EXECUTIVE SUMMARY

Many organizations work together to protect and advance public health. These organizations need to use the internet to securely exchange sensitive data between various different public health information systems. The exchange of data, also known as "messaging," is enabled through electronic messages created by using special file formats and a standard vocabulary. The exchange uses a common approach to security and encryption, methods for dealing with various firewalls, and internet protection schemes. The system provides a standard way for addressing and routing content and a standard and consistent way for information systems to confirm an exchange.

The Centers for Disease Control and Prevention (CDC) Public Health Information Network Messaging System (PHINMS) is the software that facilitates this messaging. The system securely sends sensitive data to and receives sensitive data from public health information systems through the internet.

This technical reference guide provides advanced instructions for configuring the PHINMS 3.0 application.
## REVISION HISTORY

<table>
<thead>
<tr>
<th>VERSION #</th>
<th>IMPLEMENTER</th>
<th>DATE</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Joseph Mai</td>
<td>06-13-2017</td>
<td>Create PHINMS 3.0 Technical Guide</td>
</tr>
<tr>
<td>1.0</td>
<td>Charlie Peng</td>
<td>06-19-2017</td>
<td>Minor changes and proofreading</td>
</tr>
<tr>
<td>1.0</td>
<td>Regina Collins</td>
<td>09-15-2017</td>
<td>Proofreading</td>
</tr>
</tbody>
</table>
## TABLE OF CONTENTS

1.0 Introduction .................................................................................................................. 8
   1.1 Communiqués .............................................................................................................. 8

2.0 Advanced Database Information .................................................................................. 9
   2.1 TransportQ Database Fields .................................................................................... 9
   2.2 WorkerQ Database Fields ....................................................................................... 10
   2.3 Create MSSQL Database ......................................................................................... 11
   2.4 Create MSSQL Tables ............................................................................................ 11
   2.5 Transport Codes ....................................................................................................... 11

3.0 File System-Based Transport Queues ....................................................................... 12
   3.1 XML File Descriptor ............................................................................................... 12
   3.2 XML File Descriptor Response ............................................................................... 12
   3.3 File-Based TransportQ ............................................................................................ 13
   3.4 Name-Value Based File Descriptor ........................................................................... 14
   3.5 Sending File Response ............................................................................................ 13

4.0 JDBC Drivers AND SYNTAX Information .................................................................. 14
   4.1 SQL .......................................................................................................................... 14
      4.1.1 SQL 2000 .......................................................................................................... 14
      4.1.2 SQL 2005 .......................................................................................................... 14
      4.1.3 SQL 2008 .......................................................................................................... 14
   4.2 HSQL ........................................................................................................................ 14
   4.3 MS Access ................................................................................................................ 16
   4.4 MySQL ..................................................................................................................... 15
   4.5 Oracle ....................................................................................................................... 15

5.0 Advanced Console Information .................................................................................. 16
   5.1 Transport Status and Error Codes ............................................................................ 16

6.0 Table Scripts ................................................................................................................ 16
   6.1 MSSQL Scripts ........................................................................................................ 16
      6.1.1 TransportQ Table - Sender .............................................................................. 16
      6.1.2 RnRworkerQ Table - Sender .......................................................................... 17
      6.1.3 ErrorQ Table - Sender ..................................................................................... 18
      6.1.4 Messaging Cache Table - Sender ..................................................................... 18
      6.1.5 Messaging Queue Table - Receiver ................................................................. 18
      6.1.6 TransportQ Table - Receiver .......................................................................... 19
      6.1.7 ErrorQ Table - Receiver .................................................................................. 19
   6.2 Oracle Scripts .......................................................................................................... 20
      6.2.1 TransportQ Table - Sender .............................................................................. 20
      6.2.2 WorkerQ Table - Sender .................................................................................. 21
      6.2.3 ErrorQ Table - Sender ..................................................................................... 22
      6.2.4 Messaging Cache Table - Sender ..................................................................... 22
      6.2.5 Messaging Queue Table - Receiver ................................................................. 23
6.2.6 TransportQ Table - Receiver ..............................................................23
6.2.7 ErrorQ Table - Receiver ........................................................................24
6.2.8 Route-not-Read Table - Sender ............................................................25
6.3 MySQL Scripts ..........................................................................................26
  6.3.1 TransportQ Table - Sender .................................................................26
  6.3.2 WorkerQ Table - Sender .....................................................................26
  6.3.3 Error Q Table - Sender .......................................................................27
  6.3.4 Messaging Cache Table - Sender .........................................................27
  6.3.5 Messaging Queue Table - Receiver .....................................................27
  6.3.6 TransportQ - Receiver .......................................................................28
  6.3.7 Message ErrorQ - Receiver .................................................................28
  6.3.8 Route-not-Read Table - Sender ...........................................................29
6.4 HSQL Scripts ............................................................................................30
  6.4.1 TransportQ Table - Sender .................................................................30
  6.4.2 WorkerQ Table - Sender ....................................................................30
  6.4.3 ErrorQ Table - Sender .......................................................................31
  6.4.4 Messaging Cache Table - Sender .........................................................31
  6.4.5 Messaging Queue Table – Receiver ......................................................31
  6.4.6 TransportQ Table - Receiver ...............................................................32
  6.4.7 ErrorQ Table - Receiver ....................................................................32
LIST OF TABLES

Table 1. TransportQ Database Fields .............................................................. 11
Table 2. WorkerQ Database Fields ................................................................. 12
Table 3. Transport Status Codes ................................................................. 12
Table 4. Transport Error Codes ................................................................. 12
Table 5. JDBC Drivers ................................................................................. 15
Table 6. Status Codes ................................................................................. 17
Table 7. Error Codes ..................................................................................... 17
<table>
<thead>
<tr>
<th>ACRONYM LIST</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BA</td>
<td>Basic Authentication</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
</tr>
<tr>
<td>CDC</td>
<td>Centers for Disease Control and Prevention</td>
</tr>
<tr>
<td>CPA</td>
<td>Collaboration Protocol Agreement</td>
</tr>
<tr>
<td>CSE</td>
<td>Communications Security Establishment</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>ebMS</td>
<td>Electronic Business Extensible Markup Language Messaging Service</td>
</tr>
<tr>
<td>ebXML</td>
<td>Electronic Business Extensible Markup Language</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise JavaBeans</td>
</tr>
<tr>
<td>ErrorQ</td>
<td>Error Queue</td>
</tr>
<tr>
<td>FIPS</td>
<td>Federal Information Processing Standard</td>
</tr>
<tr>
<td>HF3</td>
<td>Hot Fix 3 (repackaged installer will work with 32 or 64 bit java)</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol over Secure Socket Layer</td>
</tr>
<tr>
<td>IMAP</td>
<td>Internet Message Access Protocol</td>
</tr>
<tr>
<td>J2EE</td>
<td>Java 2 Platform Enterprise Edition</td>
</tr>
<tr>
<td>J2SE</td>
<td>Java 2 Platform Standard Edition</td>
</tr>
<tr>
<td>JDBC</td>
<td>Java Database Connectivity</td>
</tr>
<tr>
<td>JDK</td>
<td>Java Development Kit</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>ITL</td>
<td>Information Technology Laboratory</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute for Standards and Technology</td>
</tr>
<tr>
<td>NVLAP</td>
<td>National Voluntary Laboratory Accreditation Program</td>
</tr>
<tr>
<td>PHIN</td>
<td>Public Health Information Network</td>
</tr>
<tr>
<td>PHINMS</td>
<td>Public Health Information Network Messaging System</td>
</tr>
<tr>
<td>PKCS</td>
<td>Public-Key Cryptography Standards</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>POP</td>
<td>Post Office Protocol</td>
</tr>
<tr>
<td>PSK</td>
<td>Pre-Shared Key</td>
</tr>
<tr>
<td>OASIS</td>
<td>Organization for the Advancement of Structured Information Standards</td>
</tr>
<tr>
<td>ODBC</td>
<td>Open Database Connectivity</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>SP1</td>
<td>Service Pack 1</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>SSO</td>
<td>Single Sign-On</td>
</tr>
<tr>
<td>STP</td>
<td>Secure Transport Protocol</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transport Control Protocol Internet Protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>TransportQ</td>
<td>Transport Queue</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WorkerQ</td>
<td>Worker Queue</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

