SAP Point of Sale 2.2
Centralized Electronic Funds Transfer 1.9
Technical Reference Guide

Technical Reference Guide

Copyright

© Copyright 2008 SAP AG. All rights reserved.

SAP Library document classification: PUBLIC

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML, and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. Transactionware, POS Xpress, Store Manager, and Configurator are all registered trademarks of SAP-Triversity. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves information purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP - Important Disclaimers

SAP Library document classification: PUBLIC

This document is for informational purposes only. Its content is subject to change without notice, and SAP does not warrant that it is error-free. SAP MAKES NO WARRANTIES, EXPRESS OR IMPLIED, OR OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Coding samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Internet hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint where to find supplementary documentation. SAP does not warrant the availability and correctness of such supplementary documentation or the ability to serve for a particular purpose. SAP shall not be liable for any damages caused by the use of such documentation unless such damages have been caused by SAP's gross negligence or willful misconduct.

Accessibility

The information contained in the SAP Library documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP specifically disclaims any liability with respect to this document and no contractual obligations or commitments are formed either directly or indirectly by this document. This document is for internal use only and may not be circulated or distributed outside your organization without SAP's prior written authorization.

Table of Contents

Chapter 6 About the Credit APM Adapter API

Chapter 7 Configuring the APM XML Files

1 Introduction to the APMs

This guide provides advanced technical reference information for users who plan to alter or augment their File Transfer or Centralized EFT system with additional custom Application Processing Module (APM), code, or functionality.

Note: This information is intended for an audience possessing a high level of technical literacy (such as software engineers and system administrators).

About the APMs

SAP Centralized EFT is a distributed transaction transport layer for your network. A highly configurable system, Centralized EFT allows any number of applications to attach to a simple programming interface and send messages to a transaction server. The main purpose of the transaction server is to receive a transaction from an application on one machine, deliver it to an APM, wait for the it to respond and return the response to the application. The APM might forward the transaction to an outside party for an authorization of some sort, or may simply perform a central database query. It can also be used to perform message exchange among connected applications, and to broadcast messages to any group of applications. New APM modules can be developed and interfaced into the system at any time offering virtually limitless use.

An application wishing to use Centralized EFT must be linked against the included Application Programming Interface (API). The application uses the subroutines in this API to connect to a Centralized EFT Server and send or receive transactions. The API is available as a C library or a set of Java Classes.

The Centralized EFT Server may be connected to any number of Application Processing Modules (APM). These programs may include a range of functionality from credit card authorization to shared database lookup. For a complete list of available APMs, see “About the individual APMs” on page 199.

As the need for new transaction processing tasks arise, new programs which perform the desired function may be written and interfaced into the Centralize EFT system.

2 Socket Server Interface

This section describes the connection interfaces.

About the Socket Server interface

- Socket Server will listen for connections at a particular port. The default port is 10000 (it is configurable)
- You can establish a socket connection to the Socket Server
- You will first need to send a sign on message and Socket Server will validate the request and either approve it or not
- If sign on was successful, you can then start sending request and receiving responses
- You do not need to wait for a response in order to send the next request
- You need to match any responses received with the related request sent
- It is your responsibility to maintain the socket connection

Sign on process

The sign on message is a XML document.

```
<SignOn>
   <Originator>
      <Chain> </Chain>
      <Store> </Store>
      <Register> </Register>
   </Originator>

   <ClientId> </ClientId>
</SignOn>
```

The chain, store, register and client id will need to be supplied in the sign on request. The Socket Server will validate the sign on request by checking to see if no client has already signed on and if the client can be connected to the system. If the validation is successful, it will send the following response:

```
<SignOn>
   <Originator>
      <Chain> </Chain>
      <Store> </Store>
      <Register> </Register>
   </Originator>
```

```
    <ClientId> </ClientId>

    <Result> OK </Result>
</SignOn>
```

And if the validation fails, then it will send the following response and close the socket.

```
<SignOn>
    <Originator>
        <Chain> </Chain>
        <Store> </Store>
        <Register> </Register>
    </Originator>

    <ClientId> </ClientId>

    <Result> Error message </Result>
</SignOn>
```

Clients should check if the sign on response was successful by checking the value of the `<Result>` element. If it reads OK, then the sign on response was successful. If not, then clients should consider that the sign on response was not successful. If clients need to sign on again, then clients will need to establish a new socket connection and repeat the sign on process.

Shutdown process

Clients will send a shut down message to the Socket Server, which will validate the request and send a response back to the client. If the validation was successful, the Socket Server will close the clients socket connection and if there are no more clients connected to it, will send the message to the system and then the Socket Server will terminate. The system on receiving the shut down message will terminate itself. The shutdown request message is a XML document.

```
<Shutdown>
    <Originator>
        <Chain> </Chain>
        <Store> </Store>
        <Register> </Register>
    </Originator>

    <ClientId> </ClientId>
</Shutdown>
```

Note: If the Chain, Store, Register and ClientId are all 0, then the Socket Server will send a shutdown message to the system irrespective of the number of clients connected to it.

The shutdown response message is a XML document as shown below

```
<Shutdown>
    <Originator>
        <Chain> </Chain>
        <Store> </Store>
        <Register> </Register>
    </Originator>

    <ClientId> </ClientId>

    <Result> OK </Result>
</Shutdown>
```

If shutdown failed, then the result will not be OK and will contain the reason for failure.

Message format

All request and response messages including the sign on message from/to the Socket Server will be of the following format:

1 Byte Length size (Binary)
Message Length (in bytes) (ASCII)
Message Data

Each message will first contain a 1-byte binary field, which specifies the number of bytes in the message length field. The message length field specifies the number of bytes in the message data field in ASCII. Message length should be right aligned and zero filled. The next field is the message data field.

For example:

- If the message is **<SignOn> </SignOn>**, the message length is 18 bytes and the size of the message length is 2 bytes. The following message should be sent through the socket:

  ```
  0x02 0x31 0x38 0x3C 0x53 0x69 0x67 0x6E 0x4F 0x6E 0x3E 0x20 0x3C 0x2F
  0x53 0x69 0x67 0x6E 0x4F 0x6E 0x3E
  ```

About the Sequential APM Connection interface

The SequentialConnection or InStoreConnection interface is an interface that supports any device and any protocol where messages are handled sequentially. Synchronous EFT service has a similar use but is designed to open a new connection, process a transaction and close the transaction when processing is complete. It does not allow the connection to know whether a new transaction is pending, so it can leave the connection open. The SequentialConnection interface gives the connection protocol complete control over when the device is opened, closed, read from and written to as well as time out conditions. It knows when a new transaction is pending and can leave the connection open to process it. It allows processing of transactions that have no unique identifier in order to match responses to requests. It is designed for use in the in store server as follows:

- Dial-up Connections using a Modem Device class and appropriate Protocol class
- Socket Connections using a Socket Device class and appropriate Protocol class

- Dial-up Connection and Socket Connection where the message formatting is identical for both. This would use the same APM, but a separate Device class and separate Protocol class for each connection type.
- Dial-up Connection and Socket Connection where the message formatting and protocol are identical for both. This would use the same APM, and the same Protocol class, but a separate device class for each connection.
- Dial-up connection and socket connection where the message formatting and protocol are identical for both. This would use the same APM, and the same Protocol class, but a separate device class for each connection
- Two separate APMs that use the same device and protocol can share the same device. In this case we can multi-trans across the APMs. By multi-trans, we mean that the first APM can dial-up, then both APMS can process a transaction before the modem hangs up

The SequentialConnection provides the generic IAPMConnection interface between the system and the connection. It abstracts the connection so any device with any protocol looks the same to the system. When SequentialConnection is instantiated, it in turn instantiates the Device Interface and Protocol Handler as specified in the configuration. The Device Interface presents a generic abstraction of either a socket connection or modem connection to the Protocol Handler. The Protocol Handler is a state machine that handles the specifics of the communication protocol independent of the device. The Protocol Handler maintains a queue of pending transactions for the purpose of handling the transactions sequentially and to match the response to the request simply by the order in which they are processed. The message identifier normally used to match responses to requests is optional. Each transaction has a running timer associated with it to limit the time spent in processing the overall transaction. In addition, the Protocol Handler has a running timer for the purpose of limiting the time in any given state of its state machine.

Transactions come into the SequentialConnection and are queued. Each time a transaction comes in, the protocol thread is triggered to wake it up. The protocol handler then gets the transaction, connects to the EFT provider if necessary and sends the request. The protocol then waits for the response, handling any protocol exchanges needed by the implementation. When the request comes in, it sets the response for the transaction and then goes to handle the next request or disconnects, depending on the protocol implementation. SequentialConnection then picks up the response and matches it in order with the request. All during this process more transactions may be queued for processing, but the transactions are handled sequentially.

Detailed description of the SequentialConnection class

The SequentialConnection class interfaces to the system by implementing the APM model using CreditService as the APM class and SequentialConnection as the Connection Class and a Handler class extended from BaseCreditServiceHandler. (See Dial-up and Socket Configuration below). The interface that SequentialConnection implements consists of the following methods from IAPMConnection:

- void openConnection(String connectionName) throws IOException— This method makes subsequent calls to open the device and protocol. The connectionName is currently not used but should be used to set the connection name for Single Connection – Multiple APM configurations (covered later). This is called on startup of an instance of the APM and should handle any initialization.
- void openConnection(String connectionName, int timeout) throws IOException— This method is currently not implemented but should be the same as above except a time limit is place on completing the open.
- void close() throws IOException— makes subsequent calls to close the device and protocol.

- DataInputStream getInputStream() throws IOException— This method is here to satisfy the IAPMConnection interface and is not intended to be used for this interface.
- DataOutputStream getOutputStream() throws IOException— This method is not intended to be used for this interface.
- void write(byte[] data) throws IOException— This method is not intended to be used for this interface
- void writeObject(Object objectOut) throws IOException— This method is not intended to be used for this interface
- void writePersistent(TNObject tnObject) throws IOException— This method is used to send a request to the protocol to queue it for processing. Since transactions are queued, additional requests may be sent to the protocol before the response to the first request comes back. The responses come back in the same order as the requests were sent.
- byte[] read() throws IOException— This method is not intended to be used for this interface.
- boolean read(byte[] data) throws IOException— This method is not intended to be used for this interface.
- Object readObject() throws IOException— This method returns the response if one is available. Since this interface is sequential, responses are returned in the order that the requests are sent out.
- Object readObject(long timeoutMillis) throws IOException— This method is the same as above except it waits for the specified number of milliseconds before returning. This is the method called by CreditService to poll for a response.
- boolean readPersistent(TNObject tnObject) throws IOException— This method is not intended to be used in this interface
- boolean setConfiguration(Object cfg)— This method is called during startup of this connection instance. It instantiates and configures the specified device and protocol objects.

