form of a batch file "`run_MessageTester.bat`". To start the Message Tester application, execute the batch file. You will be presented with the user interface show in Figure 10 - Message Tester UI.



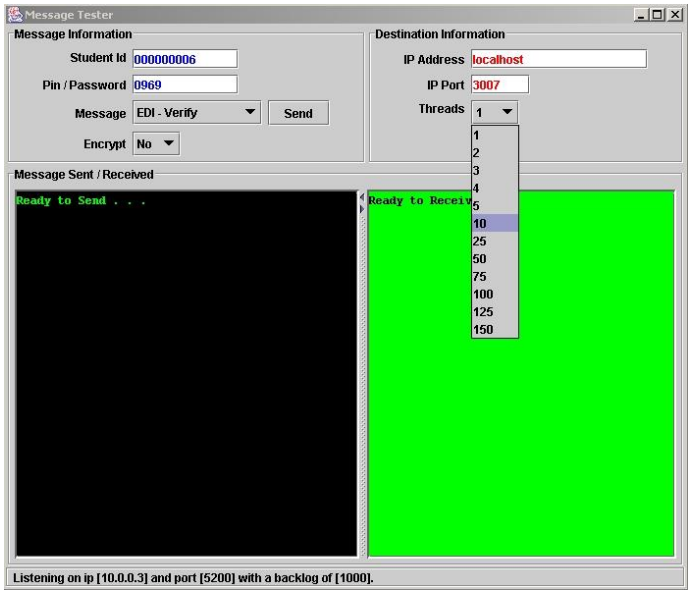
On the left half of the window there are sample requests to be sent to Java Message Manager. These are hard coded requests within the Message Tester class files. This half acts as the Message Director, sending requests to Java Message Manager. The right half of the window listens for responses from Java Message Manager and displays these responses.

Figure 10 - Message Tester UI



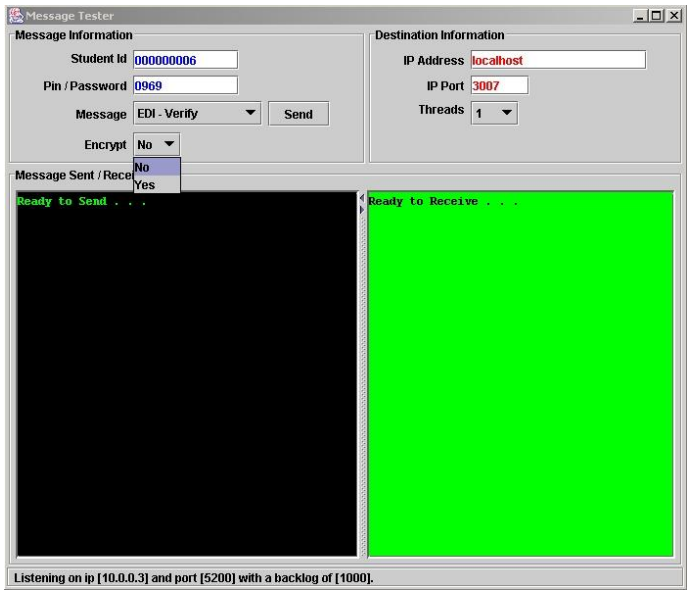
Choose the Message Type you want to test as shown.

Figure 11 - Message Tester - Choosing the Message Type



Select the number of threads you want to test. Message Tester will generate as many concurrent requests to Java Message Manager as the number of threads you have selected to test.

Figure 12 – Message Tester – Selecting Number of Threads



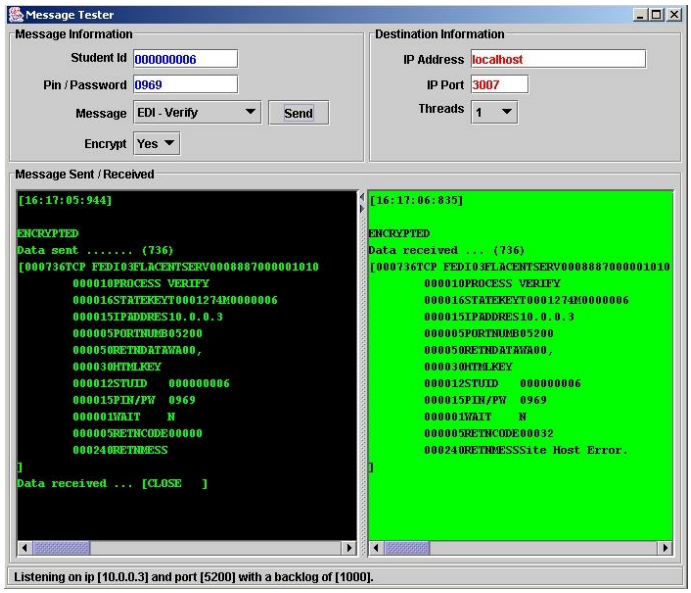
Select if this message must be encrypted and decrypted. A choice of "Yes" for the field labeled "Encrypt" will encrypt the message request and decrypt the message response to/from the Java Message Manager.