The Centers for Disease Control and Prevention (CDC) Public Health Information Network Messaging System (PHINMS) Technical Reference Guide assists users with performing actions manually, outside the graphical user interface configurations.

Note: Navigate to www.cdc.gov/phin/tools/phinms when this manual references the PHINMS website.

1.1 Communication with Users

Send questions, suggestions, and/or comments concerning PHINMS support or documentation to the PHINMS website by using the Contact Us email link or the phone number located on the left of the home page. The PHINMS team responds to users’ communications.
2.0 ADVANCED DATABASE INFORMATION

A database contains tables that store incoming and outgoing messages. The messaging cache is an index of incoming messages. Section 4 explains procedures for creating a database, cache, and tables for the transport queue (TransportQ) and worker queue (WorkerQ).

2.1 Transport Queue (TransportQ) Database Fields

Table 1 identifies each field in the PHINMS Sender’s TransportQ and whether the field’s value is set by the PHINMS Sender or the application placing the message into the TransportQ.

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>DESCRIPTION</th>
<th>SOURCE</th>
<th>OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordId</td>
<td>Unique ID of the record in the table and the table’s primary key.</td>
<td>Auto Generated</td>
<td>Mandatory</td>
</tr>
<tr>
<td>messageId</td>
<td>Application-level message identifier.</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>payloadFile</td>
<td>File name of the payload file of an outgoing message relative to a local directory such as myinputs.txt.</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>payloadContent</td>
<td>Used only when the payloadFile field is not specified. Populates the contents of a file within the table.</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>destinationFilename</td>
<td>The name of the payload file when it is stored on the receiver/handler.</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>routeInfo</td>
<td>Points to the routemap table that points to the message route. Maps to a CPA, a configuration file that maps to the uniform resource locator (URL) of the message receiver.</td>
<td>Application</td>
<td>Mandatory</td>
</tr>
<tr>
<td>service</td>
<td>ebXML service name – Case sensitive.</td>
<td>Application</td>
<td>Mandatory</td>
</tr>
<tr>
<td>action</td>
<td>ebXML action – Case sensitive.</td>
<td>Application</td>
<td>Mandatory</td>
</tr>
<tr>
<td>arguments</td>
<td>Arguments specified by the message sender.</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>messageCreationTime</td>
<td>Time when record was created, in UTC format.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>messageRecipient</td>
<td>Recipient’s ID specified by the sender in the TransportQ_out.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>processingStatus</td>
<td>Initial value of the status of record created queued.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>applicationStatus</td>
<td>Status of the application.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>encryption</td>
<td>The value is Yes if payload is encrypted and No if it is not.</td>
<td>Application</td>
<td>Mandatory</td>
</tr>
<tr>
<td>signature</td>
<td>If Yes, XML signature is applied to the payload.</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>publicKeyLdapAddress</td>
<td>LDAP address of the LDAP directory server.</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>publicKeyLdapBaseDN</td>
<td>LDAP Base Distinguished Name of the public key such as o=,</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>publicKeyLdapDN</td>
<td>LDAP Distinguished Name of the public key such as cn=.</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>certificateURL</td>
<td>URL of a recipient’s public key certificate.</td>
<td>Application</td>
<td>Optional</td>
</tr>
<tr>
<td>transportStatus</td>
<td>Transport-level status.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>transportErrorCode</td>
<td>Error code describing the transport failure.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>applicationErrorCode</td>
<td>The error code returned by the service/action in a synchronous manner.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>FIELD NAME</td>
<td>DESCRIPTION</td>
<td>SOURCE</td>
<td>OPTION</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>applicationResponse</td>
<td>The synchronous response returned by the service/action.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>messageSentTime</td>
<td>Time when the message was sent, in UTC format.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>messageReceivedTime</td>
<td>Time when the message was received, in UTC format.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>responseMessageId</td>
<td>Message ID of the response message in the route-not-read scenario.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>responseArguments</td>
<td>Used in the route-not-read scenario to convey arguments being sent by a message sender to a receiving client.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>responseLocalFile</td>
<td>The response to a poll-type request, which may contain a payload file in the route-not-read scenario.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>responseFilename</td>
<td>The response file name in the route-not-read scenario.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>responseContent</td>
<td>Used when the sender.xml configuration file in the message sender specifies that the response payload should be written into a database field instead of a disk.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>responseMessageOrigin</td>
<td>The partyID of the party originating the message in the route-not-read scenario.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>responseMessageSignature</td>
<td>The partyID of the party signing the message in the route-not-read scenario.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>priority</td>
<td>An integer indicating the request’s priority.</td>
<td>Application</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Table 1. TransportQ Database Fields

2.2 Worker Queue (WorkerQ) Database Fields

Table 2 identifies each field in the PHINMS Receiver’s WorkerQ and whether the field’s value was set by the PHINMS Sender or by the PHINMS Receiver.

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>DESCRIPTION</th>
<th>SOURCE</th>
<th>OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordId</td>
<td>Unique ID of the record in the table and the table’s primary key.</td>
<td>Receiver</td>
<td>Mandatory</td>
</tr>
<tr>
<td>messageId</td>
<td>Application-level message identifier.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>payloadName</td>
<td>File name of the payload, specified by the message sender.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>payloadBinaryContent</td>
<td>Image field written to the receiver servlet.</td>
<td>Sender *</td>
<td>Optional</td>
</tr>
<tr>
<td>payloadTextContent</td>
<td>Text field populated if textPayload=true in the servicemap entry.</td>
<td>Sender *</td>
<td>Optional</td>
</tr>
<tr>
<td>localFilename</td>
<td>File written to disk instead of a database when payloadToDisk =true.</td>
<td>Receiver</td>
<td>Mandatory</td>
</tr>
<tr>
<td>service</td>
<td>ebXML service name.</td>
<td>Sender</td>
<td>Mandatory</td>
</tr>
<tr>
<td>action</td>
<td>ebXML action.</td>
<td>Sender</td>
<td>Mandatory</td>
</tr>
<tr>
<td>arguments</td>
<td>Arguments specified by the message sender.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>fromPartyId</td>
<td>PartyID of the message sender.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
<tr>
<td>messageRecipient</td>
<td>Recipient’s ID specified by the sender in the TransportQ_out.</td>
<td>Sender</td>
<td>Optional</td>
</tr>
</tbody>
</table>
TABLE 2. WorkerQ Database Fields