The SequentialConnection interface divides the connection into two parts.

- The device (such as modem or socket)
- The protocol (such as TCPIP, Visanet and variations)

Detailed description of the Device Class

This device abstracts the physical device into an interface to the Protocol Handler. It is instantiated and configured on startup by SequentialConnection. SequentialConnection passes a reference to the device object to the protocol to provide the interface. The interface consists of the following methods:

- void setAppName(String name) – This is currently not used, but is intended for use where there are named connections that can be used by multiple APMs. Each APM that needs that specify a given connection name will actually use the same instance of the given SequentialConnection (see Single Connection Multiple APM section).
- void setProtocol(AbstractProtocol protocol) – This is called by SequentialConnection to give the device access to the protocol methods needed by the device. For example protocol.timedOut() indicates that the protocol timer has expired and the device must take corrective action when reading from or writing to the device
- boolean setConfiguration(Element cfg) – This is called by the SequentialConnection to allow setting any parameters needed by the implemented device
- void open(connectionName) throws IOException – This is called by the SequentialConnection during startup to do any initialization needed in the instance of the device

- void close() – This method is by SequentialConnection.close which is called by the system to close the connection. Since the connection is controlled by the protocol, this method should not do anything.
- boolean connect() – This method is available to the Protocol handler to ensure a connection to the EFT provider. It should first check if the connection is established and if not, then should make a connection. It returns true if the connection is good and false if it could not make a connection.
- void disconnect() – This method is available to the Protocol handler to remove a connection to the EFT provider.
- int readNextByte(Trx trx) – This method reads the next byte from the connection to the EFT provider if one is available. It should check to see if the connection is still active and whether a timeout occurred when reading each byte. It returns the next byte as an integer. It returns an error code on other conditions such as NOCHAR, TRX_TIMEOUT, PROTOCOL_TIMEOUT, or LINEDROP.
- int readCountBytes(byte[] byteArray, int offset, int count) – This method waits until there are "count" bytes in the input stream then fills "byteArray" beginning at "offset" with "count" bytes. It returns the number of bytes read in. If there is no connection or there was an Exception thrown it returns 0 bytes read.
- byte[] readCountBytes(int count) – This method checks to see if there are "count" bytes in the input stream and if there is it returns a byte array filled with "count" bytes". If "count" bytes are not available, it returns null, it does not wait for input.
- boolean transmit(int aByte) – This method writes a byte to the output stream. It returns true if the write was successful, false if an error occurred.
- boolean transmit(byte[] byteArray) – This method writes a byte array to the output stream. It returns true if the write was successful, false if an error occurred.

Detailed description of the Protocol Class

This is the workhorse class of the modem connection, it has complete control over connecting, disconnecting, writing requests and reading responses between this interface and the EFT provider. It is and extension of Java's Thread class and is instantiated, configured and started from the SequentialConnection. Requests come from SequentialConnection to the protocol through the method sendRequest which wraps the request in a protocol specific packet, then queues the transaction in TrxQueue. Responses are received by ModemProtocol through the method nextResponse which just returns the next available response (no identifier is needed for matching response to request). If an identifier is present, it is pre-pended to the response byte array for parsing by the APMs response parser. The runProtocol method is the protocol's threads run method. It is a continuously running state machine that keeps track of the state of the connection and using a Trx object which holds the state of each transaction. It decides when and how to get the next available request, connect to the EFT provider, send request, receive the next response and disconnect from the EFT provider since these are all determined by the EFT providers specified protocol. The following methods are available in the Protocol class.

- boolean setConfiguration(Element cfg) – This method is called by SequentialConnection.setConfiguration to load any configuration information needed by this implemented protocol handler.
- void setDevice(AbstractDevice) – This method is called by SequentialConnection to give this protocol handler a reference to the configured device
- void open(String connectionName) throws IOException – This method is called by SequentialConnection.open and handles any initialization not handled by setConfiguration.

- void close() – This method is called by SequentialConnection.close and cleans up anything needed when the connection is destroyed. i.e. when shutting down
- boolean sendRequest(byte[] baRequest, String identifier) – This method is called by SequentialConnection to queue an incoming request for processing. If queueing is successful, then the protocol thread is triggered to wake it up. It returns true if the queuing the request was successful and false if not. The identifier is saved by the trx object for prepending on the response.
- byte[] nextResponse() – This method is called by SequentialConnection to get the next response if one is available. The identifier optionally passed in on the sendRequest method is prepended to the response byte array for parsing by the APMs response parser.
- boolean timedOut() – This method returns the status of the protocol timer. It returns true if the protocol timer has timed out or false if the timer has not been set or has not timed out yet.
- void runProtocol() – This method is the protocol threads run method. It is the protocol state machine that executes through the protocol states.

Detailed description of the TrxQueue class

The TrxQueue class is designed to hold transactions (Trx) for the protocol to pick them up and process them. It is also used to match requests to responses by the order in which they occur. The third use is to control the number of times a request can be sent (if the send fails) and the number of times it can retry to get a response (NAKing a bad response). Since some of these APMs do not have a valid echo field (a field that responses with the same info as the request), matching must be done by the order in which they occur.

The following methods are available in the TrxQueue class:

- TrxQueue(int size, int resends, int naks, int timeout) - This constructor is called during initialization of the protocol. The size is the number of elements for the queue. The resends sets the number of times a request can be resent. The naks sets the number of times a response can be NAK'd. The timeout sets the timeout for the transaction as a whole (MaxTrxTime in the configuration). This allows these values to be firm configurable for a given implementation, not through configuration but through subclassing.
- boolean setNextRequest(byte[] baRequest) – This method places the passed request on the queue in the next slot if one is available. It returns true if successfully queued and false if the queue is full.
- boolean setNextRequest(byte[] baRequest, String identifier) – This method is the same as above except it also sets the transaction identifier.
- Trx getNextRequest() – This method is called by the protocol to obtain a reference to the next transaction if one is available. It returns null if a new request is not available
- byte[] nextResponse() – This method is called by the protocol to get the nextResponse if one is available. The optional identifier is prepended to the message if available.

Detailed description of the Trx class

The Trx class holds the state of each transaction for use by the device and protocol for decision making. It contains the following methods:

- Trx(int resends, int naks, int timeout) – This constructor sets the limits on resends, naks and the transaction timeout (see TrxQueue for explanation)
- String getIdentifier() – This method returns the identifier for this transaction

- boolean setRequest(byte[] baRequest) – This method sets the request byte array for this transaction and kicks off the transaction timer. It returns true on success, false if this Trx already has a request.
- boolean setRequest(byte[] baRequest, String identifier) – same as above except it also sets the identifier for this transaction.
- boolean isNextRequest() – This method returns true if it holds the next transaction to be processed, false if is empty or being processed.
- byte[] getRequest() – This method returns the request byte array for this transaction.
- boolean testAndIncrementResends() – This method may be called by the protocol if it needs to resend a request. It returns true if resends is below maximum and false if the limit has been reached. Each time it is called, it increments the resends for this transaction.
- boolean testAndIncrementNaks() – This method may be called by the protocol if it needs to NAK a response and wait for the next response. If returns true if naks is below maximum and false if the limit has been reached.
- boolean timedOut() – This method should be called by the protocol to determine if the transaction has timeout out (MaxTrxTime in Configuration).
- boolean setResponse(byte[] baResponse) – This method is called by the protocol to set the response for this transaction. It returns true if successful, false if there was already a response set.
- boolean setResponse(String strResponse) – This method is the same as above except it is passed a String instead of a byte array.
- byte[] getResponse() – This method returns the response byte array for this transaction if available. The transaction identifier is prepended to the byte array response.
- int getResendCount() – This method may be called by the protocol to get the resend count for this transaction. The resend count is incremented by testAndIncrementResends() above.
- void setResendCount(int aResendCount) – This method may be called by the protocol to override the resend count.
- int getNakCount() – This method may be called by the protocol to get the nak count. The nak count is incremented by testAndIncrementNaks() above.
- void setNakCount(int aNakCount) – This method may be called by the protocol to override the nak count.

Configuring modem and socket connection

The configuration of both the Modem and Socket Connections uses Credit Service. Multiple phone numbers and socket addresses may be used as shown below. Note that the retry is actually number of tries not retries. There are many timeout values and the relationship between them is as follows:

- The TransactionLifeSpan in the Connection branch is the overall timeout of the transaction. It should be lower than the POS timeout value.
- On SequentialConnections— The Route timeout (timeoutSeconds) must be less than the TransactionLifespan and maxTrxTime must be less than the route timeout. maxTrxTime defaults to 55 seconds
- On SocketConnections with fail-over to modem— The Route timeout (timeoutSeconds) must be less than the TransactionLifespan but must be 1 to 4 seconds greater than the total connectTime for all destination socket addresses. maxTrxTime defaults to 35 seconds.

The following is a sample configuration.

```
<Route name="FDMSSocketRoute" connectionName="FDMSSocketAPM"/>
<Route name="FDMSModemRoute" connectionName="FDMSModemAPM"/>
```

```
<RoutingRule>
       <Route=”FDMSSocketRoute” timeoutSeconds=”3”/>
      <Route=”FDMSModemRoute” timeoutSeconds=”95”/>
</RoutingRule>
```

Dial-up configuration

```
<Connection name="FDMSModemAPM"
    class="com.triversity.transnet.core.apm.APMConnectionHandler">
    <APM name="FDMS" maximumCount="1" minimumCount="1"
        class="com.triversity.transnet.core.credit.CreditService"

connectionClass="com.triversity.transnet.xtension.credit.modem.ModemConne
ction"

CreditServiceHandlerClassName="com.triversity.transnet.xtension.credit.FD
MS.FDMSHandler"
        TransactionLifeSpan="120000">
       <AllowdIdleTime value="1" unit="hour" MerchantsFile="FDMSMerchant"
/>
        <Connection id="1" Name="Conn1">
            <Device
class="com.triversity.transnet.xtension.credit.modem.ModemDevice">
                <PhoneNumber retry="1">9,18006837986</PhoneNumber>
                <PhoneNumber retry="1">9,18006837986</PhoneNumber>
                <PhoneNumber retry="1">9,18006837986</PhoneNumber>
            </Device>
     <Protocol MaxTrxTime = “90”
class="com.triversity.transnet.xtension.credit.modem.ModemProtocol" />
        </Connection>
    </APM>
</Connection>
```

Socket configuration

```
<Connection name="FDMSSocketAPM"
    class="com.triversity.transnet.core.apm.APMConnectionHandler">
    <APM name="FDMS" maximumCount="1" minimumCount="1"
        class="com.triversity.transnet.core.credit.CreditService"

connectionClass="com.triversity.transnet.xtension.credit.modem.ModemConne
ction"

CreditServiceHandlerClassName="com.triversity.transnet.xtension.credit.FD
MS.FDMSHandler"
        TransactionLifeSpan="120000">
       <AllowdIdleTime value="1" unit="hour" MerchantsFile="FDMSMerchant"
/>
        <Connection id="1" Name="Conn1">
            <Device
class="com.triversity.transnet.xtension.credit.modem.SocketDevice">
                <Destination host="10.140.10.50" port="17916"
connectTimeout="1000" />
                <Destination host="10.140.10.51" port="17916"
connectTimeout="1000" />
```

```
            </Device>
            <Protocol
class="com.triversity.transnet.xtension.credit.modem.SocketProtocol" />
        </Connection>
    </APM>
</Connection>
```

3 About the JMS Interface

This section contains information on configuring the Java Messaging Service, or JMS, interface.