Figure 13 - Select Encrypted Message Test

Modify the *IP Address*, and *IP Port* if necessary. For *IP Address*, use the IP Address of the computer running Java Message Manager. For *IP Port* see `manager.listener.port` property of your `manager.properties` file.

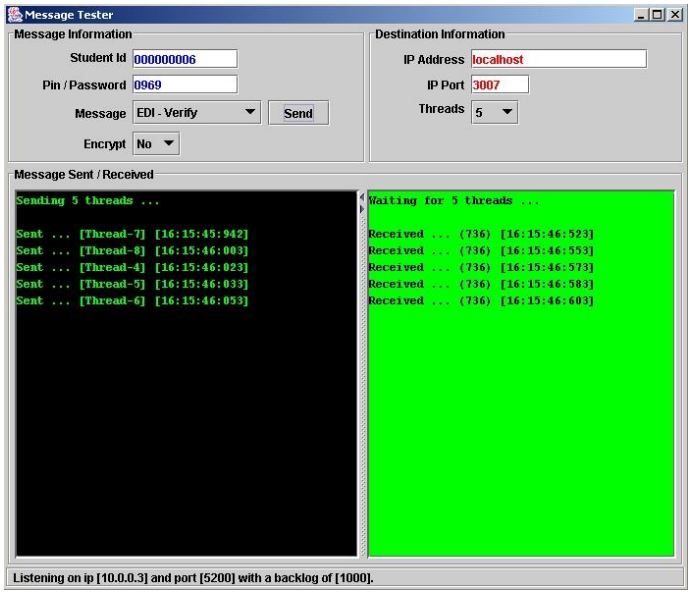
Click the "`Send`" button after selecting the message type, the number of threads you want tested and if you want encryption/decryption tested.

Figure 14 - Message Tester - Single Thread Request/Response and Figure 15 - Message Tester - Multi Thread Request/Response are illustrations of the two types of output when testing.



If one thread was selected the Message Tester will display the request on the left side and the response data on the right.

Figure 14 - Message Tester - Single Thread Request/Response



If multiple threads were selected, Message Tester shows the thread number, the date and time the request was sent on the left side. The response is on the right side showing the length of data received in parentheses "( )" followed by the date and time it was received. The data for the request and response are not shown.

Figure 15 - Message Tester - Multi Thread Request/Response

6 Cryptography

6.1 Public/Private Key Infrastructure (PKI)

Public and Private keys are used to encrypt and decrypt data. These keys are always generated as pairs by a Root Certificate Authority. The private key is, what its name tells us, it is private, and therefore for your eyes only. The private key has a pass phrase. This is a password for unlocking the private key. Without this password you will not be able to use the private key. It is used to decrypt data that was encrypted using the public key.

The public key is used by other parties to encrypt data that they want to send to you, so you need to share this key. They encrypt data using your public key and a generated session key, and you will decrypt it using your private key.

When you send encrypted data to someone, using PKI, you would normally encrypt data using their public key. However, in the FACTS network, Java Message Manager encrypts data using the session key which it saves after decrypting the request from Message Director, which was itself encrypted when Message Director encrypted the request. This is a specification of the FACTS network.

6.2 How does FACTS know when to use Encryption/Decryption

The Facts network of servers is capable of performing encryption/decryption on messages it sends and receives from Institutions. The Facts database has an entry in a table for each message type that an Institution can work with and now has two more items, one that identifies if that institution is capable of working with encrypted data for each message type, and the second that identifies the public key or certificate

When a request (Facts Message) is received by Message Director (Facts network), a database lookup determines if the Institution that the request is going to, can perform encryption/decryption. If they can, that Institution's public key which is also loaded in the FACTS database, is loaded into memory, and data is encrypted using it. The only private key that can decrypt that data is that which is associated with the public key used to encrypt the data.