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>DESCRIPTION</th>
<th>SOURCE</th>
<th>OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>errorCode</td>
<td>Error code.</td>
<td>Receiver</td>
<td>Optional</td>
</tr>
<tr>
<td>errorMessage</td>
<td>Error message.</td>
<td>Receiver</td>
<td>Optional</td>
</tr>
<tr>
<td>processingStatus</td>
<td>Initial value of the status of record created queued.</td>
<td>Receiver</td>
<td>Optional</td>
</tr>
<tr>
<td>applicationStatus</td>
<td>Status of the application.</td>
<td>Receiver</td>
<td>Optional</td>
</tr>
<tr>
<td>encryption</td>
<td>The value is Yes if payload stored in WorkerQ is encrypted and No if it is not.</td>
<td>Receiver</td>
<td>Mandatory</td>
</tr>
<tr>
<td>receivedTime</td>
<td>Time when payload was received, in UTC format.</td>
<td>Receiver</td>
<td>Optional</td>
</tr>
<tr>
<td>lastUpdateTime</td>
<td>Time when record was last updated, in UTC format.</td>
<td>Receiver</td>
<td>Optional</td>
</tr>
<tr>
<td>processId</td>
<td>Identifies the process processing the record.</td>
<td>Receiver</td>
<td>Optional</td>
</tr>
</tbody>
</table>

(*) These two fields are mutually exclusive. The payload coming from the sender is placed into one of the two fields by the receiver, depending on the receiver’s configuration value for the textPayload = true/false field.

2.3 Create MSSQL Database

Section 6.1 contains the scripts for creating MSSQL and/or Oracle databases.

2.4 Create MSSQL Tables

Appendix A contains various scripts for creating tables used with MSSQL and/or Oracle.

2.5 Transport Codes

A transport status code is sent back to the TransportQ when a message is delivered or processed. If an error occurs during the delivery of a message, an error code is sent back to the TransportQ. Table 3 describes the transport status codes, and Table 4 describes the transport error codes.

<table>
<thead>
<tr>
<th>TRANSPORT STATUS CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>Message send or receive operation succeeded.</td>
</tr>
<tr>
<td>Failure</td>
<td>Message send or receive operation failed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRANSPORT ERROR CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityFailure</td>
<td>Error logging into message receiver.</td>
</tr>
<tr>
<td>DeliveryFailure</td>
<td>Failed to deliver message.</td>
</tr>
<tr>
<td>NotSupported</td>
<td>Format of the ebXML message or CPA is unsupported.</td>
</tr>
<tr>
<td>Unknown</td>
<td>Not a standard ebXML error.</td>
</tr>
<tr>
<td>NoSuchService*</td>
<td>Service/action failed to map a service on the message receiver.</td>
</tr>
<tr>
<td>CheckSumFailure*</td>
<td>File checksum verification failure at the message receiver.</td>
</tr>
</tbody>
</table>

Table 3. Transport Status Codes

Table 4. Transport Error Codes

Note: The asterisk (*) symbol indicates a custom error code (code not in ebXML specifications).
3.0 FILE SYSTEM-BASED TRANSPORT QUEUES

File System-Based TransportQ is a folder-based option used to send and receive messages. This option is a substitute for database sending and receiving configurations.

3.1 XML File Descriptor

An example XML File Descriptor is shown below:

```xml
<fileDescriptor>
  <recordId>22</recordId>
  <payloadFile>D:\phinms\shared\outgoing\test.txt</payloadFile>
  <payloadContent>
  </payloadContent>
  <destinationFilename>test.txt</destinationFilename>
  <routeInfo>CDCStaging</routeInfo>
  <service>Router</service>
  <action>send</action>
  <arguments>XXDOHelr</arguments>
  <messageRecipient>XXDOH</messageRecipient>
  <messageCreationTime>time</messageCreationTime>
  <encryption>yes</encryption>
  <signature>yes</signature>
  <publicKeyLdapAddress>directory.verisign.com:389</publicKeyLdapAddress>
  <publicKeyLdapBaseDN>o=Centers for Disease Control and Prevention</publicKeyLdapBaseDN>
  <publicKeyLdapDN>cn=cdc phinms</publicKeyLdapDN>
  <acknowledgementFile>D:\phinms\shared\acknowledgments\ack_send.xml</acknowledgementFile>
</fileDescriptor>
```

3.2 XML File Descriptor Response

An example XML File Descriptor response is shown below:

```xml
<acknowledgement>
  <transportStatus>success</transportStatus>
  <transportError>none</transportError>
  <applicationStatus>retrieveSucceeded</applicationStatus>
  <applicationError>none</applicationError>
  <applicationData>targetTable=payroll</applicationData>
  <responseLocalFile>1018387200432</responseLocalFile>
  <responseFileName>test.txt</responseFileName>
  <responseSignature>unsigned</responseSignature>
  <responseMessageOrigin>Poller’s_PartyID</responseMessageOrigin>
</acknowledgement>
```

3.3 File-Based TransportQ

When the TransportQ is implemented as a file system directory, the file descriptors may be name-value pairs or XML standard files. The fields used in the file system directory have the same name and semantics as the ones used in the relational database table.
3.4 Name-Value-Based File Descriptor
An example name-value-based file descriptor is shown below:
recordId=22
payloadFile=d:\\phinms\\outgoing\\test.txt
destinationFilename=test.txt
routeInfo=CDCStaging service=Router
action=send arguments=XXDOHe1r
messageRecipient=XXDOH

3.5 Sending File Response
An example of a response (written to the acknowledgment file specified in the outgoing file descriptor) from a file send operation is shown below:
transportStatus=success transportError=none
applicationStatus=retrieveSucceeded
applicationError=none
applicationData=TargetTable=payroll
responseLocalFile=1018379449158
responseFileName=test.txt responseSignature=unsigned
responseMessageOrigin=Poller’s_PartyID
4.0 JDBC DRIVERS AND SYNTAX INFORMATION

PHINMS has tested the Java Database Connectivity (JDBC) drivers to connect to the supported databases shown in Table 5. Based on these tests, no issues were found. PHINMS does not guarantee or support the JDBC drivers shown here. It is up to the PHINMS customer to decide which JDBC driver to use. The table is provided for reference only.

