



Hermes Message Service Handler Installation Guide

Version 0.9.3.1

Maintained by

**Patrick Yee (kcyee@cecid.hku.hk)
Center for E-commerce Infrastructure Development
The University of Hong Kong**

Copyright © Center for E-commerce Infrastructure Development 2002. All Rights Reserved.

Table of Content

Table of Content	2
1. Introduction	3
1.1. Document Purpose.....	3
1.2. Project Background	3
2. Pre-requisites.....	4
2.1. Java SDK	4
2.2. Ant.....	4
2.3. Database	4
2.4. Application Server	6
2.5. Mail Server	6
3. Getting Source Code	7
3.1. Getting release files.....	7
3.2. Getting most updated files.....	7
4. Building	9
5. Deployment to application server.....	10
5.1. Deployment based on source code	10
5.2. Deployment based on distribution package.....	10
5.3. Configuration file	10
5.4. Generate password file for authentication	11
5.5. Other database JDBC drivers.....	11
5.4. Patches	12
6. Administrating Hermes.....	14
6.1. Halt / Resume.....	14
6.2. Backup / Restore	14
6.3. Archive	14
6.4. Diagnosis Dump	15
7. Client applications setup	16
7.1. Files and libraries	16
7.2. Configuration file	16
7.3. Patches	17
8. Test drive	18
Appendix A. Fields in Properties Files	19
A.1. msh.properties.xml.....	19
A.2. msh_client.properties.xml	23
A.3. monitor.properties.xml	24
Appendix B. Fields in diagnosis.properties.xml.....	25
Appendix C. Error Codes	26

1. Introduction

1.1. Document Purpose

The aim of this document is to present the guidelines for installation of Hermes - our implementation of Message Service Handler (MSH). It provides step-by-step instructions to guide the users to download, to build and to deploy our MSH.

1.2. Project Background

Hermes is developed under the Project Phoenix, by the Center for E-Commerce Infrastructure Development, the University of Hong Kong.

Project Phoenix is officially titled "Establishment of ebXML Software Infrastructure in Hong Kong". As its name says, the project aims to establish an e-commerce infrastructure in Hong Kong using the ebXML standard.

The project is primarily funded by the Hong Kong Government's Innovation and Technology Fund, and is also supported by some sponsorship from several industrial partners.

The project commenced in 1st January 2002 and shall be completed by the end of year 2003. The project duration is two years.

Hermes is released as an open-source project under Academic Freeware License. The source code repository is being hosted at SourceForge.

2. Pre-requisites

2.1. Java SDK

Our MSH is developed using Java. It needs Java 2 SDK, Standard Edition. We developed the MSH using J2SDK version 1.4.x and 1.3.1. You can download J2SDK from <http://java.sun.com>.

The only difference for our MSH to use the version 1.3.1 and version 1.4 is that we will conduct certificate path verification of digital signature only when J2SDK version 1.4.0 or above is used. We have made use of an API from J2SDK version 1.4.x for doing the certificate path verification. If J2SDK version 1.3.1 is used to run our MSH, the verification step is automatically skipped. Therefore, we recommend version 1.4 to be used.

Although not mandatory, you are recommended to define an environment variable `$JAVA_HOME` to point to the installation directory of the J2SDK installed. This ensures smooth installation and running of most Java based tools. If you want to add the `$JAVA_HOME/bin` directory to the environment variable `$PATH`, please place the directory in the beginning of the `$PATH` variable. From our experience in Win32 platform, if we do not place `$JAVA_HOME/bin` in the beginning of `$PATH` variable, some bugs in XALAN library will be hit.

Also, if J2SDK version 1.4.x is used, the built-in XALAN library should be patched. For details of the patch, please see the instruction in Section 5.5.

2.2. Ant

If you want to rebuild MSH from its source code, you will need Jakarta Ant. You can download and find more information about Jakarta Ant at <http://jakarta.apache.org/ant/>.

We are using Jakarta Ant version 1.5 for development. For your convenience, please make sure that `$ANT_HOME/bin` directory is added to the environment variable `$PATH`, so that Ant can be invoked in other directories.

2.3. Database

The MSH needs a database to store the status of communication, to support resilience, reliable messaging, message tracking, etc.

We followed SQL92 standard when designing the MSH database tables. So, We basically support all RDBMS, as long as their JDBC drivers are available. We used PostgreSQL for our development.