Configuring the JMS interface

To configure the Java Messaging Service, or JMS, interface, you need to create two JMS destinations in the JMS server: one for sending messages to the system and one for receiving messages from the system. The destinations may be queue or topic. The destinations can be of different types. The destination names for sending messages to the system is ToTransnet and for receiving from the system is FromTransnet as shown in the two configuration samples listed under the section entitled "Configuring Transnet.xml file" on page 21. This can be changed later, but you must ensure that you update any required configurations with the change.

JMS message types

Only JMS TextMessage type messages containing XML formatted strings can be sent to the system via JMS. You will need to create an XML message for the type of request, convert it to a string and set that as the body of a TextMessage. Before posting to JMS, you will need to set the following boolean properties in the message.

Name	Explanation
TN_Response	Set to true if sending a request Set to false if sending a response
TN_ResponseExpected	Set to true if expecting a response Set to false if not expecting a response

Matching requests to responses

The system sets the JMS Correlation ID field of the response with the Message ID field of the request. This could be used for matching requests and responses. The system also echoes back other properties in the request message with the restriction that these properties should start with something other than JMS or TN. The system will also expect the response to a request sent by it to have the Message ID of the request in the Correlation ID field of the response.

Configuring Transnet.xml file

To configure the system, a connection entry needs to be added to the configuration XML file (usually Transnet.xml). In the configuration the *class* property is defined by the development group and should

not be changed. In the JMSContext configuration, the properties *contextBroker* and *urlPackage* are dependent on the type of JMS server used. For the two JMS servers, the values provided in the following configurations must be used. The properties *topicBroker* and *queueBroker* must match the appropriate JNDI name set for them with the JMS server.

The following are examples of connection configurations for JBOSS MQ and WASMQ.

Connection to JMS implementation of JBOSS

```
<Connection name="JBOSS"
class="com.triversity.transnet.core.tms.jms.TNJMSClientHandler"
connectionID="114"
sendTo="FromTransnet"
receiveFrom="ToTransnet"
durableName="">
   <JMSContext name="JBOSSMQ"
   contextBroker="org.jnp.interfaces.NamingContextFactory"
   providerURL="jnp://127.0.0.1:1099"
   urlPackage="org.jboss.naming"
   topicBroker="ConnectionFactory"
   queueBroker="ConnectionFactory"
   username="" password="">
   <jndiProperty name="java.naming.rmi.security.manager" value="no"/>
   </JMSContext>
</Connection>
```

Connection to JMS implementation of WAS MQ

```
<Connection name="IBM_MQ_EFT"
class="com.triversity.transnet.core.tms.jms.TNJMSClientHandler"
connectionID="114"
sendTo="FromTransnet"
receiveFrom="ToTransnet"
durableName="">
   <JMSContext name="IBM_MQ"
   contextBroker="com.ibm.websphere.naming.WsnInitialContextFactory"
   providerURL="iiop://localhost:9810"
   urlPackage=""
   topicBroker="jms/TopicConnFactory"
   queueBroker="jms/QeueConnFactory"
   username=""
   password="">
   </JMSContext>
</Connection>
```

Required library files

While invoking the application, the class files for communication with the JMS provider must be available in the classpath.

For JBOSS, the following files in the JBOSS home directory must be in the classpath:

```
client\jbossmq-client.jar
client\jnp-client.jar
client\jboss-j2ee.jar
client\oswego-concurrent.jar
```

For WAS MQ, the following files must be in the classpath:

```
MQ_HOME/java/lib
MQ_HOME/java/lib/com.ibm.mq.jar
MQ_HOME/java/lib/com.ibm.mqjms.jar
MQ_HOME/java/lib/com.ibm.mqbind.jar
MQ_HOME/java/lib/connector.jar
MQ_HOME/java/lib/ldap.jar
MQ_HOME/java/lib/jndi.jar
MQ_HOME/java/lib/fscontext.jar
MQ_HOME/java/lib/providerutil.jar
```

If you are also using the WAS Application Server, you will need to add the following to the classpath as well:

```
WAS_HOME/properties
WAS_HIME/lib/naming.jar
WAS_HIME/lib/ecutils.jar
WAS_HIME/lib/ffdc.jar
WAS_HIME/lib/idl.jar
WAS_HIME/lib/iwsorb.jar
WAS_HIME/lib/j2ee.jar
WAS_HIME/lib/namingclient.jar
WAS_HIME/lib/ras.jar
WAS_HIME/lib/sas.jar
WAS_HIME/lib/tx.jar
WAS_HIME/lib/txPrivate.jar
WAS_HIME/lib/utils.jar
WAS_HIME/lib/wsexception.jar
WAS_HIME/lib/messagingImpl.jar
```

4 Financial XML Message Specification

This section defines the XML messages that can be sent from the SAP POS solutions (Transactionware GM and Transactionware *Enterprise*) to the Centralized EFT. Note that this document does not describe the envelope, context, or transport protocols used to exchange these messages.

About the Financial XML message specification

This specification represents a translation from the internal representation of the ISO8583 standard that has been implemented in SAP solutions into XML. We have found the need to extend the structure to include information not explicitly defined in the ISO specification based on the delivery of numerous integration projects with major American financial service providers and Canadian banks over the last two years. This XML specification reflects those extensions.

Functions and use cases

The following functions and use cases are covered by this specification:

- Authorization (Sale/Purchase/Redeem)
- PreAuthorization
- PostAuthorization
- Return
- Activate
- Payment (Reload)
- BalanceInquiry
- Replace
- CashOut
- History (Not yet supported)
- Apply
- Reconciliation
- Settlement
- Deactivate
- Sign On
- PinPad Key Exchange
- PinPad Close Batch

- Transfer

Function modifiers

The following function modifiers are included in this specification:

- Original
- OriginalWithAddressVerification
- Void
- VoidOriginalWithAddressVerification
- Reversal
- ReversalOriginalWithAddressVerification

Financial instruments/account types

The following financial instrument/account types are included in this specification. If account types are not required in the specification, AccountType should be set to 'None'.

- CreditCard
- DebitCard
- PrivateLabelCard
- StoredValueCard
- GiftCard
- Cheque
- EBT

Account names

The following are valid account names.

- Default
- Saving
- Checking
- Cash
- Food Stamp

Common aggregates

All the elements in an aggregate must always be present. Wherever an aggregate name appears in the document, it should be replaced with the aggregate elements. Aggregate names will be enclosed in curly braces (that is, { }).

For example:

```
<RequestedAmount> {Amount Aggregate} </RequestedAmount>
```

would mean…

```
<RequestedAmount Value="" CurrencyCode="" > </RequestedAmount>
```

1. Name

```
<Name>
```

```
    <Prefix />
    <Last />
    <First />
    <Middle />
    <Suffix />
</Name>
```

Note: If not using a name element set it as an empty string.

2. Phone

```
<Phone>
    <CountryCode></CountryCode>
    <AreaCode></AreaCode>
    <ExchangeCode></ExchangeCode>
    <LineNum></LineNum>
    <Extension></Extension>
</Phone>
```

Note: If not using a phone element set it to 0.

3. Address

```
<Address>
    <StreetNumber />
    <StreetName />
    <UnitNumber />
    <City />
    <State />
    <Country />
    <PostalCode />
</Address>
```

Note: If not using an address element set it as an empty string.

4. Person

```
<Person>
    <Id></Id>
    <Name> {Name Attribute} </Name>
    <Male>True / False</Male>
    <WorkPhone> {Phone Attribute} </WorkPhone>
    <HomePhone> {Phone Attribute} </HomePhone>
    <MobilePhone> {Phone Attribute} </MobilePhone>
    <EmailAddress />
    <HomeAddress> {Address Attribute} </HomeAddress>
    <DateOfBirth> </DateOfBirth>
</Person>
```

Note: As of now, all elements must be provided. If not using the id element, set it to 0. If not using EmailAddress element, set it as an empty string. If not using the DateOfBirth element, set it as “0000”. See the Date data type.

5. Amount

```
<Amount  Value = “” CurrencyCode =”” />
```

6. Account Balance Amount

```
<AccountBalanceAmount AccountName=”” Value = “” CurrencyCode =”” />
```

7. Total Amount

```
<TotalAmount  Count=””  Value = “”  CurrencyCode =”” />
```

Common elements

The following are the common elements:

Message type

```
<MessageType>
   <AccountType> </AccountType>
   <Function> </Function>
   <Modifier> </Modifier>
   <StoreAndForward> True / False <StoreAndForward>
</MessageType>
```

Verification method

```
<VerificationMethod Type=””>
   <Number> </Number>
   <State> </State> [Optional]
   <DateOfBirth> </DateOfBirth> [Optional]
   <ZipCode> </ZipCode> [Optional]
</VerificationMethod
```

The following are the valid types:

- DriversLicense
- SSN
- PlasticCard
- MICR

PinPad Information

```
<PinPadInfo>
   <SequenceNumber> </SequenceNumber>   [Optional]
   <TransmissionNumber> </TransmissionNumber>   [Optional]
   <MACData> </MACData>   [Optional]

   <FormattedData> </FormattedData>   [Optional]

   <RawData> </RawData> [Optional]
```

```
    <DisplayInfo> </DisplayInfo>  [Optional]

    <PinData> </PinData>  [Optional]
    SecurityControlInfo> </SecurityControlInfo>  [Optional]

    <PinEncryptionKey> </PinEncryptionKey>  [Optional]
    <MacEncryptionKey> </MacEncryptionKey>  [Optional]
    <FieldEncryptionKey> </FieldEncryptionKey>  [Optional]

</PinPadInfo>
```

Note: Notes:RawData and FormattedData must contain data represented in hex format

Merchant Information

```
<MerchantInfo>
    <MerchantID> </MerchantID> [Optional]

    <CreditName> </CreditName> [Optional]
    <DebitName> </DebitName> [Optional]

    <TerminalID> <TerminalID> [Optional]

    <PinPadKeyExchangeRequired> </PinPadKeyExchangeRequired> [Future]
    <PinPadFormattedDataExchangeRequired>
    </PinPadFormattedDataExchangeRequired> [Future]
</MerchantInfo>
```

Note: The signOn response will contain the MerchantID, CreditName and DebitName and all other responses might optionally contain the MerchantID and TerminalID but will not contain the CreditName and DebitName.