<table>
<thead>
<tr>
<th>DB SERVER</th>
<th>VERSION</th>
<th>JDBC DRIVER NAME</th>
<th>TYPE</th>
<th>VERSION</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS SQL</td>
<td>2005</td>
<td>sqljdbc.jar</td>
<td>4</td>
<td>1.2.2828</td>
<td>10/11/2007</td>
</tr>
<tr>
<td>MS SQL</td>
<td>2008</td>
<td>sqljdbc4.jar</td>
<td>4</td>
<td>2.0</td>
<td>03/25/2009</td>
</tr>
<tr>
<td>Oracle</td>
<td>10g Rel 2</td>
<td>ojdbc14.jar</td>
<td>4</td>
<td>10.2.0.2</td>
<td>01/22/2006</td>
</tr>
<tr>
<td>Oracle</td>
<td>11g Rel 1</td>
<td>ojdbc6.jar</td>
<td>4</td>
<td>11.1.0.7.0</td>
<td>08/28/2008</td>
</tr>
<tr>
<td>MySQL</td>
<td>5.0.67</td>
<td>mysql-connector-java-5.1.6-bin.jar</td>
<td>4</td>
<td>3.51.27</td>
<td>11/20/2008</td>
</tr>
</tbody>
</table>

Table 5. JDBC Drivers

4.1 SQL Server

4.1.1 SQL 2000

JDBC Driver ----- com.microsoft.jdbc.sqlserver.SQLServerDriver

Database URL PreFix ----- jdbc:microsoft:sqlserver:

Database URL Suffix ----- //(Computer Name):(Port);DatabaseName=(Name of Database)

4.1.2 SQL 2005

JDBC Driver ----- com.microsoft.sqlserver.jdbc.SQLServerDriver

Database URL PreFix ----- jdbc:sqlserver:

Database URL Suffix ----- //(Computer Name):(Port);DatabaseName=(Name of Database);encrypt=true;trustServerCertificate=true < add only if encryption enabled on DB >

4.1.3 SQL 2008

JDBC Driver ----- com.microsoft.sqlserver.jdbc.SQLServerDriver

Database URL PreFix ----- jdbc:sqlserver:

Database URL Suffix ----- //(Computer Name):(Port);DatabaseName=(Name of Database)
4.2 HSQL
   JDBC Drive: org.hsqldb.jdbcDriver
   Database URL PreFix: jdbc:hsqldb:hsq1:
   Database URL Suffix: //(Computer Name):(Port)/(Name of Database)

4.3 MS Access
   JDBC Driver:   sun.jdbc.odbc.JdbcOdbcDriver
   Database URL PreFix:   jdbc:odbc:
   Database URL Suffix:   PhinmsgAccessDSN270

4.4 MySQL
   JDBC Driver:   com.mysql.jdbc.Driver
   Database URL PreFix:   jdbc:mysql:
   Database URL Suffix:   //(Computer Name):(Port)/(Name of Database)

4.5 Oracle
   JDBC Driver:   oracle.jdbc.driver.OracleDriver
   Database URL PreFix:   jdbc:oracle:thin:
   Database URL Suffix:   @(Computer Name):(Port):(Name of Database)

Note: Remove the parentheses “( )” from the URL suffix when the Computer Name, Port, and Name of Database are inserted. Be careful not to remove any other characters.
5.0 ADVANCED CONSOLE INFORMATION

5.1 Transport Status and Error Codes

The following tables show status and error codes that may be written to the message queues based on the outcome of the message delivery or processing. Applications that use the PHINMS system can read these codes and act on them.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>Message send/receive operation successful.</td>
</tr>
<tr>
<td>Failure</td>
<td>Message send/receive operation failure.</td>
</tr>
</tbody>
</table>

Table 6. Status Codes

<table>
<thead>
<tr>
<th>ERRORCODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityFailure</td>
<td>Error logging into message receiver.</td>
</tr>
<tr>
<td>DeliveryFailure</td>
<td>Failed to deliver message.</td>
</tr>
<tr>
<td>NotSupported</td>
<td>Format of ebXML message or CPA unsupported.</td>
</tr>
<tr>
<td>Unknown</td>
<td>Not a standard ebXML error.</td>
</tr>
<tr>
<td>NoSuchService (*)</td>
<td>Service/action did not map to a service on the message receiver.</td>
</tr>
<tr>
<td>CheckSumFailure (*)</td>
<td>File checksum verification failure at the message receiver.</td>
</tr>
</tbody>
</table>

Table 7. Error Codes

Note: The asterisk (*) symbol indicates a custom error code, meaning the code is not in the ebXML specifications.

6.0 TABLE SCRIPTS

The table scripts identified in the following sections are examples for a database administrator to use to create tables for senders and receivers. The PHINMS account permissions needed to create tables are as follows:

- read
- write
- insert
- update

6.1 MSSQL Scripts

Section 6.1 lists the scripts used to create MSSQL databases.

6.1.1 TransportQ Table - Sender

CREATE TABLE [dbo].[TransportQ_out] (  
    [recordId] [bigint] IDENTITY (1, 1) NOT NULL ,  
    [messageId] [char] (255) NULL ,  
    [payloadFile] [char] (255) NULL ,  
    [payloadContent] [image] NULL ,  
    [destinationFilename] [char] (255) NULL ,  
)
[routeInfo] [char] (255) NOT NULL ,
[service] [char] (255) NOT NULL ,
[action] [char] (255) NOT NULL ,
[arguments] [char] (255) NULL ,
[messageRecipient] [char] (255) NULL ,
[messageCreationTime] [char] (255) NULL ,
[encryption] [char] (10) NOT NULL ,
[signature] [char] (10) NOT NULL ,
[publicKeyLdapAddress] [char] (255) NULL ,
[publicKeyLdapBaseDN] [char] (255) NULL ,
[publicKeyLdapDN] [char] (255) NULL ,
[certificateURL] [char] (255) NULL ,
[processingStatus] [char] (255) NULL ,
[transportStatus] [char] (255) NULL ,
[transportErrorCode] [char] (255) NULL ,
[applicationStatus] [char] (255) NULL ,
[applicationErrorCode] [char] (255) NULL ,
[applicationResponse] [char] (255) NULL ,
[messageSentTime] [char] (255) NULL ,
[messageReceivedTime] [char] (255) NULL ,
[responseMessageId] [char] (255) NULL ,
[responseArguments] [char] (255) NULL ,
[responseLocalFile] [char] (255) NULL ,
[responseFilename] [char] (255) NULL ,
[responseContent] [image] NULL ,
[responseMessageOrigin] [char] (255) NULL ,
[responseMessageSignature] [char] (255) NULL ,
[priority] [int] NULL
)
ON [PRIMARY] TEXTIMAGE_ON [PRIMARY] GO

6.1.2 RnRworkerQ Table - Sender
CREATE TABLE [dbo].[Sender_inq] (  
[recordId] [bigint] IDENTITY (1, 1) NOT NULL ,  
[messageId] [varchar] (255) NULL ,  
[payloadName] [varchar] (255) NULL ,  
[payloadBinaryContent] [image] NULL ,  
[payloadTextContent] [text] NULL ,  
[localFileName] [varchar] (255) NOT NULL ,  
[service] [varchar] (255) NOT NULL ,  
[action] [varchar] (255) NOT NULL ,  
[arguments] [varchar] (255) NULL ,  
[fromPartyId] [varchar] (255) NULL ,  
[messageRecipient] [varchar] (255) NULL ,  
[errorCode] [varchar] (255) NULL ,  
[errorMessage] [varchar] (255) NULL ,
6.1.3 ErrorQ Table - Sender
CREATE TABLE [dbo].[PHINMS_errq] (  
[recordId] [bigint] IDENTITY (1, 1) NOT NULL ,  
[messageId] [varchar] (255) NULL ,  
[payloadName] [varchar] (255) NULL ,  
[payloadBinaryContent] [image] NULL ,  
[payloadTextContent] [text] NULL ,  
[localFileName] [varchar] (255) NOT NULL ,  
[service] [varchar] (255) NOT NULL ,  
[action] [varchar] (255) NOT NULL ,  
[arguments] [varchar] (255) NULL ,  
[fromPartyId] [varchar] (255) NULL ,  
[messageRecipient] [varchar] (255) NULL ,  
[errorCode] [varchar] (255) NULL ,  
[errorMessage] [varchar] (255) NULL ,  
[processingStatus] [varchar] (255) NULL ,  
[applicationStatus] [varchar] (255) NULL ,  
[encryption] [varchar] (10) NOT NULL ,  
[receivedTime] [varchar] (255) NULL ,  
[lastUpdateTime] [varchar] (255) NULL ,  
[processId] [varchar] (255) NULL  
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY] GO

