Technical Guide 3.2

Public Health Information Network Messaging System (PHINMS)

Version 1.0

Prepared by:
Centers for Disease Control and Prevention

March 31, 2021

EXECUTIVE SUMMARY

Many organizations work together to protect and advance public health. These organizations use the internet to securely exchange sensitive data between various 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 data security, data encryption, methods for dealing with various firewalls, and internet protection schemes. The Centers for Disease Control and Prevention (CDC) Public Health Information Network Messaging System (PHINMS) is the software that facilitates this secure messaging. PHINMS provides a standard way for addressing and routing content and a standard and consistent way for information systems to confirm an exchange. It 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 3.2 application.

REVISION HISTORY

VERSION #	IMPLEMENTER	DATE	EXPLANATION
1.0	The PHINMS Team	12-30-2020	Update PHINMS 3.2 Technical Guide from previous versions.
1.0	Jannie Williams	03-17-2021	Review and update.

TABLE OF CONTENTS

LIST OF TABLES

ACRONYM LIST

API	Application Programming Interface
BA	Basic Authentication
CA	Certificate Authority
CDC	Centers for Disease Control and Prevention
CPA	Collaboration Protocol Agreement
CSE	Communications Security Establishment
DNS	Domain Name System
ebMS	Electronic Business Extensible Markup Language Messaging Service
ebXML	Electronic Business Extensible Markup Language
EJB	Enterprise JavaBeans
ErrorQ	Error Queue
FIPS	Federal Information Processing Standard
HF3	Hot Fix 3 (repackaged installer will work with 32 or 64 bit java)
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
IMAP	Internet Message Access Protocol
J2EE	Java 2 Platform Enterprise Edition
J2SE	Java 2 Platform Standard Edition
JDBC	Java Database Connectivity
JDK	Java Development Kit
JVM	Java Virtual Machine
ITL	Information Technology Laboratory
LDAP	Lightweight Directory Access Protocol
NIST	National Institute for Standards and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
PHIN	Public Health Information Network
PHINMS	Public Health Information Network Messaging System
PKCS	Public-Key Cryptography Standards
PKI	Public Key Infrastructure
POP	Post Office Protocol
PSK	Pre-Shared Key
OASIS	Organization for the Advancement of Structured Information Standards
ODBC	Open Database Connectivity
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol

SP1	Service Pack 1
SQL	Structured Query Language
SSL	Secure Socket Layer
SSO	Single Sign-On
STP	Secure Transport Protocol
TCP/IP	Transport Control Protocol Internet Protocol
TLS	Transport Layer Security
TransportQ	Transport Queue
URL	Uniform Resource Locator
WorkerQ	Worker Queue
XML	Extensible Markup Language

1.0 INTRODUCITON

The Centers for Disease Control and Prevention (CDC) Public Health Information Network Messaging System (PHINMS) Technical Guide assists users with performing actions manually, outside the graphical user interface configurations. The most recent version of the PHINMS Technical Guide can be found on the PHINMS website. **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 through the PHINMS website by using the Contact Us email link or the phone number located on the left of the home page. The PHINMS support team will respond 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.

FIELD NAME	DESCRIPTION	SOURCE	OPTION
recordId	Unique ID of the record in the table and the table's primary key.	Auto Generated	Mandatory
messageId	Application-level message identifier.	Application	Optional
payloadFile	File name of the payload file of an outgoing message relative to a local directory such as myinputs.txt.	Application	Optional
payloadContent	Used only when the payloadFile field is not specified. Populates the contents of a file within the table.	Application	Optional
destinationFilename	The name of the payload file when it is stored on the receiver/handler.	Application	Optional
routeInfo	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.	Application	Mandatory
service	ebXML service name – Case sensitive.	Application	Mandatory
action	ebXML action – Case sensitive.	Application	Mandatory
arguments	Arguments specified by the message sender.	Application	Optional
messageCreationTime	Time when record was created, in UTC format.	Sender	Optional
messageRecipient	Recipient's ID specified by the sender in the TransportQ_out.	Sender	Optional
processingStatus	Initial value of the status of record created queued.	Sender	Optional
applicationStatus	Status of the application.	Sender	Optional
encryption	The value is Yes if payload is encrypted and No if it is not.	Application	Mandatory
signature	If Yes, XML signature is applied to the payload.	Application	Optional
publicKeyLdapAddress	LDAP address of the LDAP directory server.	Application	Optional
publicKeyLdapBaseDN	LDAP Base Distinguished Name of the public key such as o=.	Application	Optional
publicKeyLdapDN	LDAP Distinguished Name of the public key such as cn=.	Application	Optional
certificateURL	URL of a recipient's public key certificate.	Application	Optional
transportStatus	Transport-level status.	Sender	Optional
transportErrorCode	Error code describing the transport failure.	Sender	Optional