In our distribution, we bundled three JDBC drivers: PostgreSQL and MySQL and HSQLDB. PostgreSQL is a full feature database server. MySQL is a fast and powerful database server. HSQLDB is a file-based, single user database.

2.3.1. PostgreSQL, MySQL or other database server

If PostgreSQL, MySQL or other database server is used, it should be properly setup first. Please refer to the manual of the corresponding database server for details of installing and configuring. For convenience, please turn on network mode (in particular, TCP/IP) for accessing the database remotely, even if your database server and MSH are going to be deployed in the same computer.

After installation, a database for MSH should be created manually. We recommend naming the database as “`msh`”. Also, a user with password should be created manually. The database user should be granted permissions to create and drop tables, read and write tables in database “`msh`”, as the MSH will create the necessary tables automatically upon the first time start up.

Since the database schema in this version may not be compatible with the older version, so we recommend dropping all tables in the old database manually before installing the current version. If you need to retain old data, please use update SQL in our source code distribution to patch the old database.

2.3.2. HSQLDB

If HSQLDB is used, there is no installation work needed. Since it is a file-based database, data is stored into one or more files in the file system.

The JDBC URL of HSQLDB is as follows:

```
jdbc:hsqldb:/path/databasename
```

In this example, the data will be stored into files in the directory “`path`”. Please note that the directory “`path`” be created first manually. Those files will be prefixed by “`databasename`”, and they will be created on-the-fly automatically. For JDBC access to HSQLDB, the user “`sa`” with empty password will be used.

Since the database schema in this version is not compatible with the older version, so we recommend deleting the old database files manually before installing the current version.

2.4. Application Server

The core part of our MSH is a servlet. Therefore, it runs on a servlet container.

We developed and tested the MSH under Jakarta Tomcat 4.0.x, which is a reference implementation of Java Servlet specification. Therefore, we believe that our MSH can be run on other application servers that support the Java Servlet specification.

We have also tested to deploy our MSH successfully on BEA WebLogic Server, JBoss, Macromedia JRun, Sun ONE Web Server. More application servers will be tested in the future.

The binary code of our MSH will be packaged into a web archive file (or WAR file). This is a standard way to deploy servlets to application servers. The installation and deployment instructions for different application servers differ. So, we are not going to cover this in details here. However, sometimes, some patches should be applied to avoid bugs in the application servers. Please refer to Chapter 5 for the tricks we have tested.

Note that we recommend to deploy the MSH on the application with `/msh/` as the context path. In other words, we target to access the MSH application server with the following URL:

```
http://<host.name>:<port>/msh/
```

The default port to use for Tomcat is `8080`. You may want to change the port number as well as the context path, but please note that you may have to change the source code of the sample applications in order to run the test. Please refer to Chapter 7 for testing issues.

2.5. Mail Server

Our MSH supports to use SMTP as the underlying transport protocol optionally. In our implementation, we rely on a standard SMTP based mail server to handle the SMTP protocol details.

In other words, whenever our MSH needs to send out messages using SMTP protocol, it will make use of a SMTP based mail server as the outgoing mail server. We supports SMTP authentication optionally.

And the MSH will assume its incoming messages via SMTP protocol to be handled and stored by the SMTP based mail server. Periodically, the MSH will query the mail server using POP3 or IMAP protocol to get those incoming messages for further processing.

Therefore, the choice of mail servers is arbitrary, as long as it supports SMTP, as well as POP3 or IMAP. We used Qmail for our development.

3. Getting Source Code

If you want to build the deployment file from source code, you can follow the instructions in this and the next chapters. Or, if you want to deploy the pre-compiled binary file from the binary distribution, you can jump directly to the Chapter 5.

We use SourceForge.net's CVS repository to control the versions of our source code. Currently the project only contains one module: ebxmlms. The CVS repository can be checked out through anonymous (pserver) CVS. To access the CVS server, you may make use of command line program cvs or a GUI program likes WinCVS.

3.1. Getting release files

You can download release files of the ebxmlms project from the SourceForge web site or freebXML.org. Just go to the following URLs and download the most updated release file packages.

<http://sourceforge.net/projects/ebxmlms/>
<http://www.freebxml.org/download.htm>

3.2. Getting most updated files

3.2.1. Command Line

You can follow the instructions in SourceForge here:

http://sourceforge.net/cvs/?group_id=56612

Briefly, to login to the CVS server, please type (in one line with no spaces between):

```
cvs -d:pserver:anonymous@cvs.ebxmlms.sourceforge.net  
:/cvsroot/ebxmlms login
```

When prompted for a password for *anonymous*, simply press the Enter key.