6.1.4 Messaging Cache Table - Sender
CREATE TABLE [dbo].[messagingcache] (  
[sequence] [int] IDENTITY (1, 1) NOT NULL ,  
[partyId] [char] (50) NULL ,  
[convId] [char] (50) NULL ,  
[recordId] [char] (50) NULL ,  
[response] [text] NULL ,  
[timestamp] [char] (20) NULL ,  
[status] [char] (10) NULL  
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY] GO

6.1.5 Messaging Queue Table - Receiver
CREATE TABLE [dbo].[message_inq] (  
[recordId] [bigint] IDENTITY (1, 1) NOT NULL ,  
[messageId] [varchar] (255) NULL ,
6.1.6 TransportQ Table - Receiver

CREATE TABLE [dbo].[PTD_outq] (  
    [recordId] [bigint] IDENTITY (1, 1) NOT NULL ,  
    [messageId] [varchar] (255) NULL ,  
    [payloadName] [varchar] (255) NULL ,  
    [payloadBinaryContent] [image] NULL ,  
    [payloadTextContent] [text] NULL ,  
    [localFileName] [varchar] (255) NOT NULL ,  
    [service] [varchar] (255) NOT NULL ,  
    [action] [varchar] (255) NOT NULL ,  
    [arguments] [varchar] (255) NULL ,  
    [fromPartyId] [varchar] (255) NULL ,  
    [messageRecipient] [varchar] (255) NULL ,  
    [errorCode] [varchar] (255) NULL ,  
    [errorMessage] [varchar] (255) NULL ,  
    [processingStatus] [varchar] (255) NULL ,  
    [applicationStatus] [varchar] (255) NULL ,  
    [encryption] [varchar] (10) NOT NULL ,  
    [receivedTime] [varchar] (255) NULL ,  
    [lastUpdateTime] [varchar] (255) NULL ,  
    [processId] [varchar] (255) NULL  
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY] GO

6.1.7 ErrorQ Table - Receiver

CREATE TABLE [dbo].[message_errq] (  
    [recordId] [bigint] IDENTITY (1, 1) NOT NULL ,  
    [messageId] [varchar] (255) NULL ,  
    [payloadName] [varchar] (255) NULL ,  
    [payloadBinaryContent] [image] NULL ,  
    [payloadTextContent] [text] NULL ,  
    [localFileName] [varchar] (255) NOT NULL ,  
    [service] [varchar] (255) NOT NULL ,  
    [action] [varchar] (255) NOT NULL ,  
    [arguments] [varchar] (255) NULL ,  
    [fromPartyId] [varchar] (255) NULL ,  
    [messageRecipient] [varchar] (255) NULL ,  
    [errorCode] [varchar] (255) NULL ,  
    [errorMessage] [varchar] (255) NULL ,  
    [processingStatus] [varchar] (255) NULL ,  
    [applicationStatus] [varchar] (255) NULL ,  
    [encryption] [varchar] (10) NOT NULL ,  
    [receivedTime] [varchar] (255) NULL ,  
    [lastUpdateTime] [varchar] (255) NULL ,  
    [processId] [varchar] (255) NULL  
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY] GO
6.2 Oracle Scripts

6.2.1 TransportQ Table - Sender

CREATE TABLE TransportQ_out ( recordId number(19,0) NOT NULL , messageId char (255) NULL , payloadFile char (255) NULL , payloadContent BLOB NULL , destinationFilename char (255) NULL , routeInfo char (255) NOT NULL , service char (255) NOT NULL , action char (255) NOT NULL , arguments char (255) NULL , messageRecipient char (255) NULL , messageCreationTime char (255) NULL , encryption char (10) NOT NULL , signature char (10) NOT NULL , publicKeyLdapAddress char (255) NULL , publicKeyLdapBaseDN char (255) NULL , publicKeyLdapDN char (255) NULL , certificateURL char (255) NULL , processingStatus char (255) NULL , transportStatus char (255) NULL , transportErrorCode char (255) NULL , applicationStatus char (255) NULL , applicationErrorCode char (255) NULL , applicationResponse char (255) NULL , messageSentTime char (255) NULL , messageReceivedTime char (255) NULL , responseMessageId char (255) NULL ,
CREATE SEQUENCE TransportQ_out_recordId
    START WITH 1
    INCREMENT BY 1;

CREATE TRIGGER TransportQ_out_IDENTITY
before insert on TransportQ_out for each row begin
    select TransportQ_out_recordId.nextval into :new.recordId from dual;
end; /

6.2.2 WorkerQ Table - Sender
CREATE TABLE Sender_inq(
    recordId number(19,0) NOT NULL ,
    messageId varchar2 (255) NULL ,
    payloadName varchar2 (255) NULL ,
    payloadBinaryContent BLOB NULL ,
    payloadTextContent CLOB NULL ,
    localFileName var char2 (255) NOT NULL ,
    service varchar2 (255) NOT NULL ,
    action varchar2 (255) NOT NULL ,
    arguments varchar2 (255) NULL ,
    fromPartyId varchar2 (255) NULL ,
    messageRecipient varchar2 (255) NULL ,
    errorCode varchar2 (255) NULL ,
    errorMessage varchar2 (255) NULL ,
    processingStatus varchar2 (255) NULL ,
    applicationStatus varchar2 (255) NULL ,
    encryption varchar2 (10) NOT NULL ,
    receivedTime varchar2 (255) NULL ,
    lastUpdateTime varchar2 (255) NULL ,
    processId varchar2 (255) NULL
) ;

ALTER TABLE Sender_inq ADD
PRIMARY KEY (recordId);

CREATE SEQUENCE Sender_inq_record_count
INCREMENT BY 1 START WITH 1 MINVALUE 1 MAXVALUE 9999999999999999999999999999
NOCYCLE
NOORDER
CACHE 20;
CREATE TRIGGER Sender_inq_IDENTITY
before insert on Sender_inq for
each row begin
select Sender_inq_record_count.nextval into :new.recordId from dual;
end;

6.2.3 ErrorQ Table - Sender
CREATE TABLE PHINMS_errq
(recordId number(19,0) NOT NULL,
messageId varchar2(255) NULL,
payloadName varchar2(255) NULL,
payloadBinaryContent BLOB NULL,
localFileName varchar2(255) NOT NULL,
service varchar2(255) NOT NULL,
action varchar2(255) NOT NULL,
arguments varchar2(255) NULL,
fromPartyId varchar2(255) NULL,
messageRecipient varchar2(255) NULL,
errorCode varchar2(255) NULL,
errorMessage varchar2(255) NULL,
processingStatus varchar2(255) NULL,
applicationStatus varchar2(255) NULL,
encryption varchar2(10) NOT NULL,
receivedTime varchar2(255) NULL,
lastUpdateTime varchar2(255) NULL,
processId varchar2(255) NULL);

