JMS Data Format Exchange Specification

1.0 Purpose of the document

The purpose of this document is to outline the JMS Data Format Exchange to be used as part of the JMS backend integration solution for the e-CODEX gateway.

2.0 MESSAGE FORMATS

The following section specifies the data format to be used to enable the following functions via JMS:

- Submit a message to the Gateway
- Push pending messages to a queue for retrieval by an ESB

It uses the JMS MapMessage type in order to implement the request and response data formats for each of the functions mentioned above. The Meta data in each case will be set in the JMS message properties using name/value pairs and these will be outlined in each case.

2.1 SUBMIT A MESSAGE

2.1.1 SUBMIT MESSAGE FORMAT

The following outlines the JMS Message format for submitting a message to the Gateway using the JMS interface.

JMS Message Type: MapMessage

Communication Properties

The following properties should be set using the setStringPropertyMethod of the MapMessage class.

messageType: submitMessage
Action
ConversationID
JMScorrelationID
Action
fromPartyID
fromRole
fromPartyType
toPartyID
toRole
toPartyType
originalSender
finalRecipient
Service
serviceType
protocol: AS4
ReftoMessageId
AgreementRef
EndPointAddress
totalNumberOfPayloads: number outlining the total number of payloads

Payload Properties

The following properties should be set for each payload in the message. [NUM] in each property name should be replaced with a numerical value representing each payload. The payload with the prefix payload-1 should be used for the bodyload of the message.

Each payload should be sent in byte format and set in the MapMessage using the setBytes method of the MapMessage class. Each payload should be identified by the property **payload-[NUM].**

The following properties should be set for each payload using the setStringProperty method of the MapMessage class:

payload-[NUM]-description: A description of the payload
payload-[NUM]-MimeContentID: For example the MimeContentID for the first payload will be identified by the property payload-1-MimeContentID.
payload-[NUM]-MimeType: The mime type of the payload

2.1.2 SUBMIT MESSAGE RESPONSE

Once the message has been successfully submitted to the Gateway a response will be placed on the outgoing queue. It will contain a message Id as well as a JMSCorrelation ID which matches the submitted message to allow the submitter of the message to reconcile their records.

JMS Message Type: MapMessage

<u>Properties</u>

JMScorrelationID: returns the correlation ID so that the backend can link message id to submitted messages

The following property should be set using the setStringProperty method of the MapMessage class

messageID: the ID allocated by the Gateway to the successfully submitted message

2.2 PUSH MESSAGE

The format of the message to be pushed to a queue for further processing by the ESB is detailed below.

2.2.1 FORMAT OF MESSAGE TO PUSHED

JMS Message Type: MapMessage

<u>Communication Properties</u>

The following properties should be set using the setStringPropertyMethod of the MapMessage class.

messageID: ID of the message as allocated by the Gateway
Action
ConversationID
JMScorrelationID
Action
fromPartyID
fromRole
fromPartyType
toPartyID
toRole
toPartyType
originalSender
finalRecipient
Service
serviceType
protocol: AS4
ReftoMessageId
AgreementRef
EndPointAddress
totalNumberOfPayloads: number outlining the total number of payloads

Payload Properties

The following properties should be set for each payload in the message. [NUM] in each property name should be replaced with a numerical value representing each payload. The payload with the prefix payload-1 should be used for the bodyload of the message.

Each payload should be sent in byte format and set in the MapMessage using the setBytes method of the MapMessage class. Each payload should be identified by the property **payload-[NUM].**

The following properties should be set for each payload using the setStringProperty method of the MapMessage class:

payload-[NUM]-description: A description of the payload
payload-[NUM]-MimeContentID: For example the MimeContentID for the first payload will be identified by the property payload-1-MimeContentID.
payload-[NUM]-MimeType: The mime type of the payload

3.0 EXAMPLE CLIENT

The following details a simple function for submitting a message in the correct format to a queue where it will be picked up by a MessageListener on the Gateway.

```
public static void sendMessageToDemoQueue() throws NamingException, JMSException {

    Properties env = new Properties();
    env.put(Context.INITIAL_CONTEXT_FACTORY, "net.timewalker.ffmq3.jndi.FFMQInitialContextFactory");
    env.put(Context.PROVIDER_URL, "tcp://localhost:10002");
    InitialContext jndi = new InitialContext(env);
    ConnectionFactory connectionFactory = null;
    Destination dest = null;
    connectionFactory = (ConnectionFactory) jndi.lookup("factory/ConnectionFactory");
    dest = (Destination) jndi.lookup("queue/inQueue");
    Connection connection = null;
    MessageProducer producer = null;
    connection = connectionFactory.createConnection();
    Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
    producer = session.createProducer(dest);
    producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);
    MapMessage messageMap = session.createMapMessage();


    // Set up the Communication properties for the message

    messageMap.setStringProperty("Service", "JMSTEST");
    messageMap.setStringProperty("Action", "UNRELIABLE_LOOPBACK_POSITIVE");
    messageMap.setStringProperty("ConversationID", "");
    messageMap.setStringProperty("fromPartyID", "GW1");
    messageMap.setStringProperty("fromPartyIDType", "urn:oasis:names:tc:ebcore:partyid-type:iso3166-1");
    messageMap.setStringProperty("fromRole", "GW");
    messageMap.setStringProperty("toPartyID", "GW1");
    messageMap.setStringProperty("toPartyIDType", "urn:oasis:names:tc:ebcore:partyid-type:iso3166-1");
    messageMap.setStringProperty("toRole", "GW");
    messageMap.setStringProperty("originalSender", "sending_court_id");
    messageMap.setStringProperty("finalRecipient", "receiving_court_id");
    messageMap.setStringProperty("serviceType", "");
    messageMap.setStringProperty("protocol", "AS4");
    messageMap.setStringProperty("refToMessageId", "");
    messageMap.setStringProperty("AgreementRef", "");
    messageMap.setStringProperty("endPointAddress", "");
    messageMap.setJMSCorrelationID("MESS1");

    //Set up the payload properties

    messageMap.setStringProperty("totalNumberOfPayloads", "3");

    messageMap.setStringProperty("payload-1-MimeContentID", "cid_of_payload_1");
    messageMap.setStringProperty("payload-2-MimeContentID", "cid_of_payload_1");
    messageMap.setStringProperty("payload-3-MimeContentID", "cid_of_payload_1");
    messageMap.setStringProperty("payload-1-MimeType", "application/xml");
    messageMap.setStringProperty("payload-2-MimeType", "application/xml");
    messageMap.setStringProperty("payload-3-MimeType", "application/xml");

    messageMap.setStringProperty("payload-1-description", "description1");
    messageMap.setStringProperty("payload-2-description", "description2");
    messageMap.setStringProperty("payload-3-description", "description3");

    String pay1 = "<XML><test></test></XML>";
    byte [] payload = pay1.getBytes();

    messageMap.setBytes("payload-1", payload);
    messageMap.setBytes("payload-2", payload);
    messageMap.setBytes("payload-3", payload);

    producer.send(messageMap);
    connection.close();
}
```