To check out the code from the CVS server, please type (in one line with no spaces between):

```
cvs -z3 -d:pserver:anonymous@cvs.ebxmlms.sourceforge.net  
:/cvsroot/ebxmlms co ebxmlms
```

3.2.2. WinCVS

You should configure WinCVS with the following CVSROOT:

```
anonymous@cvs.ebxmlms.sourceforge.net:/cvsroot/ebxmlms
```

Please also choose Authentication as ‘“passwd” file on the server’.

Then, login the server by choosing “Admin / Login...”. When prompted for a password for *anonymous*, simply press OK.

To check out the code from the CVS server, choose “Checkout module...” and enter “ebxmlms” as the module name.

4. Building

Building of the MSH project requires Jakarta Ant. Using Ant is extremely easy. All you need to do is to change into the module directory you checked out in Chapter 3, and type:

```
ant <target>
```

Different targets can be used to execute different actions. The meaning of the targets is listed below.

Target	Meaning
<code>compile</code>	To compile the source.
<code>build</code>	To package a JAR file from the compiled classes.
<code>dist</code>	To make a ZIP file for binary distribution. The ZIP file is just the same as the binary distribution that we are releasing. It is ready for distribution and installation. It contains a WAR file, which is for deployment of MSH to the application server.
<code>doc</code>	To make JavaDoc from the source.
<code>clean</code>	To clean up all intermediate and resultant files generated by the above targets.

5. Deployment to application server

5.1. Deployment based on source code

You should compile the source code by following the instructions in the last chapter. In particular you need to build a distribution package by running the following target:

```
ant dist
```

You should be able to create a ZIP package in the `dist` subdirectory. The package structure will be exactly the same as the pre-compiled (binary) package released by us. You can then follow the instructions in the next section to deploy the binary files.

5.2. Deployment based on distribution package

Firstly you should extract the distribution package. Within the package, you can find a web application archive file:

```
msh.war
```

This is the file packaged in standard format according to the servlet specification. And this file is ready to be deployed to the application server. You should follow the manual of the chosen application server for the instructions to deploy a new servlet application. For Tomcat, you can do so by just copying the WAR file to

```
<tomcat_home>/webapps
```

and then restart Tomcat.

5.3. Configuration file

You need to edit and install the configuration file in order to make MSH server to work properly. The file is named as

```
msh.properties.xml
```

The file is XML based and human readable. The fields and their meanings in the configuration file can be found in Appendix A. Please set the values carefully, one by one. Our experience shows that over 90% of the installation problems are due to mis-configuration in the property file.

Our MSH will search for the configuration file in four locations in a searching sequence as follows:

1. First, MSH will look for a system property that supposed to point to a directory, called `prop.home`. If the variable is defined, MSH will look for the property file in that directory.
2. Secondly, MSH will look for the property file in `$CLASSPATH` of the application server. You can place the configuration file in a specific directory and then add that directory to the `$CLASSPATH` in the configuration of the application server.
3. Thirdly, MSH will search for the configuration file in the current directory. For application server, usually it is the directory where you issue the start up command of the application server, or the installation directory of the application server.
4. Finally, if the configuration file cannot be found in the current directory, our MSH will search for the configuration file in the `$HOME` directory of the current user.

If you deploy from compiled source, the file can be found in `conf` sub-directory of the module directory. If you deploy from distribution package, the file can be found in `conf` sub-directory of the extracted package. You should copy the file to one of the four locations mentioned above, and edit the file according to your preferred configuration.

5.4. Generate password file for authentication

Optionally, a password file can be specified at MSH application server to facilitate authentication. To create and maintain the password file, type:

```
java -cp /path/to/msh.jar  
      hk.hku.cecid.phoenix.common.util.PassTool add
```

You can input the location of the password file, user name and password to add a record in the password file. A password file can contain multiple records. Simply run the above command multiple times to append to the file.

This file generated will be specified in the property named `MSH/Config/AuthenticationFile` in the property file of the MSH application server. And in the client side, the user name and password will be specified in the property file of the MSH client. You can suppress the authentication feature by removing the `MSH/Config/AuthenticationFile` property in the property file of the MSH application server.