CREATE SEQUENCE PHINMS_errq_recordID
START WITH 1
INCREMENT BY 1;

CREATE TRIGGER PHINMS_errq_IDENTITY
before insert on PHINMS_errq for
each row begin
select PHINMS_errq_recordID.nextval into :new.recordId from dual;
end; /

6.2.4 Messaging Cache Table - Sender
CREATE TABLE messagingcache
(sequence number(10,0) NOT NULL,
partyId char(50) NULL,
convId char(50) NULL,
recordId char(50) NULL,
response CLOB NULL,
timestamp char(20) NULL,
status char(10) NULL);

CREATE SEQUENCE messagingcache_sequence
START WITH 1
INCREMENT BY 1;

CREATE TRIGGER messagingcache_IDENTITY
before insert on messagingcache for
each row begin
select messagingcache_sequence.nextval into :new.sequence from dual;
end; /

6.2.5 Messaging Queue Table - Receiver
CREATE TABLE message_inq (    recordId
number(19,0) NOT NULL ,  messageId varchar2
(255) NULL ,  payloadName varchar2 (255)
NULL ,  payloadBinaryContent BLOB NULL ,
payloadTextContent CLOB NULL ,
localFileName varchar2 (255) NOT NULL ,
service varchar2 (255) NOT NULL ,  action
varchar2 (255) NOT NULL ,  arguments
varchar2 (255) NULL ,  fromPartyId varchar2
(255) NULL ,  messageRecipient varchar2
(255) NULL ,  errorCode varchar2 (255)
NULL ,  errorMessage varchar2 (255) NULL ,
processingStatus varchar2 (255) NULL ,
applicationStatus varchar2 (255) NULL ,
encryption varchar2 (10) NOT NULL ,
receivedTime varchar2 (255) NULL ,
lastUpdateTime varchar2 (255) NULL ,
processId varchar2 (255) NULL
) ;

CREATE SEQUENCE message_inq_record_count
START WITH 1
INCREMENT BY 1;

CREATE TRIGGER message_inq_IDENTITY
before insert on message_inq for
each row begin
select message_inq_record_count.nextval into :new.recordId from dual;
end; /

6.2.6 TransportQ Table - Receiver
CREATE TABLE PTD_outq (    recordId number(19,0)
NOT NULL ,  messageId varchar2 (255) NULL ,
payloadName varchar2 (255) NULL ,
payloadBinaryContent BLOB NULL ,
payloadTextContent CLOB NULL ,
localFileName varchar2 (255) NOT NULL ,
service varchar2 (255) NOT NULL ,  action
varchar2 (255) NOT NULL ,  arguments
varchar2 (255) NULL ,  fromPartyId varchar2
(255) NULL, messageRecipient varchar2 (255) NULL, errorCode varchar2 (255) NULL, errorMessage varchar2 (255) NULL, processingStatus varchar2 (255) NULL, applicationStatus varchar2 (255) NULL, encryption varchar2(10) NOT NULL, receivedTime varchar2 (255) NULL, lastUpdateTime varchar2 (255) NULL, processId varchar2 (255) NULL);

CREATE SEQUENCE PTD_outq_recordID
START WITH 1
INCREMENT BY 1;

CREATE TRIGGER PTD_outq_IDENTITY
before insert on message_outq
for each row begin
select PTD_outq_recordID.nextval into :new.recordId from dual;
end; /

6.2.7 ErrorQ Table - Receiver
CREATE TABLE message_errq (recordId number(19,0) NOT NULL, messageId varchar2 (255) NOT NULL, payloadName varchar2 (255) NULL, payloadBinaryContent BLOB NULL, payloadTextContent CLOB NULL, localFileName varchar2 (255) NOT NULL, service varchar2 (255) NOT NULL, action varchar2 (255) NOT NULL, arguments varchar2 (255) NULL, fromPartyId varchar2 (255) NULL, messageRecipient varchar2 (255) NULL, errorCode varchar2 (255) NULL, errorMessage varchar2 (255) NULL, processingStatus varchar2 (255) NULL, applicationStatus varchar2 (255) NULL, encryption varchar2 (10) NOT NULL, receivedTime varchar2 (255) NULL, lastUpdateTime varchar2 (255) NULL, processId varchar2 (255) NULL);

CREATE SEQUENCE message_errq_recordID
START WITH 1
INCREMENT BY 1;
CREATE TRIGGER message_errq_IDENTITY
before insert on message_errq for
each row begin
select message_errq_recordID.nextval into :new.recordId from dual;
end; /

6.2.8 Route-not-Read Table - Sender

CREATE TABLE broadcast (  
name char (100) NULL ,
addresses char (1000) NULL  
);

CREATE TABLE messagebins (   recordId
number(19,0) NOT NULL , messageId varchar2
(255) NULL , payloadName varchar2 (255) NULL , payloadBinaryContent BLOB NULL , payloadTextContent CLOB NULL , localFileName varchar2 (255) NULL , service varchar2 (255) NOT NULL , action varchar2 (255) NOT NULL , arguments varchar2 (255) NULL , fromPartyId varchar2
(255) NULL , messageRecipient varchar2
(255) NULL , errorCode varchar2 (255) NULL , errorMessage varchar2 (255) NULL , processingStatus varchar2 (255) NULL , applicationStatus varchar2 (255) NULL , encryption varchar2 (10) NOT NULL , receivedTime varchar2 (255) NULL , lastUpdateTime varchar2 (255) NULL , processId varchar2 (255) NULL  
);

CREATE SEQUENCE messagebins_recordId
START WITH 1
INCREMENT BY 1;

CREATE TRIGGER messagebins_IDENTITY
before insert on messagebins for
each row begin
select messagebins_recordId.nextval into :new.recordId from dual;
end; /

CREATE TABLE partyid_user (  
partyId char (255) NULL ,
"user" char (255) NULL ,
sdnuser char (255) NULL  
);
CREATE TABLE users ( name char (100) NULL,
    description char (255) NULL );

6.3 MySQL Scripts

6.3.1 TransportQ Table - Sender
CREATE TABLE TransportQ_out ( recordId bigint NOT NULL AUTO_INCREMENT, 
    messageId char (255) NULL, 
    payloadFile char (255) NULL, 
    payloadContent LONGBLOB NULL, 
    destinationFilename char (255) NULL, 
    routeInfo char (255) NOT NULL, 
    service char (255) NOT NULL, 
    action char (255) NOT NULL, 
    arguments char (255) NULL, 
    messageRecipient char (255) NULL, 
    messageCreationTime char (255) NULL, 
    encryption char (10) NOT NULL, 
    signature char (10) NOT NULL, 
    publicKeyLdapAddress char (255) NULL, 
    publicKeyLdapBaseDN char (255) NULL, 
    publicKeyLdapDN char (255) NULL, 
    certificateURL char (255) NULL, 
    processingStatus char (255) NULL, 
    transportStatus char (255) NULL, 
    transportErrorCode char (255) NULL, 
    applicationStatus char (255) NULL, 
    applicationErrorCode char (255) NULL, 
    applicationResponse char (255) NULL, 
    messageSentTime char (255) NULL, 
    messageReceivedTime char (255) NULL, 
    messageIdId char (255) NULL, 
    responseArguments char (255) NULL, 
    responseLocalFile char (255) NULL, 
    responseFilename char (255) NULL, 
    responseContent LONGBLOB NULL, 
    responseMessageOrigin char (255) NULL, 
    responseMessageSignature char (255) NULL, 
    priority int NULL, 
    PRIMARY KEY (recordId) );