6.3 What You have to do to make Encryption/Decryption work

FCAAS will generate a public/private key pair with a pass phrase for you. It will be signed with FCAAS being the root authority.

- **You must copy or move the public key to the `cryptography` directory.**
- **You must copy or move the private key to the `cryptography` directory.**

- **You must encrypt the pass phrase into a file and then copy or move the encrypted pass phrase file to the `cryptography` directory.** See section 6.4 The PassPhraseEncryptor Tool for how to encrypt the pass phrase.
- **You must modify the `cryptography` properties file.** section 3.5.5 Cryptography properties.
- **Restart your application or service.**

6.4 The PassPhraseEncryptor Tool

This tool is used to encrypt a text pass phrase and save the encrypted pass phrase in a file. The file is later used by Java Message Manager when decrypting data.

You must encrypt the pass phrase using the *PassPhraseEncryptor* tool. The tool is made up of a set of java classes compressed into a java archive, for which a batch script to run it is provided. Java Message Manager will always expect to read an encrypted pass phrase file to get the pass phrase, in order to unlock the private key, which in turn is used to attempt to decrypt data. Figure 16 - Pass Phrase Encryptor batch script shows the contents of the *PassPhraseEncryptor* batch script `run_PassPhraseEncryptor.bat` and following that are steps to use it.

```
@echo off
      set JAVA_HOME=c:\j2sdk1.4.0_01
      set JARS=jars
      set CP=.
      set CP=%CP%;%JARS%/fccsc-manager-encryptor.jar
      start "Pass Phrase Encryptor" /min %JAVA_HOME%\bin\javaw -cp %CP%
fccsc.manager.apps.passphrase.PassPhraseEncryptor
```

Figure 16 - Pass Phrase Encryptor batch script

- Edit run_PassPhraseEncryptor.bat script with a text editor.
- Change JAVA_HOME to your JDK home.
- Save it.
- Either double click on the batch file or execute it in a MS-DOS command prompt window.

You will then be presented with user interface as shown in Figure 17 - Pass Phrase Encryptor UI.

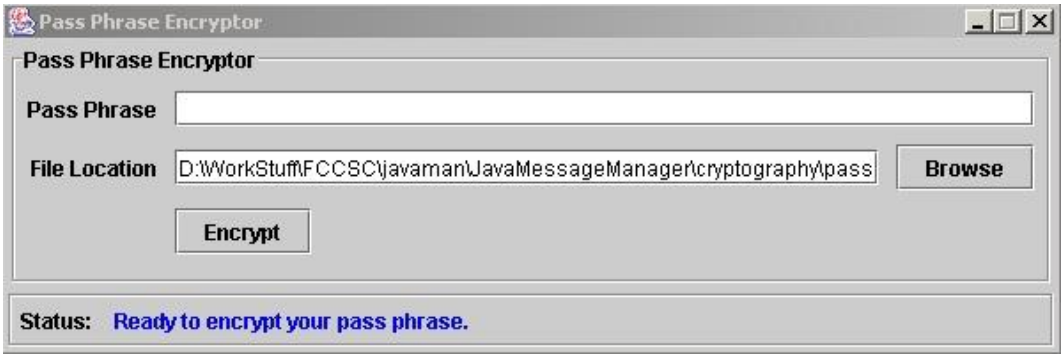


Figure 17 - Pass Phrase Encryptor UI

- Enter the text pass phrase in the field labeled "`Pass Phrase`"
- Click on "`Browse`" to set the location and file name for the encrypted pass phrase file. Note that the Java Message Manager requires that the encrypted pass phrase file be in the `cryptography` directory.
- After you have the correct location and file name, click the button labeled "`Encrypt`". This will encrypt the text pass phrase into the file you have specified.
- Now that you have the encrypted pass phrase file you need to update the cryptography properties file. Specify the private key, public key and the name of the encrypted pass phrase file. See section 3.5.5 Cryptography properties.