5.5. Other database JDBC drivers

We bundled three JDBC drivers: PostgreSQL, MySQL and HSQLDB. If you plan to use other database server other than those three brands, you need to manually add the JDBC driver to the application server.

Generally, you should add the JAR file of the JDBC driver to the `$CLASSPATH` of the application server. And you need to specify the class name and the JDBC URL of the driver in the configuration file of the MSH application server. Our MSH will dynamically load the driver in run time.

For Tomcat, you can add the JAR file of the JDBC driver to `$CATALINA_HOME/common/lib`.

5.4. Patches

5.4.1. J2SDK

J2SDK 1.4.x bundles with an XML library XALAN. However, the version of XALAN bundled is not up-to-dated. And it doesn't go well with the XML security library we are using. So if J2SDK version 1.4.x is used, you should patch the J2SDK with the correct version of XALAN. To do so, you should make a new directory:

```
$JAVA_HOME/jre/lib/endorsed
```

Then, copy the file `xalan.jar` to `$JAVA_HOME/jre/lib/endorsed`.

You can find the file from the `lib` sub-directory of the source code directory tree, or you can get the file by extracting `msh.war` by the command:

```
jar xf msh.war
```

5.4.2. Tomcat version 4.0.3

Due to some unknown problem, JAXM library does not go well with Tomcat if the JAR files of the library are placed in the individual deployment directory of the web application module. So, if Tomcat version 4.0.3 is used, you should copy both the files:

```
jaxm-api.jar  
saaj-api.jar
```

from the `lib` sub-directory of the source code directory tree, to

```
$CATALINA_HOME/common/lib
```

You can find the files from the `lib` sub-directory of the source code directory tree, or you can get the files by extracting `msh.war` by the command:

```
jar xf msh.war
```

5.4.3. Tomcat version 4.0.4, 4.0.5 or 4.0.6

Due to some unknown problem, XALAN library does not go well with Tomcat if the JAR file of the library are placed in the individual deployment directory of the web application module. So, if Tomcat version 4.0.4 or 4.0.5 is used, you should copy the file:

```
xalan.jar
```

from the `lib` sub-directory of the module directory, to

```
$CATALINA_HOME/common/lib.
```

You can find the file from the `lib` sub-directory of the source code directory tree, or you can get the file by extracting `msh.war` by the command:

```
jar xf msh.war
```

6. Administrating Hermes

We have developed several tools for managing and administrating Hermes.

6.1. Halt / Resume

We support halting and resuming the service of Hermes without starting/stopping the application server. This might be useful when you want to pause the operation of the MSH while do not want to affect other applications deployed on the same application server.

The Halt / Resume function is provided through an API. The call is made on the MSH Stub object (i.e. Request object). Please refer to the development guide for the architecture of Hermes and refer to the JavaDoc of the source code of Hermes for the API.

We might wrap up this function as a standalone tool in later releases.

6.2. Backup / Restore

We support backup and restore of the database data and file repository of Hermes. This is useful for protecting your data.

The Backup / Restore function is provided through an API. The call is made on the MSH Stub object (i.e. Request object). Please refer to the development guide for the architecture of Hermes and refer to the JavaDoc of the source code of Hermes for the API. The backup function will save the backup of the MSH system to a file specified in the MSH configuration file. By the same token the restore function will restore the backup image located in the file specified in the MSH configuration file. Please refer to Appendix A for the configuration.

We might wrap up this function as a standalone tool in later releases.

6.3. Archive

We support archiving of the database data and file repository of Hermes. This is useful for saving your data for offline analysis and for cutting down the size of the repository.

The Archive function is provided through an API. The call is made on the MSH Stub object (i.e. Request object). Please refer to the development guide for the architecture of Hermes and refer to the JavaDoc of the source code of Hermes for the API. The archive function will save the data of the MSH

system to the directory specified in the MSH configuration file. Please refer to Appendix A for the configuration.

We might wrap up this function as a standalone tool in later releases.

6.4. Diagnosis Dump

We support a function for the administrator to dump all the states of the MSH out. This will be useful when something wrong happened, and the dump may help to trace the problem.

The diagnosis dump function is provided as a command line utility. The script to invoke the utility can be found at the diagnosis directory in the binary distribution package. You have to modify the `diagnosis.properties.xml` file to configure the behavior of the dump. The meaning of individual property inside the configuration file can be found in Appendix B.