6.3.2 WorkerQ Table - Sender
CREATE TABLE Sender_inq ( recordId bigint NOT NULL AUTO_INCREMENT, 
    messageIdId varchar (255) NULL, 
    payloadName varchar (255) NULL, 
    payloadBinaryContent LONGBLOB NULL, 
    payloadTextContent LONGBLOB NULL, 
    localFileName varchar (255) NULL, 
    service varchar (255) NOT NULL, 
    actionId varchar (255) NOT NULL, 
    arguments
6.3.3 Error Q Table - Sender
CREATE TABLE PHINMS_errq ( recordId bigint NOT NULL AUTO_INCREMENT, messageId varchar (255) NULL, payloadName varchar (255) NULL, payloadBinaryContent LONGBLOB NULL, payloadTextContent LONGTEXT NULL, localFileName varchar (255) NOT NULL, service varchar (255) NOT NULL, action varchar (255) NOT NULL, arguments varchar (255) NULL, fromPartyId varchar (255) NULL, messageRecipient varchar (255) NULL, errorCode varchar (255) NULL, errorMessage varchar (255) NULL, processingStatus varchar (255) NULL, applicationStatus varchar (255) NULL, encryption varchar (10) NOT NULL, receivedTime varchar (255) NULL, lastUpdateTime varchar (255) NULL, processId varchar (255) NULL, PRIMARY KEY (recordId) );

6.3.4 Messaging Cache Table - Sender
CREATE TABLE messagingcache ( sequence int NOT NULL AUTO_INCREMENT, partyId char (50) NULL, convId char (50) NULL, recordId char (50) NULL, response LONGTEXT NULL, timestamp char (20) NULL, status char (10) NULL, PRIMARY KEY (sequence) );
6.3.5 Messaging Queue Table - Receiver

CREATE TABLE message_inq ( recordId bigint NOT NULL AUTO_INCREMENT, messageId varchar (255) NULL , payloadName varchar (255) NULL , payloadBinaryContent LONGBLOB NULL , payloadTextContent LONGTEXT NULL , localFileName varchar (255) NOT NULL , service varchar (255) NOT NULL , action varchar (255) NOT NULL , arguments varchar (255) NULL , fromPartyId varchar (255) NULL , messageRecipient varchar (255) NULL , errorCode varchar (255) NULL , errorMessage varchar (255) NULL , processingStatus varchar (255) NULL , applicationStatus varchar (255) NULL , encryption varchar (10) NOT NULL , receivedTime varchar (255) NULL , lastUpdateTime varchar (255) NULL , processId varchar (255) NULL, PRIMARY KEY (recordId) );

6.3.6 TransportQ - Receiver

CREATE TABLE PTD_outq ( recordId bigint NOT NULL AUTO_INCREMENT, messageId varchar (255) NULL , payloadName varchar (255) NULL , payloadBinaryContent LONGBLOB NULL , payloadTextContent LONGTEXT NULL , localFileName varchar (255) NOT NULL , service varchar (255) NOT NULL , action varchar (255) NOT NULL , arguments varchar (255) NULL , fromPartyId varchar (255) NULL , messageRecipient varchar (255) NULL , errorCode varchar (255) NULL , errorMessage varchar (255) NULL , processingStatus varchar (255) NULL , applicationStatus varchar (255) NULL , encryption varchar (10) NOT NULL , receivedTime varchar (255) NULL , lastUpdateTime varchar (255) NULL , processId varchar (255) NULL, PRIMARY KEY (recordId) );

6.3.7 Message ErrorQ - Receiver

CREATE TABLE message_errq ( recordId bigint NOT NULL AUTO_INCREMENT, messageId varchar (255) NULL , payloadName varchar (255) NULL , payloadBinaryContent LONGBLOB NULL , payloadTextContent LONGTEXT NULL , localFileName varchar (255) NOT NULL,
CREATE TABLE [dbo].[broadcast] (  [name] [char] (100) NULL ,  [addresses] [char] (1000) NULL ) ON [PRIMARY]  GO

CREATE TABLE [dbo].[messagebins] (  [recordId] [bigint] IDENTITY (1, 1) NOT NULL ,  [messageId] [varchar] (255) NULL ,  [payloadName] [varchar] (255) NULL ,  [payloadBinaryContent] [image] NULL ,  [payloadTextContent] [text] NULL ,  [localFileName] [varchar] (255) NULL ,  [service] [varchar] (255) NOT NULL ,  [action] [varchar] (255) NOT NULL ,  [arguments] [varchar] (255) NULL ,  [fromPartyId] [varchar] (255) NULL ,  [messageRecipient] [varchar] (255) NULL ,  [errorCode] [varchar] (255) NOT NULL ,  [errorMessage] [varchar] (255) NULL ,  [processingStatus] [varchar] (255) NULL ,  [applicationStatus] [varchar] (255) NULL ,  [encryption] [varchar] (10) NOT NULL ,  [receivedTime] [varchar] (255) NULL ,  [lastUpdateTime] [varchar] (255) NULL ,  [processId] [varchar] (255) NULL ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]  GO

CREATE TABLE [dbo].[partyid_user] (  [partyId] [char] (255) NULL ,  [user] [char] (255) NULL ,

6.3.8 Route-not-Read Table - Sender

CREATE TABLE [dbo].[broadcast] (  [name] [char] (100) NULL ,  [addresses] [char] (1000) NULL ) ON [PRIMARY]  GO

CREATE TABLE [dbo].[messagebins] (  [recordId] [bigint] IDENTITY (1, 1) NOT NULL ,  [messageId] [varchar] (255) NULL ,  [payloadName] [varchar] (255) NULL ,  [payloadBinaryContent] [image] NULL ,  [payloadTextContent] [text] NULL ,  [localFileName] [varchar] (255) NULL ,  [service] [varchar] (255) NOT NULL ,  [action] [varchar] (255) NOT NULL ,  [arguments] [varchar] (255) NULL ,  [fromPartyId] [varchar] (255) NULL ,  [messageRecipient] [varchar] (255) NULL ,  [errorCode] [varchar] (255) NOT NULL ,  [errorMessage] [varchar] (255) NULL ,  [processingStatus] [varchar] (255) NULL ,  [applicationStatus] [varchar] (255) NULL ,  [encryption] [varchar] (10) NOT NULL ,  [receivedTime] [varchar] (255) NULL ,  [lastUpdateTime] [varchar] (255) NULL ,  [processId] [varchar] (255) NULL ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]  GO

CREATE TABLE [dbo].[partyid_user] (  [partyId] [char] (255) NULL ,  [user] [char] (255) NULL ,


CREATE TABLE [dbo].[users] (  
    [name] [char] (100) NULL ,  
    [description] [char] (255) NULL  
) ON [PRIMARY] GO