FIELD NAME	DESCRIPTION	SOURCE	OPTION
applicationErrorCode	The error code returned by the service/action in a synchronous manner.	Sender	Optional
applicationResponse	The synchronous response returned by the service/action.	Sender	Optional
messageSentTime	Time when the message was sent, in UTC format.	Sender	Optional
messageReceivedTime	Time when the message was received, in UTC format.	Sender	Optional
responseMessageId	Message ID of the response message in the route-not-read scenario.	Sender	Optional
responseArguments	Used in the route-not-read scenario to convey arguments being sent by a message sender to a receiving client.	Sender	Optional
responseLocalFile	The response to a poll-type request, which may contain a payload file in the route-not-read scenario.	Sender	Optional
responseFilename	The response file name in the route-not-read scenario.	Sender	Optional
responseContent	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.	Sender	Optional
responseMessageOrigin	The partyID of the party originating the message in the route-not-read scenario.	Sender	Optional
responseMessageSignature	The partyID of the party signing the message in the route-not-read scenario.	Sender	Optional
priority	An integer indicating the request's priority.	Application	Optional

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.

FIELD NAME	DESCRIPTION	SOURCE	OPTION
recordId	Unique ID of the record in the table and the table's primary key.	Receiver	Mandatory
messageId	Application-level message identifier.	Sender	Optional
payloadName	File name of the payload, specified by the message sender.	Sender	Optional
payloadBinaryContent	Image field written to the receiver servlet.	Sender *	Optional
payloadTextContent	Text field populated if textPayload=true in the servicemap entry.	Sender *	Optional
localFilename	File written to disk instead of a database when payloadToDisk =true.	Receiver	Mandatory
service	ebXML service name.	Sender	Mandatory

FIELD NAME	DESCRIPTION	SOURCE	OPTION
action	ebXML action.	Sender	Mandatory
arguments	Arguments specified by the message sender.	Sender	Optional
fromPartyId	PartyID of the message sender.	Sender	Optional
messageRecipient	Recipient's ID specified by the sender in the TransportQ_out.	Sender	Optional
errorCode	Error code.	Receiver	Optional
errorMessage	Error message.	Receiver	Optional
processingStatus	Initial value of the status of record created queued.	Receiver	Optional
applicationStatus	Status of the application.	Receiver	Optional
encryption	The value is Yes if payload stored in WorkerQ is encrypted and No if it is not.	Receiver	Mandatory
receivedTime	Time when payload was received, in UTC format.	Receiver	Optional
lastUpdateTime	Time when record was last updated, in UTC format.	Receiver	Optional
processId	Identifies the process processing the record.	Receiver	Optional

Table 2. WorkerQ Database Fields

(*) 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.

TRANSPORT STATUS CODE	DESCRIPTION
Success	Message send or receive operation succeeded.
Failure	Message send or receive operation failed.

Table 3. Transport Status Codes

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

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:

```
<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:

```
<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=XXDOHelr
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

Java Database Connectivity (JDBC) drivers listed in Table 5 below were successfully tested for connectivity. However, CDC does not guarantee or support any potential defect of JDBC drivers themselves as listed in Table 5.b It is up to PHINMS users to determine the appropriate JDBC driver to use. The below table is provided for referential purpose only.

DB SERVER	VERSION	JDBC DRIVER NAME	VERSION	DATE
MS SQL	2016	Sqljdbc41.jar, sqljdbc42.jar	6.0	02/27/2018

Table 5. JDBC Drivers

4.1 SQL Server (2016)

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:hsql:

Database URL Suffix: //(Computer Name):(Port)/(Name of Database)

4.3 MySQL – this database has never been tested with PHINMS; therefore it is completely up to user to experience or use it at their own risk.

JDBC Driver: com.mysql.jdbc.Driver

Database URL PreFix: jdbc:mysql:

Database URL Suffix: //(Computer Name):(Port)/(Name of Database)

4.4 Oracle – this database has not been tested with PHINMS; therefore, use is up to the discretion of the user.

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.

STATUS	DESCRIPTION
Success	Message send/receive operation successful.
Failure	Message send/receive operation failure.

Table 6. Status Codes

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

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 ,
          [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
```

6.1.3 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 ,
             [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
```

6.1.6 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 ,
responseArguments char (255) NULL ,
responseLocalFile char (255) NULL ,
responseFilename char (255) NULL ,
responseContent BLOB NULL ,
responseMessageOrigin char (255) NULL ,
responseMessageSignature char (255) NULL ,
priority number(10,0) 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
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
) ;

ALTER TABLE Sender_inq
ADD PRIMARY KEY
(recordId);
```

```
CREATE SEQUENCE Sender_inq_record_count
INCREMENT BY 1
START WITH 1
MINVALUE 1
MAXVALUE 999999999999999999999999999
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 ,
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 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)
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 ,
responseMessageId 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,  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)
);
```

6.3.3 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 ,   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)
);
```

6.3.8 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) 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 ,
            [sdnuser] [char] (255) NULL
) ON [PRIMARY]
GO

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 char (255)
NULL ,           payloadFile char (255) NULL ,
payloadContent LONGVARBINARY 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
LONGVARBINARY 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 ,
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 )
```