To invoke the utility, please run inside the diagnosis directory:

```
./run.sh <path/to/diagnosis.properties.xml>
```

or

```
run.bat <path/to/diagnosis.properties.xml>
```

Alternatively you can run the script without passing the path to the properties file. The utility will try to locate the property file in the current directory.

7. Client applications setup

Our MSH architecture allows client applications to be installed on a separate machine besides the machine running the application server. For more details about our architecture, please refer to our Development Guide.

This implies, in order for the client applications to connect to the MSH application server, some more setup should be done on the machine running the client applications. If you plan to install both your client application and the MSH application server on the same machine, you can skip some steps in this chapter that already done in previous chapter.

7.1. Files and libraries

Some files should be installed to the client application machine. We assume you should start with the distribution package. If you are compiling from the source, you can run

```
ant dist
```

to create the distribution package. Then, you should follow the below instructions to install necessary files on the client application machine.

1. Unzip the file `msh.zip` to a temporary directory.
2. On the client application machine, create a directory, we recommend to name it as `msh_client`.
3. Copy `lib/*.jar` to `msh_client/lib`. These libraries are used to link to the client applications.
4. Copy `sample/*` to `msh_client/sample`. These are sample client applications.

7.2. Configuration file

The MSH stub will refer to a configuration file for parameters. You need to edit and install the configuration file in order to make client application to work properly. The file is named as

```
msh_client.properties.xml
```

The file is XML based and human readable. The fields and their meanings in the configuration file can be found in Appendix A. The MSH stub will search for the configuration file in four locations in a searching sequence as follows:

1. First, MSH stub will look for a system property that supposed to point to a directory, called `prop.home`. If the variable is defined, MSH stub will look for the property file in that directory.
2. Secondly, MSH stub will look for the property file in `$CLASSPATH` of the current Java process. You can place the configuration file in a specific directory and then add that directory to the `$CLASSPATH` environment variable before starting the client application.
3. Thirdly, MSH will search for the configuration file in the current directory.
4. Finally, if the configuration file cannot be found in the current directory, our MSH will search for the configuration file in the `$HOME` directory of the current user.

The file can be found in the `conf` subdirectory of the temporary directory that you created in the previous section. You should just copy it to the appropriate directory as described above and edit the file according to your preferred configuration.

7.3. Patches

7.3.1. J2SDK

J2SDK 1.4.x bundles with an XML library XALAN. However, the version of XALAN bundled is not up-to-dated. And it doesn't go well with the XML security library we are using. So if J2SDK version 1.4.x is used, you should patch the J2SDK with the correct version of XALAN. To do so, you should make a new directory:

```
$JAVA_HOME/jre/lib/endorsed
```

Then, copy the file `xalan.jar` to `$JAVA_HOME/jre/lib/endorsed`.

You can find the file from the `lib` sub-directory of the source code directory tree, or you can get the file by extracting `msh.war` by the command:

```
jar xf msh.war
```

8. Test drive

You can test your installation by executing the following steps:

1. Start your application server that hosting the MSH
2. On the client machine, go to `msh_client/sample`. Run `RunLoopBack.bat` (for Windows) or `RunLoopBack.sh` (for Unix) to run a loop back test. If you have modified the port number or context path of your application server, you will have to modify line 27 of `LoopBack.java` to point to the correct URL.
3. On the client machine, go to `msh_client/sample`. Run `RunMonitor.bat` (for Windows) or `RunMonitor.sh` (for Unix) to start a GUI for sending messages. You should modify the "To MSH URL" to point to the target peer for sending/receiving. The default setting runs a loop back test. Similar to step 2, you may have to modify the port number or context path in "To MSH URL". Optionally, the default setting can be overwritten by the settings set by a properties file called `monitor.properties.xml`. To use the file, you need to copy the file from the `conf` subdirectory to `msh_client/sample` and modify the file to suit your environment.

Note: You should have `$JAVA_HOME/bin` added to the `$PATH` environment variable in order to run the tests successfully.