Card detail

```
<CardDetail Name="">
    <InputMode> Keyed / Swiped / MICR </InputMode>
    <CardPresent> true / false </CardPresent>

    <AccountNumber></AccountNumber>

    <ChequeNumber> </ChequeNumber> [Optional]
    <BankNumber> </BankNumber> [Optional]
    <MICRNumber> </MICRNumber> [Optional]

    <VerificationMethod Type=""> </VerificationMethod> [Optional]

    <ExpiryDate > </ExpiryDate>
    <Track2></Track2> [Optional]
    <EffectiveDate> </EffectiveDate> [Optional]
```

```
    <CardHolder> {Person Aggregate} </CardHolder> [Optional]

    <CardHolderPresent> True / False </CardHolderPresent> [Optional]
    <CardHolderAuthenticationMethod /> [Optional]
    <CardHolderAuthenticationEntity /> [Optional]
    <OperatingEnvironment> </OperatingEnvironment> [Optional]
</CardDetail>
```

Note:

- The Name attribute is optional. The following are the valid "Name" values: New and Old
- All the elements not declared as optional must be present. Account number and expiry date must always be present even if track2 data is present.
- If the optional fields are not present, then the following default values will be assumed:
  - EffectiveDate will be assumed as today's date
  - CardHolderPresent will be assumed as true
  - CardHolderAuthenticationMethod will be assumed as 6
  - CardHolderAuthenticationEntity will be assumed as 0
  - OperatingEnvironment will be assumed as 0
- CardHolderAuthenticationMethod, CardHolderAuthenticationEntity and OperatingEnvironment are enumerated types. Please refer to the section on Enumerated Data types for valid values.

Tender total

```
<TenderTotal Desc="">
    <NetAmount> {Amount Aggregate} </NetAmount>

    <Authorization>
        <Original> {TotalAmount Aggregate} </Original>
        <Void> {TotalAmount Aggregate} </Void>
        <Reversal> {TotalAmount Aggregate} </Reversal>
    </Authorization>
    <Return>
        <Original> {TotalAmount Aggregate} </Original>
        <Void> {TotalAmount Aggregate}</Void>
        <Reversal {TotalAmount Aggregate} </Reversal>
    </Return>
    <Payment>
        <Original> {TotalAmount Aggregate} </Original>
        <Void> {TotalAmount Aggregate} </Void>
        <Reversal> {TotalAmount Aggregate} </Reversal>
    </Payment>
    <CashOut>
        <Original> {TotalAmount Aggregate} </Original>
        <Void> {TotalAmount Aggregate} </Void>
        <Reversal> {TotalAmount Aggregate} </Reversal>
    </CashOut>
</TenderTotal>
```

The description is the tender description. For example, the Visa tender total could read as follows:

```
<TenderTotal Desc="VISA"> </TenderTotal>
```

And if the totals represent all the tenders:

```
<TenderTotal Desc="ALL"> </TenderTotal>
```

The following are the valid tender descriptions:

- VISA
- MASTERCARD
- AMEX
- DISCOVER
- JCB
- DINERSCLUB
- CARTEBLANCHE
- PRIVATELABEL
- BVISA
- BMASTERCARD
- ALL
- DEBIT
- CHEQUE

Note: The Authorization tender total also includes post authorizations.

Data types

Binary

All binary fields must be converted to hex. For example if the data is the following:

- ASCII DATA = PIN
- BINARY DATA = 01010000 01001001 01001110

then the following hex data must be set (without any spaces):

- HEX DATA = 05 00 04 09 04 0E

Boolean

True / False or true / false – case insensitive

Date

- It uses an OFX date standard.
- YYYYMMDDHHMMSS.XXX *[gmt offset:tz name]*

  For example, "19961005132200.124[-5:EST]" represents October 5, 1996, at 1:22 and 124 milliseconds p.m., in Eastern Standard Time. This is the same as 6:22 p.m. Greenwich Mean Time (GMT).

Date accept values with fields omitted from the right. They assume the following defaults if a field is missing:

- Specified date or datetime assumed defaults
- YYYYMMDD - 12:00 AM (the start of the day), GMT
- YYYYMMDDHHMMSS - GMT
- YYYYMMDDHHMMSS.XXX - GMT

Note: Times zones are specified by an offset and optionally, a time zone name. The offset defines the time zone. Valid offset values are in the range from –12 to +12 for whole number offsets. Formatting is +12.00 to -12.00 for fractional offsets, plus sign may be omitted.

If the date value is set to "0000", then the date should be considered as undefined or a null value.

Amount

Decimal Number with / without decimal point. For example: 10.00

Enumerated types

For the following enumerated types, only the number should be provided.

Card Holder Authentication Methods

- 0 –Not authenticated
- 1 – PIN
- 2 – Electronic Signature Analysis
- 3 – Biometrics
- 4 – Biographic
- 5 – Manual Signature Verification
- 6 – Other Manual Verification

Card Holder Authentication Entity

- 0 – Not authenticated
- 1 – ICC
- 2 – CAD
- 3- Authorizing Agent (Identified by a code)
- 4 – By Merchant
- 5 – Other

Operating Environment

- 0 –No Terminal Used
- 1 – On premises of card acceptor, attended
- 2 - On premises of card acceptor, unattended
- 3 - Off premises of card acceptor, attended
- 4 - Off premises of card acceptor, unattended

- 5 - On premises of card holder, unattended

ISO Message specifications

This document is proprietary and confidential, and intended for the internal use of SAP.

This document defines the ISO 8583 v1993 messages that can be sent from the SAP POS solutions (Transactionware and Transactionware Enterprise) through Centralized EFT. This document does not describe the envelope, context, or transport protocols used to exchange these messages.

This is the first draft of this document, and as such may contain errors and omissions that will be corrected in future versions.

Message ID

Message Type	**Message ID** Online/SAF
Reversal	1420
Debit/Check/EBT	1200/1220
Other card types	1100/1120

Function code

Message Type	**Function Code**
Reversal	
SAF Reversal	441
Online Reversal	400
Others	
Debit/Check/EBT	200
Pre Auth	101
Post Auth	102
Others	100

Account type

Account Type	**Value**
Default	00

Account Type	Value
Private Label Card	90
Stored Value Card	91
Gift Card	92
Debit Card	
Checking (Default)	20
Saving	10
EBT	
Cash	93
FoodStamp	94

Amount type

Amount Type	Value
Ledger Balance	01
Available Balance	02
Amount Owing	03
Amount Due	04
Available Credit	05
Amount Remaining	20
Cash	40
Goods And Services	41

Processing code

Message Type	Processing Code
Default	00
Check	24
Pre Auth	00
Auth Reversal/Void	22
Auth Online	00

Refund Reversal/Void	02
Refund Online	20
Payment Reversal/Void	58
Payment Online	50
Activate Reversal/Void	91
Activate Online	90
DeActivate Reversal/Void	93
DeActivate Online	92
CashOut Reversal/Void	28
CashOut Online	01
Replacement Reversal/Void	48
Replacement Online	40
Balance Inquiry	31

Note: From Account = To Account = Account Type

Message reason codes

Reason Code	**Reason Description**
1005	Above Floor Limit (Phone Auth)
1006	Below Floor Limit

Fields

Bit #	**Field**	**Comment**
2	Primary Account Number	
4	Amount	
11	Transaction Number	
12	Transaction Date Time	
14	Expiry Date	yyMM
22	POS Data Code	
25	Offline Reason	See section 2.6 for valid message reason values

26	Acceptor Business Code	Always set to 3355
32	Acquirer Institution Id	Always set to TNClient
35	Track 2	
37	Sequence Number	
38	Approval Code	
39	Action Code	See section 2.10 for valid action codes
41	Terminal ID	
42	Merchant ID	
43	Acceptor Name	Always set to TNClient
47	Pin Pad Display Info	
49	Currency Code	
53	Security Control Info	
56	Original Data Elements	Format: <Orig Message ID><Orig Txn Number> <Orig Txn DateTime><Orig Acq. Inst. Id Length><Orig Acq. Inst. Id> Where <Orig Message ID> - original message id (size 4) <Orig Txn Number> - original txn number (size 6) <Orig Txn DateTime> - original txn date time (yyMMddHHmmss) <Orig Acq. Inst. Id Length> - Original acquiring institution id code length (size 2). <Orig Acq. Inst. Id> - Original acquiring institution id code. Always set to TNClient.
62	Register details	Format <Length><Chain><Store><Register> Where <Length> is fixed length of size 3 <Chain> is fixed length of size 3 <Store> is fixed length of size 5 <Register> is fixed length of size 3
54	Additional Amount	Each additional amount will include Account Type, Amount Type, Currency Code, Amount See section 2.8 for various additional amount fields

63	Additional Fields	Will contain fields in the format <Field Name><Field Value Length><Field Value> Where <Field Name> is fixed length of size 4 <Field Value Length> is fixed length of size 3 <Field Value> can be maximum of size 999. See section 2.9 for Additional field names.
64	MAC Data	
102	Second Card Details	Used for Replacement function. Format: <Account Number>=<Expiry Date> <Expiry Date> is in yyMM format
111	Raw Data	

Additional amount fields

Additional Amount Field	**Account Type**	**Amount Type**
Account Available Balance	*See sec 2.3*	02
Cash Back Amount	00	40
EBT Cash Balance	93	02
EBT Food Stamp Balance	94	02