7 Creating the Windows Service

The service creation tool is a third party tool that is specifically used to create Windows services to run java programs. Please read `JMM_License.txt` license file. A batch script file, `JMM_createService.bat` is provided to easily create this service. Modify the variables in this script as described below in Table 2 – Service Variables for your review to prepare to create your Java Message Manager Service:

Variable	Description / Use / Changes
TITLE	The title for the service. This is the name that you will see in your services manager. The title for your service must be enclosed in double quotes if it includes spaces.
JAVA_HOME	Specify your JDK home directory. E.G. c:\jdk1.4
APPROOT	This is the root directory of the Java Message Manager application. If you are creating this service on Windows NT you will need to change %CD% to the root directory for the application. E.G C:\JMM\fccsc\manager Other Windows platforms, W2K and WINXP etc. will require no change to this variable as they will correctly resolve `%CD%` to the current directory.
LOUT	This file is used for standard output messages from the jvm. These messages are informational. If you wish, you may change the name of the file.
LERR	This file is used for standard error from the jvm. These messages are errors that have occurred. If you wish, you may change the name of the file.

Table 2 – Service Variables for your review

After you have reviewed these variables and made your changes to `JMM_createService.bat`, save this file. You are now ready to execute this file to create your service.

Either double click on the batch script or execute it in a MS-DOS command prompt window. You should now have your Windows service. Check your Windows Services Manager for the service. It will be created with automatic startup. Test it by starting it then running the Message Tester. See section 5 Testing the Application for information on using the Message Tester. Stop your service. Check that it starts and stops in a timely manner without errors. Look at your Windows Application and System Event log for errors. Check you standard output and error log files specified with LERR and LOUT variables (see Table 2 – Service Variables for your review).

Contents of JMM_createService.bat

```
rem -------------------------------------------------
rem The service display name.
rem -------------------------------------------------
set TITLE="FCCSC Java Message Manager"



rem -------------------------------------------------
rem The root directory of the Java VM you want to use.
rem -------------------------------------------------
  set JAVA_HOME=D:\j2sdk1.4.0_01



rem -------------------------------------------------
rem The root directory of the application
rem -------------------------------------------------
  set APPROOT=%CD%



rem -------------------------------------------------
rem The location of all application dependent jar files.
rem -------------------------------------------------
set JARS=%APPROOT%\jars



rem -------------------------------------------------
rem The classpath to be use by the application.
rem -------------------------------------------------
set CP=%APPROOT%
set CP=%CP%;%JARS%\fccsc-manager.jar
set CP=%CP%;%JARS%\fccsc-manager-encryptor.jar
set CP=%CP%;%JARS%\intarsys-delta.jar
set CP=%CP%;%JARS%\log4j-1.2.7.jar
set CP=%CP%;%JARS%\entirex-521-rt.jar
set CP=%CP%;%JARS%\entirex-521.jar
```

```
set CP=%CP%;%JARS%\javamail1.3-activation.jar
set CP=%CP%;%JARS%\javamail1.3-mail.jar
set CP=%CP%;%JARS%\jaxp1.0-jaxp.jar
set CP=%CP%;%JARS%\jaxp1.0-parser.jar
set CP=%CP%;%JAVA_HOME%\lib\tools.jar


rem ---------------------------------------------------
rem Service Parameter - The Java VM library location
rem ---------------------------------------------------
set JVM=%JAVA_HOME%\jre\bin\server\jvm.dll


rem ---------------------------------------------------
rem Service Parameter - The class path to use.
rem ---------------------------------------------------
set JCP=-Djava.class.path=%CP%


rem ---------------------------------------------------
rem Service Parameter - The library path to use.
rem ---------------------------------------------------
set JLP=-Djava.library.path=%APPROOT%\cryptography


rem ---------------------------------------------------
rem Service Parameter - The main class name.
rem ---------------------------------------------------
set CMAIN=fccsc.manager.MessageManager


rem ---------------------------------------------------
rem Service Parameter - The shutdown class name.
rem ---------------------------------------------------
set CSHUT=fccsc.manager.MessageManager
```

```
rem ------------------------------------------------
rem Service Parameter - The shutdown class METHOD name.
rem ------------------------------------------------
set MSHUT=doShutdown


rem ------------------------------------------------
rem Service Parameter - Log file for System.out calls.
rem ------------------------------------------------
set LOUT=%APPROOT%\logs\mm.out


rem ------------------------------------------------
rem Service Parameter - Log file for System.err calls.
rem ------------------------------------------------
set LERR=%APPROOT%\logs\mm.err


rem ------------------------------------------------
rem The final service command using all of the above.
rem ------------------------------------------------
JMM.exe -install %TITLE% %JVM% %JCP% %JLP% -start %CMAIN% -stop %CSHUT%
-method %MSHUT% -out %LOUT% -err %LERR% -current %APPROOT%


pause
```


INTEGRATION
TECHNOLOGIES
TARSYS