Appendix A. Fields in Properties Files

A.1. msh.properties.xml

This properties file is used by the MSH server:

Field	Description
MSH/Log/UseLogger	The logging parameter for the servlet. Choose whether Log4J or JDK logging package is used for logging. Valid values are "LOG4J" and "JDK".
MSH/Log/LogPath	The logging parameter for the servlet. The directory for storing the log file.
MSH/Log/LogFile	The logging parameter for the servlet. The log file name.
MSH/Log/LogLevel	The logging parameter for the servlet. The log level of the logger. Valid values range from 0 to 4. "0" means most log, and "4" means no log.
MSH/Log/MaxFileSize	The maximum log file size in bytes. If the maximum size is reached, a new log file will be created and the logger will "roll over" to use the new log file. If this property is missing, or the value is smaller than 0, the roll over is disabled.
MSH/Config/URL	The URL for the external systems to locate the MSH application server. You should fill in the host name, port number and context path of the MSH application server.
MSH/Config /AuthenticationFile	A password file storing the user name and password for authentication. If this property is missing, no authentication will be done on MSH application server.
MSH/Config /PositiveAcknowledgment	Optional property controlling whether a positive acknowledgment message will be generated if MSH successfully sends a message.
MSH/Config /AugmentedErrorMessage	Optional property controlling whether an error message being sent back to the sender application is augmented with the original message as a MIME payload attachment.
MSH/Config /ContentTransferEncoding	Optional property controlling the content transfer encoding of the payloads while sending out messages in HTTP
MSH/Proxy/Host	The host name of HTTP proxy server for sending out messages to another MSH through HTTP. If you can connect to Internet

	without using proxy server, this field should be commented.
MSH/Proxy/Port	The port number of HTTP proxy server for sending out messages to another MSH through HTTP. If you can connect to Internet without using proxy server, this field should be commented.
MSH/Mail/SMTP/Host	The host name of the SMTP server for sending out messages to another MSH through SMTP.
MSH/Mail/SMTP/User	The user name to connect to the SMTP server. Use only for SMTP authentication. If missing, ordinary SMTP without authentication is assumed.
MSH/Mail/SMTP/Password	The password to connect to the SMTP server. Use only for SMTP authentication. If missing, ordinary SMTP without authentication is assumed.
MSH/Mail/Debug	Controls whether debug message will be output for mail protocol handling or not.
MSH/Mail/Poll/Protocol	The protocol used by MSH server to connect to the mail server for incoming messages. If you do not need to support SMTP protocol for incoming messages, this field should be commented.
MSH/Mail/Poll/Host	The host name used by MSH server to connect to the mail server for incoming messages. If you do not need to support SMTP protocol for incoming messages, this field should be commented.
MSH/Mail/Poll/Port	The port number used by MSH server to connect to the mail server for incoming messages. If you do not need to support SMTP protocol for incoming messages, this field should be commented.
MSH/Mail/Poll/Folder	The folder name on the mail server for retrieving messages, used by MSH server to connect to the mail server for incoming messages. If you do not need to support SMTP protocol for incoming messages, this field should be commented.
MSH/Mail/Poll/User	The user name used by MSH server to connect to the mail server for incoming messages. If you do not need to support SMTP protocol for incoming messages, this field should be commented.
MSH/Mail/Poll/Password	The password used by MSH server to connect to the mail server for incoming messages. If you do not need to support SMTP protocol for incoming messages, this

	field should be commented.
MSH/Mail/Poll /MonitorInterval	The time interval for the MSH server to connect to the mail server for incoming messages. If you do not need to support SMTP protocol for incoming messages, this field should be commented.
MSH/Mail/Poll /ForceChangeSubType	A boolean value to instruct MSH whether to rebuild the content type of the incoming MIME message. This is useful when some mail servers mistakenly change the MIME message content type upon receiving the message.
MSH/Mail/SMIME /Encryption/KeyStore /Path	RESERVED. For future S/MIME development.
MSH/Mail/SMIME /Encryption/KeyStore /File	RESERVED. For future S/MIME development.
MSH/Mail/SMIME /Encryption/KeyStore /Password	RESERVED. For future S/MIME development.
MSH/Mail/SMIME /Decryption/KeyStore /Path	RESERVED. For future S/MIME development.
MSH/Mail/SMIME /Decryption/KeyStore /File	RESERVED. For future S/MIME development.
MSH/Mail/SMIME /Decryption/KeyStore /Alias	RESERVED. For future S/MIME development.
MSH/Mail/SMIME /Decryption/KeyStore /Password	RESERVED. For future S/MIME development.
MSH/DigitalSignature /TrustedAnchor /KeyStore/Path	The directory holding the keystore that holding trusted certificates. This is used for referencing a keystore for trust anchor and certificate path verification.
MSH/DigitalSignature /TrustedAnchor /KeyStore/File	The file name of the keystore that holding trusted certificates. This is used for referencing a keystore for trust anchor and certificate path verification.
MSH/DigitalSignature /TrustedAnchor /KeyStore/Password	The password of the keystore that holding trusted certificates. This is used for referencing a keystore for trust anchor and certificate path verification.
MSH/DigitalSignature /AckSign/KeyStore /Path	The directory holding the keystore for signing acknowledgments.
MSH/DigitalSignature /AckSign/KeyStore /File	The file name of the keystore for signing acknowledgments.
MSH/DigitalSignature	The signing algorithm used for signing