Additional fields

```
CUSTOMER_NAME = "CNAM";
CUSTOMER_PRIMARY_ADDRESS = "CPAD";
CUSTOMER_SECONDARY_ADDRESS = "CSAD";
CUSTOMER_CITY = "CCIT";
CUSTOMER_STATE_CODE = "CSTA";
CUSTOMER_ZIP_CODE = "CZIP";
CUSTOMER_COUNTRY = "CNTR";
DETAIL_RESPONSE = "RCPT";
CREDIT_PLAN = "PLAN";
PRODUCT_CODES = "PROD";
PIN_DATA = "EPIN";
ORIGINAL_TXN_DATETIME = "OTDT";
ORIGINAL_TXN_NUMBER = "OTNU";
ORIGINAL_SEQ_NUMBER = "OSEQ";
DOCUMENT_REFERENCE_NUMBER = "DORN";
SETTLEMENT_DATA = "SETL";
```

```
EMPLOYEE_ID = "EMPL";
HOST_SEQUENCE_NUMBER = "HSEQ";
LANGUAGE_CODE = "LANG";
DISPLAY_INFO = "DISP";
HOST_REFERENCE_NUMBER = "HREF";
APPROVAL_DATETIME = "APDT";
PIN_LENGTH = "PINL";
BANK_NUMBER = "BNUM";
CHEQUE_NUMBER = "CHQN";
MICR_NUMBER = "MICR";
PIN_PAD_SEQUENCE_NUMBER = "SEQN";
PIN_PAD_TRANSMISSION_NUMBER = "TNUM";
PIN_PAD_REQUEST = "PPRQ";
PIN_PAD_RESPONSE = "PPRS";
TERMINAL_CREDIT_NAME = "TRMC";
TERMINAL_DEBIT_NAME = "TRMD";
MERCHANT_INFO = "TRMN";
MERCHANT_NAME = "MRCH";
PIN_ENCRYPTION_KEY = "PINK";
MAC_ENCRYPTION_KEY = "AUTK";
FIELD_ENCRYPTION_KEY = "FDCK";
EXTERNAL_RESPONSE_CODE = "BRSP";
DRIVERS_LICENSE_NUMBER = "DRNO";
DRIVERS_LICENSE_STATE = "DRST";
ECA_CHECK = "ECAC";
SSN = "SSNO";
```

Fields for check tenders

Bit #	**Field**	**Comment**
2	SSN	Social Security Number
4	Amount	Over tender amount/cash back amount/cash check amount
49	Currency Code	Amount’s currency code.

Action codes

Note: This reference list was taken from the ISO8583-1993 specification.

```
RC_APPROVED= 000; // approved
RC_APPROVED_WITH_ID= 001; // honour with identification
RC_APPROVED_PARTIAL= 002; // approved for partial amount
RC_APPROVED_VIP= 003; // approved(VIP)
RC_APPROVED_TRACK3= 004; // approved; update track 3
RC_APPROVED_ACCT_SPEC = 005; // approved, account type specified by card
issuer
RC_APPROVED_PARTIAL_SPEC= 006; // approved for partial amount; account
type specified by card issuer
RC_APPROVED_ICC= 007; // approved, update ICC


// 100-199 Used in 1110,1120,1121,1140 and 1210,1220,1221 and 1240
messages to indicate that the transaction has been processed for
authorization by or on behalf of the card issuer and has been denied(not
requiring a card pick-up)
RC_DECLINED_DO_NOT_HONOUR= 100; // do not honour
RC_DECLINED_EXPIRED_CARD= 101; // expired card
RC_DECLINED_SUSPECTED= 102; // suspected fraud
RC_DECLINED_CONTACT_ACQ= 103; // card acceptor contact acquirer
RC_DECLINED_RESTRICTED= 104; // restricted card
RC_DECLINED_CALL_ACQ= 105; // card acceptor call acquirer's security
department
RC_DECLINED_PIN_EXCEED= 106; // allowable PIN tries exceeded
RC_DECLINED_REFER_ISSR= 107; // refer to card issuer
RC_DECLINED_REFER_ISSR_COND= 108; // refer to card issuer's special
conditions
RC_DECLINED_INVALID_MERCHANT= 109; // invalid merchant
RC_DECLINED_INVALID_AMOUNT= 110; // invalid amount
RC_DECLINED_INVALID_CARD= 111; // invalid card number
RC_DECLINED_PIN_REQUIRED= 112; // PIN data required
RC_DECLINED_UNACCEPT_FEE= 113; // unacceptable fee
RC_DECLINED_ACCT_REQ= 114; // no account of type requested
RC_DECLINED_FUNC_NOT_SUPPORT= 115; // requested function not supported
RC_DECLINED_NOT_SUFF_FUNDS= 116; // not sufficient funds
RC_DECLINED_INCORRECT_PIN= 117; // incorrect PIN
RC_DECLINED_NO_CARD_RECORD= 118; // no card record
RC_DECLINED_NOT_ALLOW_CARDHOLDER= 119; // transaction not permitted to
cardholder
RC_DECLINED_NOT_ALLOW_TERMINAL= 120; // transaction not permitted to
terminal
RC_DECLINED_EXCEED_AMOUNT_LIMIT= 121; // exceeds withdrawal amount limit
RC_DECLINED_VIOLATION= 122; // security violation
RC_DECLINED_EXCEED_FREQ_LIMIT= 123; // exceeds withdrawal frequency limit
RC_DECLINED_VIOLATION_LAW= 124; // violation of law
RC_DECLINED_NOT_EFFECTIVE= 125; // card not effective
RC_DECLINED_INVALID_PIN= 126; // invalid PIN block
RC_DECLINED_PIN_LENGTH= 127; // PIN length error
RC_DECLINED_PIN_SYNC= 128; // PIN key synch error
```

```
RC_DECLINED_SUSPECTED_COUNTER= 129; // suspected counterfeit card

// 200-299 Used in 1110,1120,1121,1140 and 1210,1220,1221 and 1240 messages to indicate
// that the transaction has been processed for authorization by or on behalf of
// the card issuer and has been denied requiring a card to be pick-up.
RC_PICKUP_DO_NOT_HONOUR= 200; // do not honour
RC_PICKUP_EXPIRED_CARD= 201; // expired card
RC_PICKUP_SUSPECTED= 202; // suspected fraud
RC_PICKUP_CONTACT_ACQ= 203; // card acceptor contact acquirer
RC_PICKUP_RESTRICTED= 204; // restricted card
RC_PICKUP_CALL_ACQ= 205; // card acceptor call acquirer's security department
RC_PICKUP_PIN_EXCEED= 206; // allowable PIN tries exceeded
RC_PICKUP_SPEC_COND= 207; // special condition
RC_PICKUP_LOST= 208; // lost card
RC_PICKUP_STOLEN= 209; // stolen card
RC_PICKUP_SUSPECTED_COUNTER= 210; // suspected countfeid card

// 300-399 Used in 1314 1324,1325 and 1344 messages to indicate the result of the file action
RC_FILE_SUCCESS= 300; // successful
RC_FILE_NOT_SUPPORT= 301; // not supported by receiver
RC_FILE_UNABLE_LOCATE_RECORD= 302; // unable to locate record on file
RC_FILE_DUP_REPLACE= 303; // duplicate record; old record replaced
RC_FILE_EDIT_ERROR= 304; // field edit error
RC_FILE_LOCKED_OUT= 305; // file locked out
RC_FILE_NOT_SUCCESS= 306; // not successful
RC_FILE_FORMAT_ERROR= 307; // format error
RC_FILE_DUP_REJECT= 308; // duplicate; new record rejected
RC_FILE_UNKNOWN= 309; // unknown file

// 400-499 Used in 1430,1432,1440 and 1442 messages to indicate the result of the
// reversal or chargeback.
   RC_REVERSAL_ACCEPT= 400; // accepted

// 500-599 Used in 1510,1512,1530 and 1532 messages to indicate the result of a reconciliation.
RC_RECON_IN_BALANCE= 500; // reconciled; in balance
RC_RECON_OUT_BALANCE= 501; // reconciled; out balance
RC_RECON_AMOUNT_NOT_RECON= 502; // amount not reconciled; total provided
RC_RECON_TOTAL_NOT_AVAILABLE= 503; // totals not avalable
RC_RECON_NOT_RECON= 504; // not reconciled; totals provided

// 600-699 Used in 1614;1624;1625 and 1644 messages.
RC_ADMIN_ACCEPT= 600; // accepted
RC_ADMIN_NOT_TRACE_ORIGIN= 601; // not able to trace back original transaction
RC_ADMIN_INVALID_REFERENCE= 602; // invalid reference number
RC_ADMIN_PAN_INCOMPATIBLE= 603; // reference number/PAN incompatible
```

```
RC_ADMIN_PHOTO_NOT_AVAILABLE= 604; // POS photograph is not available
RC_ADMIN_ITEM_SUPP= 605; // item supplied
RC_ADMIN_DOC_NOT_SUPP= 606; // request cannot be fulfilled-required/
requested documentation is not available


// 700-799 Used in 1720;1721;1740;1722;1723 and 1742 messages.
   RC_FEE_ACCEPT= 700; // accepted


// 800-901 Used in 1814;1824;1825 and 1844 messages.
RC_NETWORK_ACCEPT= 800; // accepted
RC_NETWORK_NO_LIABILITY= 900; // advice acknowledged; no financial
liability accepted
RC_NETWORK_LIABILITY= 901; // advice acknowledged; financial liability
accepted
// 902-949 Used in request response and advice response messages to
indicate transaction
// could not be processed.
RC_REJECT_INVALID_TXN= 902; // invalid transaction
RC_REJECT_RE_ENTER_TXN= 903; // re-enter transaction
RC_REJECT_FORMAT_ERROR= 904; // format error
RC_REJECT_ACQ_NOT_SUPP= 905; // acquirer not supported by switch
RC_REJECT_CUTOVER_IN_PROCESS= 906; // cutover in process
RC_REJECT_ISSUER_INOPERATIVE= 907; // card issuer or switch inoperative
RC_REJECT_DEST_NOT_FOUND= 908; // transaction destination cannot be found
for routing
RC_REJECT_SYSTEM_MALFUNCTION= 909; // system malfunction
RC_REJECT_ISSUER_SIGNOFF= 910; // card issuer signed off
RC_REJECT_ISSUER_TIMEOUT= 911; // card issuer timed out
RC_REJECT_ISSUER_NOT_AVAILABLE= 912; // card issuer unavailable
RC_REJECT_DUP_TRANSMISSION= 913; // duplicate transmission
RC_REJECT_NOT_TRACE_ORIGIN= 914; // not able to trace back to original
transaction
RC_REJECT_CHECKPOINT_ERROR= 915; // reconciliation cutover or checkpoint
error
RC_REJECT_MAC_ERROR= 916; // MAC incorrect
RC_REJECT_MAC_KEY_SYNC= 917; // MAC key sync error
RC_REJECT_NO_COMM_KEY= 918; // no communication keys available for use
RC_REJECT_ENCRYPTION_KEY_SYNC= 919; // encryption key sync error
RC_REJECT_SECURITY_ERROR_TRY_AGAIN= 920; // security software/hardware
error - try again
RC_REJECT_SECURITY_ERROR_NO_ACTION= 921; // security software/hardware
error - no action
RC_REJECT_MSGNO_ERROR= 922; // message number out of sequence
RC_REJECT_REQ_IN_PROCESS= 923; // request in progress


// 950-999 Used in advice response(1x3x) to indicate the reason for
rejection of the transfer
// of financial liability.
RC_REJECT_VIOLATION= 950; // violation of business arrangement
```