6.4 HSQL Scripts

6.4.1 TransportQ Table - Sender
CREATE TABLE TransportQ_out (  
    recordId bigint NOT NULL IDENTITY,  
    messageId varchar(255) NULL ,  
    payloadFile char (255) NULL ,  
    payloadContent LONGBINARY NULL ,  
    destinationFilename char (255) NULL ,  
    routeInfo char (255) NOT NULL ,  
    service char (255) NOT NULL ,  
    action char (255) NOT NULL ,  
    arguments char (255) NULL ,  
    messageRecipient char (255) NULL ,  
    messageCreationTime char (255) NULL ,  
    encryption char (10) NOT NULL ,  
    signature char (10) NOT NULL ,  
    publicKeyLdapAddress char (255) NULL ,  
    publicKeyLdapBaseDN char (255) NULL ,  
    publicKeyLdapDN char (255) NULL ,  
    certificateURL char (255) NULL ,  
    processingStatus char (255) NULL ,  
    transportStatus char (255) NULL ,  
    transportErrorCode char (255) NULL ,  
    applicationStatus char (255) NULL ,  
    applicationErrorCode char (255) NULL ,  
    applicationResponse char (255) NULL ,  
    messageSentTime char (255) NULL ,  
    messageReceivedTime char (255) NULL ,  
    responseMessageId char (255) NULL ,  
    responseArguments char (255) NULL ,  
    responseLocalFile char (255) NULL ,  
    responseFilename char (255) NULL ,  
    responseContent LONGBINARY NULL ,  
    responseMessageOrigin char (255) NULL ,  
    responseMessageSignature char (255) NULL ,  
    priority int NULL  
)

6.4.2 WorkerQ Table - Sender
CREATE TABLE Sender_inq (  
    recordId bigint NOT NULL IDENTITY,  
    messageId varchar(255) NULL ,  
    payloadName varchar (255) NULL ,  
    payloadContent LONGBINARY NULL ,  
    service char (255) NOT NULL ,  
    action char (255) NOT NULL ,  
    arguments char (255) NULL ,  
    messageRecipient char (255) NULL ,  
    messageCreationTime char (255) NULL ,  
    encryption char (10) NOT NULL ,  
    signature char (10) NOT NULL ,  
    publicKeyLdapAddress char (255) NULL ,  
    publicKeyLdapBaseDN char (255) NULL ,  
    publicKeyLdapDN char (255) NULL ,  
    certificateURL char (255) NULL ,  
    processingStatus char (255) NULL ,  
    transportStatus char (255) NULL ,  
    transportErrorCode char (255) NULL ,  
    applicationStatus char (255) NULL ,  
    applicationErrorCode char (255) NULL ,  
    applicationResponse char (255) NULL ,  
    messageSentTime char (255) NULL ,  
    messageReceivedTime char (255) NULL ,  
    responseMessageId char (255) NULL ,  
    responseArguments char (255) NULL ,  
    responseLocalFile char (255) NULL ,  
    responseFilename char (255) NULL ,  
    responseContent LONGBINARY NULL ,  
    responseMessageOrigin char (255) NULL ,  
    responseMessageSignature char (255) NULL ,  
    priority int NULL  
)
NULL, payloadBinaryContent LONGVARBINARY
NULL, payloadTextContent LONGVARCHAR
NULL, localFileName varchar (255) NOT
NULL, service varchar (255) NOT NULL,
action varchar (255) NOT NULL, arguments
varchar (255) NULL, fromPartyId varchar
(255) NULL, messageRecipient varchar
(255) NULL, errorCode varchar (255) NULL
, errorMessage varchar (255) NULL,
processingStatus varchar (255) NULL,
applicationStatus varchar (255) NULL,
encryption varchar (10) NOT NULL,
receivedTime varchar (255) NULL,
lastUpdateTime varchar (255) NULL,
processId varchar (255) NULL )

6.4.3 ErrorQ Table - Sender
CREATE TABLE PHINMS_errq (  recordId bigint NOT NULL IDENTITY,
messageId varchar (255) NULL,
payloadName varchar (255) NULL,
payloadBinaryContent LONGVARBINARY
NULL, payloadTextContent LONGVARCHAR
NULL, localFileName varchar (255) NOT
NULL, service varchar (255) NOT NULL,
action varchar (255) NOT NULL, arguments
varchar (255) NULL, fromPartyId varchar
(255) NULL, messageRecipient varchar
(255) NULL, errorCode varchar (255) NULL
, errorMessage varchar (255) NULL,
processingStatus varchar (255) NULL,
applicationStatus varchar (255) NULL,
encryption varchar (10) NOT NULL,
receivedTime varchar (255) NULL,
lastUpdateTime varchar (255) NULL,
processId varchar (255) NULL )

6.4.4 Messaging Cache Table - Sender
CREATE TABLE messagingcache (  sequence int NOT NULL IDENTITY,
partyId char (50) NULL,
convId char (50) NULL,
recordId char (50) NULL,
response LONGVARCHAR NULL, timestamp
char (20) NULL,
status char (10) NULL )

6.4.5 Messaging Queue Table – Receiver
CREATE TABLE message_inq ( recordId bigint NOT NULL IDENTITY,
messageId varchar (255) NULL,
payloadName varchar (255) NULL,
payloadBinaryContent LONGVARBINARY NULL ,
payloadTextContent LONGVARCHAR NULL ,
localFileName varchar (255) NOT NULL ,
service varchar (255) NOT NULL , action
varchar (255) NOT NULL , arguments varchar
(255) NULL , fromPartyId varchar (255) NULL
, messageRecipient varchar (255) NULL ,
errorCode varchar (255) NULL , errorMessage
varchar (255) NULL , processingStatus
varchar (255) NULL , applicationStatus
varchar (255) NULL , encryption varchar (10)
NOT NULL , receivedTime varchar (255) NULL ,
lastUpdateTime varchar (255) NULL ,
processId varchar (255) NULL )

6.4.6 TransportQ Table - Receiver
CREATE TABLE PTD_outq ( recordId bigint NOT
NULL IDENTITY, messageId varchar (255) NULL
, payloadName varchar (255) NULL ,
payloadBinaryContent LONGVARBINARY NULL ,
payloadTextContent LONGVARCHAR NULL ,
localFileName varchar (255) NOT NULL ,
service varchar (255) NOT NULL , action
varchar (255) NOT NULL , arguments
varchar (255) NULL , fromPartyId varchar
(255) NULL , messageRecipient varchar
(255) NULL , errorCode varchar (255) NULL
, errorMessage varchar (255) NULL ,
processingStatus varchar (255) NULL ,
applicationStatus varchar (255) NULL ,
encryption varchar (10) NOT NULL , receivedTime
varchar (255) NULL , lastUpdateTime varchar
(255) NULL , processId varchar (255) NULL )

6.4.7 ErrorQ Table - Receiver
CREATE TABLE message_errq ( recordId bigint NOT
NULL IDENTITY, messageId varchar
(255) NULL , payloadName varchar (255)
NULL , payloadBinaryContent LONGVARBINARY
NULL , payloadTextContent LONGVARCHAR
NULL , localFileName varchar (255) NOT
NULL , service varchar (255) NOT NULL ,
action varchar (255) NOT NULL , arguments
varchar (255) NULL , fromPartyId varchar
(255) NULL , messageRecipient varchar
(255) NULL , errorCode varchar (255) NULL
, errorMessage varchar (255) NULL ,
processingStatus varchar (255) NULL ,
applicationStatus varchar (255) NULL ,
encryption varchar (10) NOT NULL,
receivedTime varchar (255) NULL,
lastUpdateTime varchar (255) NULL,
processId varchar (255) NULL)