/AckSign/KeyStore /Algorithm	acknowledgments. If not specified, “dsa-sha1” is assumed. Currently only the values “dsa-sha1” or “rsa-sha1” are valid.
MSH/DigitalSignature /AckSign/KeyStore /Alias	The alias of the key used for signing acknowledgments.
MSH/DigitalSignature /AckSign/KeyStore /Password	The password of the keystore for signing acknowledgments.
MSH/Persistent /Database/Driver	The JDBC driver class name used for connecting to the database.
MSH/Persistent /Database/User	The user to connect to the database.
MSH/Persistent /Database/Password	The password to connect to the database.
MSH/Persistent /Database/URL	The JDBC URL to connect to the database.
MSH/Persistent /Database /TransactionIsolationLevel	The transaction isolation level of database. Valid settings are “READ_COMMITTED”, “READ_UNCOMMITTED”, “REPEATABLE_READ” and “SERIALIZABLE”.
MSH/Persistent /Database /InitialConnections	The initial pool size of the database pool.
MSH/Persistent /Database /MaxConnections	The maximum pool size of the database pool.
MSH/Persistent /Database /MaximumWait	The maximum waiting time for MSH to wait for an available database connection.
MSH/Persistent /Database /MaximumIdle	The maximum time that a database connection can be idled. The MSH would reconnect the database connection that has been idled more than the specified period.
MSH/Persistent /MessageRepository	The repository directory for storing ebXML messages persistently for tracking and resilience.
MSH/Persistent /Database/MaxWait	The maximum period of time in milliseconds for MSH to wait for an available database connection from the connection pool.
MSH/Persistent /MaxFiles	The maximum number of files to be stored in a single sub-directory in the repository directory.
MSH/Persistent /BackupFile	The file name to place the backup file when a MSH backup command is issued.
MSH/Persistent /ArchiveDirectory	The directory name where the archived data are placed when a MSH archive command is issued.
MSH/MessageListener /TrustedRepository	A semi-colon delimited string containing the path that the MSH application server can

	save the received messages.
MSH/MessageListener /ObjectStore	The repository directory for storing the state of the MSH server. The MSH server can restore its last state by loading these objects.

A.2. msh_client.properties.xml

This properties file is used by the MSH stub:

Field	Description
Request/Log/UseLogger	The logging parameter for the MSH Stub. Choose whether Log4J or JDK logging package is used for logging. Valid values are "LOG4J" and "JDK".
Request/Log/LogPath	The logging parameter for the MSH Stub. The directory for storing the log file.
Request/Log/LogFile	The logging parameter for the MSH Stub. The log file name.
Request/Log/LogLevel	The logging parameter for the MSH Stub. The log level of the logger. Valid values range from 0 to 4. "0" means most log, and "4" means no log.
Request/Log /MaxFileSize	The maximum log file size in bytes. If the maximum size is reached, a new log file will be created and the logger will "roll over" to use the new log file. If this property is missing, or the value is smaller than 0, the roll over is disabled.
Request/Config/URL	The URL for the client applications to locate MSH application server. You should fill in the host name, port number and context path of the MSH application server.
Request/Config /MonitorInterval	The time interval for the client applications to connect to MSH server at Tomcat for downloading new messages.
Request/Config /UserName	The user name to attach to every command message sending to the MSH server for authentication.
Request/Config /Password	The password to attach to every command message sending to the MSH server for authentication.
Request/Config /MaxNumPayload	Optional parameter controlling the maximum number of payload allowed. If not specified, no bound on maximum is assumed.
Request/Config /MaxPayloadSize	Optional parameter controlling the maximum size of payload allowed in bytes. If not specified, no bound on maximum is assumed.