POS data codes

- CardDataInputCapability (see below)

- CardHolderAuthenticationCapability (always 0)
- CardCaptureCapability (always 0)
- OperatingEnvironment (see below)
- CardHolderPresent (1 if present, 0 if not present)
- CardPresent (always set to 1)
- CardDataInputMode (see below)
- CardholderAuthenticationMethod (see below)
- CardAuthenticationEntity (see below)
- CardDataOutputCapability (always 0)
- TermninalOutputCapability (always 4)
- PINCapatureCapability (always 1)

Card data input capability and card data input mode

6 – Keyed

2 - Swiped

4 - MICR

Card Holder authentication methods

0 –Not authenticated

1 – PIN

2 – Electronic Signature Analysis

3 – Biometrics

4 – Biographic

5 – Manual Signature Verification

6 – Other Manual Verification

Card holder authentication entity

0 – Not authenticated

1 – ICC

2 – CAD

3- Authorizing Agent (Identified by a code)

4 – By Merchant

5 – Other

Operating environment codes

0 –No Terminal Used

1 – On premises of card acceptor, attended

2 - On premises of card acceptor, unattended

3 - Off premises of card acceptor, attended

4 - Off premises of card acceptor, unattended

5 - On premises of card holder, unattended

ISO 4217 Currency and funds code list

This table lists international currencies in alphabetical order by entity (source: http://www.bsi-global.com/Technical+Information/Publications/_Publications/tig90x.doc).

Entity	Currency	Code		Minor unit
		Alphabetic	Numeric	
AFGHANISTAN	Afghani	AFA	004	2
ALBANIA	Lek	ALL	008	2
ALGERIA	Algerian Dinar	DZD	012	2
AMERICAN SAMOA	US Dollar	USD	840	2
ANDORRA#	euro	EUR	978	2
	Spanish Peseta	ESP	724	0
	French Franc	FRF	250	2
	Andorran Peseta	ADP	020	0
ANGOLA	Kwanza	AOA	973	2
ANGUILLA	East Caribbean Dollar	XCD	951	2
ANTARCTICA	No universal currency			
ANTIGUA AND BARBUDA	East Caribbean Dollar	XCD	951	2
ARGENTINA	Argentine Peso	ARS	032	2
ARMENIA	Armenian Dram	AMD	051	2
ARUBA	Aruban Guilder	AWG	533	2
AUSTRALIA	Australian Dollar	AUD	036	2
AUSTRIA#	euro	EUR	978	2
	Schilling	ATS	040	2
AZERBAIJAN	Azerbaijanian Manat	AZM	031	2
BAHAMAS	Bahamian Dollar	BSD	044	2

BAHRAIN	Bahraini Dinar	BHD	048	3
BANGLADESH	Taka	BDT	050	2
BARBADOS	Barbados Dollar	BBD	052	2
BELARUS	Belarussian Ruble	BYR	974	0
BELGIUM#	euro Belgian Franc	EUR BEF	978 056	2 0
BELIZE	Belize Dollar	BZD	084	2
BENIN	CFA Franc BCEAO	XOF	952	0
BERMUDA	Bermudian Dollar (customarily known as Bermuda Dollar)	BMD	060	2
BHUTAN	Indian Rupee Ngultrum	INR BTN	356 064	2 2
BOLIVIA	Boliviano Mvdol*	BOB BOV	068 984	2 2
# Timetable for euro changeover detailed at the end of Table A1 CFA Franc BCEAO; Responsible authority: Banque Centrale des États de l'Afrique de l'Ouest. * Funds code [see Table A.2 (E) for definitions of funds types].				
BOSNIA & HERZEGOVINA	Convertible Marks	BAM	977	2
BOTSWANA	Pula	BWP	072	2
BOUVET ISLAND	Norwegian Krone	NOK	578	2
BRAZIL	Brazilian Real	BRL	986	2
BRITISH INDIAN OCEAN TERRITORY	US Dollar	USD	840	2
BRUNEI DARUSSALAM	Brunei Dollar	BND	096	2
BULGARIA	Lev Bulgarian Lev	BGL BGN	100 975	2 2
BURKINA FASO	CFA Franc BCEAO	XOF	952	0
BURUNDI	Burundi Franc	BIF	108	0
CAMBODIA	Riel	KHR	116	2
CAMEROON	CFA Franc BEAC	XAF	950	0
CANADA	Canadian Dollar	CAD	124	2
CAPE VERDE	Cape Verde Escudo	CVE	132	2
CAYMAN ISLANDS	Cayman Islands Dollar	KYD	136	2

CENTRAL AFRICAN REPUBLIC	CFA Franc BEAC	XAF	950	0
CHAD	CFA Franc BEAC	XAF	950	0
CHILE	Chilean Peso	CLP	152	0
	Unidades de fomento *	CLF	990	0
CHINA	Yuan Renminbi	CNY	156	2
CHRISTMAS ISLAND	Australian Dollar	AUD	036	2
COCOS (KEELING) ISLANDS	Australian Dollar	AUD	036	2
COLOMBIA	Colombian Peso	COP	170	2
COMOROS	Comoro Franc	KMF	174	0
CONGO	CFA Franc BEAC	XAF	950	0
CONGO, THE DEMOCRATIC REPUBLIC OF	Franc Congolais	CDF	976	2
COOK ISLANDS	New Zealand Dollar	NZD	554	2
COSTA RICA	Costa Rican Colon	CRC	188	2
CÔTE D'IVOIRE	CFA Franc BCEAO	XOF	952	0
CROATIA	Croatian Kuna	HRK	191	2
CUBA	Cuban Peso	CUP	192	2
CYPRUS	Cyprus Pound	CYP	196	2
CZECH REPUBLIC	Czech Koruna	CZK	203	2
DENMARK	Danish Krone	DKK	208	2
DJIBOUTI	Djibouti Franc	DJF	262	0
DOMINICA	East Caribbean Dollar	XCD	951	2
DOMINICAN REPUBLIC	Dominican Peso	DOP	214	2
CFA Franc BCEAO; Responsible authority: Banque Centrale des États de l'Afrique de l'Ouest. CFA Franc BEAC; Responsible authority: Banque des États de l'Afrique Centrale. * Funds code [see Table A.2 (E) for definitions of funds types].				
EAST TIMOR	Timor Escudo	TPE	626	0
	Rupiah	IDR	360	2
ECUADOR	US Dollar	USD	840	2
EGYPT	Egyptian Pound	EGP	818	2
EL SALVADOR	El Salvador Colon	SVC	222	2

EQUATORIAL GUINEA	CFA Franc BEAC	XAF	950	0
ERITREA	Nakfa	ERN	232	2
ESTONIA	Kroon	EEK	233	2
ETHIOPIA	Ethiopian Birr	ETB	230	2
FALKLAND ISLANDS (MALVINAS)	Falkland Islands Pound	FKP	238	2
FAROE ISLANDS	Danish Krone	DKK	208	2
FIJI	Fiji Dollar	FJD	242	2
FINLAND#	euro	EUR	978	2
	Markka	FIM	246	2
FRANCE#	euro	EUR	978	2
	French Franc	FRF	250	2
FRENCH GUIANA#	euro	EUR	978	2
	French Franc	FRF	250	2
FRENCH POLYNESIA	CFP Franc	XPF	953	0
FRENCH SOUTHERN TERRITORIES#	euro	EUR	978	2
	French Franc	FRF	250	2
GABON	CFA Franc BEAC	XAF	950	0
GAMBIA	Dalasi	GMD	270	2
GEORGIA	Lari	GEL	981	2
GERMANY#	euro	EUR	978	2
	Deutsche Mark	DEM	276	2
GHANA	Cedi	GHC	288	2
GIBRALTAR	Gibraltar Pound	GIP	292	2
GREECE#	euro	EUR	978	2
	Drachma	GRD	300	0
GREENLAND	Danish Krone	DKK	208	2
GRENADA	East Caribbean Dollar	XCD	951	2
GUADELOUPE#	euro	EUR	978	2
	French Franc	FRF	250	2
GUAM	US Dollar	USD	840	2
GUATEMALA	Quetzal	GTQ	320	2
GUINEA	Guinea Franc	GNF	324	0

GUINEA-BISSAU	Guinea-Bissau Peso	GWP	624	2
	CFA Franc BCEAO	XOF	952	0
CFA Franc BEAC; Responsible authority: Banque des États de l'Afrique Centrale. # Timetable for euro changeover detailed at the end of Table A1 CFA Franc BCEAO; Responsible authority: Banque Centrale des États de l'Afrique de l'Ouest.				
GUYANA	Guyana Dollar	GYD	328	2
HAITI	Gourde	HTG	332	2
	US Dollar	USD	840	2
HEARD ISLAND AND McDONALD ISLANDS	Australian Dollar	AUD	036	2
HOLY SEE (VATICAN CITY STATE) #	euro	EUR	978	2
	Italian Lira	ITL	380	0
HONDURAS	Lempira	HNL	340	2
HONG KONG	Hong Kong Dollar	HKD	344	2
HUNGARY	Forint	HUF	348	2
ICELAND	Iceland Krona	ISK	352	2
INDIA	Indian Rupee	INR	356	2
INDONESIA	Rupiah	IDR	360	2
INTERNATIONAL MONETARY FUND (I.M.F) **	SDR	XDR	960	N.A.
IRAN (ISLAMIC REPUBLIC OF)	Iranian Rial	IRR	364	2
IRAQ	Iraqi Dinar	IQD	368	3
IRELAND#	euro	EUR	978	2
	Irish Pound	IEP	372	2
ISRAEL	New Israeli Sheqel***	ILS	376	2
ITALY#	euro	EUR	978	2
	Italian Lira	ITL	380	0
JAMAICA	Jamaican Dollar	JMD	388	2
JAPAN	Yen	JPY	392	0
JORDAN	Jordanian Dinar	JOD	400	3
KAZAKSTAN	Tenge	KZT	398	2
KENYA	Kenyan Shilling	KES	404	2
KIRIBATI	Australian Dollar	AUD	036	2

KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF	North Korean Won	KPW	408	2
KOREA, REPUBLIC OF	Won	KRW	410	0
KUWAIT	Kuwaiti Dinar	KWD	414	3
KYRGYZSTAN	Som	KGS	417	2
LAO PEOPLE'S DEMOCRATIC REPUBLIC	Kip	LAK	418	2
LATVIA	Latvian Lats	LVL	428	2
LEBANON	Lebanese Pound	LBP	422	2
LESOTHO	Rand Loti	ZAR LSL	710 426	2 2
# Timetable for euro changeover detailed at the end of Table A1 ** This entry is not derived from ISO 3166, but is included here in alphabetic sequence for convenience. *** Currency name was effective 4th September 1985.				
LIBERIA	Liberian Dollar	LRD	430	2
LIBYAN ARAB JAMAHIRIYA	Libyan Dinar	LYD	434	3
LIECHTENSTEIN	Swiss Franc	CHF	756	2
LITHUANIA	Lithuanian Litus	LTL	440	2
LUXEMBOURG#	euro Luxembourg Franc	EUR LUF	978 442	2 0
MACAU	Pataca	MOP	446	2
MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF	Denar	MKD	807	2
MADAGASCAR	Malagasy Franc	MGF	450	0
MALAWI	Kwacha	MWK	454	2
MALAYSIA	Malaysian Ringgit	MYR	458	2
MALDIVES	Rufiyaa	MVR	462	2
MALI	CFA Franc BCEAO	XOF	952	0
MALTA	Maltese Lira	MTL	470	2
MARSHALL ISLANDS	US Dollar	USD	840	2
MARTINIQUE#	euro French Franc	EUR FRF	978 250	2 2
MAURITANIA	Ouguiya	MRO	478	2

MAURITIUS	Mauritius Rupee	MUR	480	2
MAYOTTE#	euro	EUR	978	2
	French Franc	FRF	250	2
MEXICO	Mexican Peso	MXN	484	2
	Mexican Unidad de Inversion (UDI)*	MXV	979	2
MICRONESIA (FEDERATED STATES OF)	US Dollar	USD	840	2
MOLDOVA, REPUBLIC OF	Moldovan Leu	MDL	498	2
MONACO#	euro	EUR	978	2
	French Franc	FRF	250	2
MONGOLIA	Tugrik	MNT	496	2
MONTSERRAT	East Caribbean Dollar	XCD	951	2
MOROCCO	Moroccan Dirham	MAD	504	2
MOZAMBIQUE	Metical	MZM	508	2
MYANMAR	Kyat	MMK	104	2
NAMIBIA	Rand	ZAR	710	2
	Namibia Dollar	NAD	516	2
NAURU	Australian Dollar	AUD	036	2
NEPAL	Nepalese Rupee	NPR	524	2
# Timetable for euro changeover detailed at the end of Table A1 CFA Franc BCEAO; Responsible authority: Banque Centrale des États de l'Afrique de l'Ouest. * Funds code [see Table A.2 (E) for definitions of funds types].				
NETHERLANDS#	euro	EUR	978	2
	Netherlands Guilder	NLG	528	2
NETHERLANDS ANTILLES	Netherlands Antillian Guilder	ANG	532	2
NEW CALEDONIA	CFP Franc	XPF	953	0
NEW ZEALAND	New Zealand Dollar	NZD	554	2
NICARAGUA	Cordoba Oro	NIO	558	2
NIGER	CFA Franc BCEAO	XOF	952	0
NIGERIA	Naira	NGN	566	2
NIUE	New Zealand Dollar	NZD	554	2
NORFOLK ISLAND	Australian Dollar	AUD	036	2

NORTHERN MARIANA ISLANDS	US Dollar	USD	840	2
NORWAY	Norwegian Krone	NOK	578	2
OMAN	Rial Omani	OMR	512	3
PAKISTAN	Pakistan Rupee	PKR	586	2
PALAU	US Dollar	USD	840	2
PANAMA	Balboa	PAB	590	2
	US Dollar	USD	840	2
PAPUA NEW GUINEA	Kina	PGK	598	2
PARAGUAY	Guarani	PYG	600	0
PERU	Nuevo Sol	PEN	604	2
PHILIPPINES	Philippine Peso	PHP	608	2
PITCAIRN	New Zealand Dollar	NZD	554	2
POLAND	Zloty	PLN	985	2
PORTUGAL#	euro	EUR	978	2
	Portuguese Escudo	PTE	620	0
PUERTO RICO	US Dollar	USD	840	2
QATAR	Qatari Rial	QAR	634	2
RÉUNION#	euro	EUR	978	2
	French Franc	FRF	250	2
ROMANIA	Leu	ROL	642	2
RUSSIAN FEDERATION	Russian Ruble	RUR	810	2
	Russian Ruble	RUB	643	2
RWANDA	Rwanda Franc	RWF	646	0
SAINT HELENA	Saint Helena Pound	SHP	654	2
SAINT KITTS AND NEVIS	East Caribbean Dollar	XCD	951	2
SAINT LUCIA	East Caribbean Dollar	XCD	951	2
SAINT PIERRE AND MIQUELON#	euro	EUR	978	2
	French Franc	FRF	250	2
SAINT VINCENT AND THE GRENADINES	East Caribbean Dollar	XCD	951	2
# Timetable for euro changeover detailed at the end of Table A1 CFA Franc BCEAO; Responsible authority: Banque Centrale des États de l'Afrique de l'Ouest.				
SAMOA	Tala	WST	882	2

SAN MARINO#	euro Italian Lira	EUR ITL	978 380	2 0
SÃO TOME AND PRINCIPE	Dobra	STD	678	2
SAUDI ARABIA	Saudi Riyal	SAR	682	2
SENEGAL	CFA Franc BCEAO	XOF	952	0
SEYCHELLES	Seychelles Rupee	SCR	690	2
SIERRA LEONE	Leone	SLL	694	2
SINGAPORE	Singapore Dollar	SGD	702	2
SLOVAKIA	Slovak Koruna	SKK	703	2
SLOVENIA	Tolar	SIT	705	2
SOLOMON ISLANDS	Solomon Islands Dollar	SBD	090	2
SOMALIA	Somali Shilling	SOS	706	2
SOUTH AFRICA	Rand	ZAR	710	2
SPAIN#	euro Spanish Peseta	EUR ESP	978 724	2 0
SRI LANKA	Sri Lanka Rupee	LKR	144	2
SUDAN	Sudanese Dinar	SDD	736	2
SURINAME	Suriname Guilder	SRG	740	2
SVALBARD AND JAN MAYEN	Norwegian Krone	NOK	578	2
SWAZILAND	Lilangeni	SZL	748	2
SWEDEN	Swedish Krona	SEK	752	2
SWITZERLAND	Swiss Franc	CHF	756	2
SYRIAN ARAB REPUBLIC	Syrian Pound	SYP	760	2
TAIWAN, PROVINCE OF CHINA	New Taiwan Dollar	TWD	901	2
TAJIKISTAN	Somoni	TJS	972	2
TANZANIA, UNITED REPUBLIC OF	Tanzanian Shilling	TZS	834	2
THAILAND	Baht	THB	764	2
TOGO	CFA Franc BCEAO	XOF	952	0
TOKELAU	New Zealand Dollar	NZD	554	2
TONGA	Pa'anga	TOP	776	2

TRINIDAD AND TOBAGO	Trinidad and Tobago Dollar	TTD	780	2
TUNISIA	Tunisian Dinar	TND	788	3
TURKEY	Turkish Lira	TRL	792	0
TURKMENISTAN	Manat	TMM	795	2
TURKS AND CAICOS ISLANDS	US Dollar	USD	840	2
TUVALU	Australian Dollar	AUD	036	2
# Timetable for euro changeover detailed at the end of Table A1 CFA Franc BCEAO; Responsible authority: Banque Centrale des États de l'Afrique de l'Ouest.				
UGANDA	Uganda Shilling	UGX	800	2 **
UKRAINE	Hryvnia	UAH	980	2
UNITED ARAB EMIRATES	UAE Dirham	AED	784	2
UNITED KINGDOM	Pound Sterling	GBP	826	2
UNITED STATES	US Dollar (Same day) * (Next day) *	USD USS USN	840 998 997	2 2 2
UNITED STATES MINOR OUTLYING ISLANDS	US Dollar	USD	840	2
URUGUAY	Peso Uruguayo	UYU	858	2
UZBEKISTAN	Uzbekistan Sum	UZS	860	2
VANUATU	Vatu	VUV	548	0
VENEZUELA	Bolivar	VEB	862	2
VIET NAM	Dong	VND	704	2
VIRGIN ISLANDS (BRITISH)	US Dollar	USD	840	2
VIRGIN ISLANDS (US)	US Dollar	USD	840	2
WALLIS AND FUTUNA	CFP Franc	XPF	953	0
WESTERN SAHARA	Moroccan Dirham	MAD	504	2
YEMEN	Yemeni Rial	YER	886	2
YUGOSLAVIA	Yugoslavian Dinar	YUM	891	2
ZAMBIA	Kwacha	ZMK	894	2
ZIMBABWE	Zimbabwe Dollar	ZWD	716	2

** The minor unit was changed from 0 to 2, for this edition, since the English Version of ISO 4217 had the wrong information. * Funds code [see Table A.2 (E) for definitions of funds types].				
Entity not applicable	Gold	XAU	959	N.A.
	Bond Markets Units European Composite Unit (EURCO) European Monetary Unit (E.M.U.-6) *** European Unit of Account 9 (E.U.A.-9) European Unit of Account 17 (E.U.A.-17)	 XBA XBB XBC XBD	 955 956 957 958	 N.A. N.A. N.A. N.A.
	Palladium	XPD	964	N.A.
	Platinum	XPT	962	N.A.
	Silver	XAG	961	N.A.
	Special settlement currencies			
	UIC-Franc	XFU	Nil	N.A.
	Gold-Franc	XFO	Nil	N.A.
	Codes specifically reserved for testing purposes	XTS	963	N.A.
	The codes assigned for transactions where no currency is involved are:	XXX	999	N.A.

	euro*	EUR*	978	2

*** E.M.U.-6 is sometimes known as the European Currency Unit. This should not be confused with the settlement unit of the European
Monetary Co-operation Fund (E.M.C.F.) which has the same name.

* On 1999-01-01, the euro became the currency of those Member States of the European Union which adopted the single currency in accordance with the Treaty establishing the European Community. The code element 'EU' was reserved by the ISO 3166 Maintenance Agency for use within ISO 4217 where 'R' had been appended to make an acceptable mnemonic code.