Request/Proxy/Host	The host name of HTTP proxy server for sending out commands/messages to the MSH server. If you can connect to the MSH server without using proxy server, this field should be commented.
Request/Proxy/Port	The port number of HTTP proxy server for sending out commands/messages to the MSH server. If you can connect to the MSH server without using proxy server, this field should be commented.
Request /MessageListener /MessageRepository	The temporary directory at client application side for storing messages. The MSH can be configured to forward all received messages directly to the machine running client applications, this is the directory to store the forwarded messages.

A.3. monitor.properties.xml

This properties file is used by the Monitor application:

Field	Description
Request/Monitor /DefaultConfig /ToMSHURL	The default configuration parameters for the sample application: MSH Monitor. This represents the To MSH URL field.
Request/Monitor /DefaultConfig/CPAID	The default configuration parameters for the sample application: MSH Monitor. This represents the CPA ID field.
Request/Monitor /DefaultConfig /ConversationID	The default configuration parameters for the sample application: MSH Monitor. This represents the Conversation ID field.
Request/Monitor /DefaultConfig /Service	The default configuration parameters for the sample application: MSH Monitor. This represents the Service field.
Request/Monitor /DefaultConfig /Action	The default configuration parameters for the sample application: MSH Monitor. This represents the Action field.

Appendix B. Fields in diagnosis.properties.xml

Field	Description
Diagnosis/Destination	The destination directory where the dump report will be written to.
Diagnosis/MshConfigFile	The full path of the configuration file of the MSH server to be dumped.
Diagnosis/Copy/ToDir	There can be multiple Copy elements. For each Copy element, there should be only one ToDir element, which points to the destination directory of the copy action.
Diagnosis/Copy/Path	The files to be copied to the directory specified by the ToDir element in the same Copy element. Wildcard is supported. There can be multiple Path elements in the same Copy element.
Diagnosis/Criteria /TimePeriod /StartTime	Optionally, time period can be used to specify partial data to be dumped. This is the starting time of the time period specified.
Diagnosis/Criteria /TimePeriod/EndTime	Optionally, time period can be used to specify partial data to be dumped. This is the ending time of the time period specified.
Diagnosis/Criteria /AppContext/CpaId	Optionally, the dump can be made upon specific application context. This is the CPA ID of the application context specified.
Diagnosis/Criteria /AppContext /ConversationId	Optionally, the dump can be made upon specific application context. This is the Conversation ID of the application context specified.
Diagnosis/Criteria /AppContext/Service	Optionally, the dump can be made upon specific application context. This is the Service of the application context specified.
Diagnosis/Criteria /AppContext/Action	Optionally, the dump can be made upon specific application context. This is the Action of the application context specified.

Appendix C. Error Codes

Code	Description
10001	Initialization error
10002	Unknown error
10003	Property not properly set
10004	Data error
10005	State error
10006	File IO error
10007	File not found
10008	HTTP POST request failed
10009	Servlet IO error
10010	Authentication failed
10011	Registration failed
10012	Unknown application context
10013	Error reading/writing ZIP stream
10101	Invalid SMTP server
10102	Cannot compose mail message
10103	Cannot send mail message
10104	Invalid POP/IMAP folder
10105	Invalid POP/IMAP server
10201	Invalid keytore
10202	Cannot encrypt message
10203	Cannot decrypt message
10204	Cannot sign message
10205	Verification of signature failed
10301	Cannot close DB connection
10302	Invalid parameter
10303	Cannot get DB connection
10304	DB connection are freed more than allocated
10305	Cannot load JDBC driver
10306	Cannot create DB connection
10307	Existing DB schema incorrect
10308	Data inconsistency
10309	Cannot create DB table
10310	Cannot query record from DB
10311	Cannot write record to DB
10312	Database backup error
10313	Database restore error
10314	Cannot commit DB changes
10315	Cannot rollback DB changes
10316	Cannot find message in DB
10401	Cannot unlock a non-existent object
10402	Cannot rollback
10501	Default MessageFactory cannot be instantiated
10502	Cannot serialize SOAP message

10503	Cannot internalize SOAP object
10504	Cannot save change to SOAP message
10505	Cannot send SOAP message
10506	Cannot initialize SOAP connection
10507	Invalid HTTP SOAP connection
10508	Cannot create SOAP object
10509	Cannot process SOAP message