EUR used for scriptural from 1999-01-01 (2001-01-01 for Greece) and for cash as from 2002-01-01

ATS used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-02-28

BEF used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-02-28

FIM used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-02-28

FRF used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-02-17

DEM used for scriptural until 2001-12-31 and for cash until end of legal tender 2001-12-31. Businesses will, however, accept national

currency unis at least until 2002-02-28, according to the joint statement of professional associations of 1998-10-22.

GRD used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-02-28

IEP used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-02-09

ITL used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-02-28

LUF used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-02-28

NLG used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-01-28

PTE used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-02-28

ESP used for scriptural until 2001-12-31 and for cash until end of legal tender 2002-02-28

Language codes

The language code should be in ISO 639-2 format. For example, English = eng; French = fre.

Source: http://www.loc.gov/standards/iso639-2/englangn.html#two

Language Name (English)	639-2	639-1
Abkhazian	abk	ab
Achinese	ace	
Acoli	ach	
Adangme	ada	
Afar	aar	aa
Afrihili	afh	

Afrikaans	afr	af
Afro-Asiatic (Other)	afa	
Akan	aka	ak
Akkadian	akk	
Albanian	alb/sqi*	sq
Aleut	ale	
Algonquian languages	alg	
Altaic (Other)	tut	
Amharic	amh	am
Apache languages	apa	
Arabic	ara	ar
Aragonese	arg	an
Aramaic	arc	
Arapaho	arp	
Araucanian	arn	
Arawak	arw	
Armenian	arm/ hye*	hy
Artificial (Other)	art	
Assamese	asm	as
Asturian; Bable	ast	
Athapascan languages	ath	
Australian languages	aus	
Austronesian (Other)	map	
Avaric	ava	av
Avestan	ave	ae
Awadhi	awa	
Aymara	aym	ay
Azerbaijani	aze	az
Bable; Asturian	ast	

Balinese	ban	
Baltic (Other)	bat	
Baluchi	bal	
Bambara	bam	bm
Bamileke languages	bai	
Banda	bad	
Bantu (Other)	bnt	
Basa	bas	
Bashkir	bak	ba
Basque	baq/eus*	eu
Batak (Indonesia)	btk	
Beja	bej	
Belarusian	bel	be
Bemba	bem	
Bengali	ben	bn
Berber (Other)	ber	
Bhojpuri	bho	
Bihari	bih	bh
Bikol	bik	
Bini	bin	
Bislama	bis	bi
Bokmål, Norwegian; Norwegian Bokmål	nob	nb
Bosnian	bos	bs
Braj	bra	
Breton	bre	br
Buginese	bug	
Bulgarian	bul	bg
Buriat	bua	
Burmese	bur/ mya*	my

Caddo	cad	
Carib	car	
Castilian; Spanish	spa	es
Catalan	cat	ca
Caucasian (Other)	cau	
Cebuano	ceb	
Celtic (Other)	cel	
Central American Indian (Other)	cai	
Chagatai	chg	
Chamic languages	cmc	
Chamorro	cha	ch
Chechen	che	ce
Cherokee	chr	
Chewa; Chichewa; Nyanja	nya	ny
Cheyenne	chy	
Chibcha	chb	
Chichewa; Chewa; Nyanja	nya	ny
Chinese	chi/zho*	zh
Chinook jargon	chn	
Chipewyan	chp	
Choctaw	cho	
Chuang; Zhuang	zha	za
Church Slavic; Slavonic; Church Slavonic; Old Bulgarian; Old Church Slavonic	chu	cu
Church Slavonic; Church Slavic; Old Slavonic; Old Bulgarian; Old Church Slavonic	chu	cu
Chuukese	chk	
Chuvash	chv	cv
Coptic	cop	
Cornish	cor	kw

Corsican	cos	co
Cree	cre	cr
Creek	mus	
Creoles and pidgins (Other)	crp	
Creoles and pidgins, English-based (Other)	cpe	
Creoles and pidgins, French-based (Other)	cpf	
Creoles and pidgins, Portuguese-based (Other)	cpp	
Croatian	scr/hrv*	hr
Cushitic (Other)	cus	
Czech	cze/ces*	cs
Dakota	dak	
Danish	dan	da
Dayak	day	
Delaware	del	
Dinka	din	
Divehi	div	dv
Dogri	doi	
Dogrib	dgr	
Dravidian (Other)	dra	
Duala	dua	
Dutch; Flemish	dut/nld*	nl
Dutch, Middle (ca. 1050-1350)	dum	
Dyula	dyu	
Dzongkha	dzo	dz
Efik	efi	
Egyptian (Ancient)	egy	
Ekajuk	eka	
Elamite	elx	
English	eng	en

English, Middle (1100-1500)	enm	
English, Old (ca.450-1100)	ang	
Esperanto	epo	eo
Estonian	est	et
Ewe	ewe	ee
Ewondo	ewo	
Fang	fan	
Fanti	fat	
Faroese	fao	fo
Fijian	fij	fj
Finnish	fin	fi
Finno-Ugrian (Other)	fiu	
Flemish; Dutch	dut/nld*	nl
Fon	fon	
French	fre/fra*	fr
French, Middle (ca.1400-1600)	frm	
French, Old (842-ca.1400)	fro	
Frisian	fry	fy
Friulian	fur	
Fulah	ful	ff
Ga	gaa	
Gaelic; Scottish Gaelic	gla	gd
Gallegan	glg	gl
Ganda	lug	lg
Gayo	gay	
Gbaya	gba	
Geez	gez	
Georgian	geo/kat*	ka
German	ger/deu*	de

German, Low; Saxon, Low; Low German; Low Saxon	nds	
German, Middle High (ca.1050-1500)	gmh	
German, Old High (ca.750-1050)	goh	
Germanic (Other)	gem	
Gikuyu; Kikuyu	kik	ki
Gilbertese	gil	
Gondi	gon	
Gorontalo	gor	
Gothic	got	
Grebo	grb	
Greek, Ancient (to 1453)	grc	
Greek, Modern (1453-)	gre/ell*	el
Greenlandic; Kalaallisut	kal	kl
Guarani	grn	gn
Gujarati	guj	gu
Gwich´in	gwi	
Haida	hai	
Hausa	hau	ha
Hawaiian	haw	
Hebrew	heb	he
Herero	her	hz
Hiligaynon	hil	
Himachali	him	
Hindi	hin	hi
Hiri Motu	hmo	ho
Hittite	hit	
Hmong	hmn	
Hungarian	hun	hu
Hupa	hup	

Iban	iba	
Icelandic	ice/isl*	is
Ido	ido	io
Igbo	ibo	ig
Ijo	ijo	
Iloko	ilo	
Inari Sami	smn	
Indic (Other)	inc	
Indo-European (Other)	ine	
Indonesian	ind	id
Interlingua (International Auxiliary Language Association)	ina	ia
Interlingue	ile	ie
Inuktitut	iku	iu
Inupiaq	ipk	ik
Iranian (Other)	ira	
Irish	gle	ga
Irish, Middle (900-1200)	mga	
Irish, Old (to 900)	sga	
Iroquoian languages	iro	
Italian	ita	it
Japanese	jpn	ja
Javanese	jav	jv
Judeo-Arabic	jrb	
Judeo-Persian	jpr	
Kabyle	kab	
Kachin	kac	
Kalaallisut; Greenlandic	kal	kl
Kamba	kam	
Kannada	kan	kn

Kanuri	kau	kr
Kara-Kalpak	kaa	
Karen	kar	
Kashmiri	kas	ks
Kawi	kaw	
Kazakh	kaz	kk
Khasi	kha	
Khmer	khm	km
Khoisan (Other)	khi	
Khotanese	kho	
Kikuyu; Gikuyu	kik	ki
Kimbundu	kmb	
Kinyarwanda	kin	rw
Kirghiz	kir	ky
Komi	kom	kv
Kongo	kon	kg
Konkani	kok	
Korean	kor	ko
Kosraean	kos	
Kpelle	kpe	
Kru	kro	
Kuanyama; Kwanyama	kua	kj
Kumyk	kum	
Kurdish	kur	ku
Kurukh	kru	
Kutenai	kut	
Kwanyama, Kuanyama	kua	kj
Ladino	lad	
Lahnda	lah	
Lamba	lam	

Lao	lao	lo
Latin	lat	la
Latvian	lav	lv
Letzeburgesch; Luxembourgish	ltz	lb
Lezghian	lez	
Limburgan; Limburger; Limburgish	lim	li
Limburger; Limburgan; Limburgish;	lim	li
Limburgish; Limburger; Limburgan	lim	li
Lingala	lin	ln
Lithuanian	lit	lt
Low German; Low Saxon; German, Low; Saxon, Low	nds	
Low Saxon; Low German; Saxon, Low; German, Low	nds	
Lozi	loz	
Luba-Katanga	lub	lu
Luba-Lulua	lua	
Luiseno	lui	
Lule Sami	smj	
Lunda	lun	
Luo (Kenya and Tanzania)	luo	
Lushai	lus	
Luxembourgish; Letzeburgesch	ltz	lb
Macedonian	mac/ mkd*	mk
Madurese	mad	
Magahi	mag	
Maithili	mai	
Makasar	mak	
Malagasy	mlg	mg

Malay	may/ msa*	ms
Malayalam	mal	ml
Maltese	mlt	mt
Manchu	mnc	
Mandar	mdr	
Mandingo	man	
Manipuri	mni	
Manobo languages	mno	
Manx	glv	gv
Maori	mao/ mri*	mi
Marathi	mar	mr
Mari	chm	
Marshallese	mah	mh
Marwari	mwr	
Masai	mas	
Mayan languages	myn	
Mende	men	
Micmac	mic	
Minangkabau	min	
Miscellaneous languages	mis	
Mohawk	moh	
Moldavian	mol	mo
Mon-Khmer (Other)	mkh	
Mongo	lol	
Mongolian	mon	mn
Mossi	mos	
Multiple languages	mul	
Munda languages	mun	

Nahuatl	nah	
Nauru	nau	na
Navaho, Navajo	nav	nv
Navajo; Navaho	nav	nv
Ndebele, North	nde	nd
Ndebele, South	nbl	nr
Ndonga	ndo	ng
Neapolitan	nap	
Nepali	nep	ne
Newari	new	
Nias	nia	
Niger-Kordofanian (Other)	nic	
Nilo-Saharan (Other)	ssa	
Niuean	niu	
Norse, Old	non	
North American Indian (Other)	nai	
Northern Sami	sme	se
North Ndebele	nde	nd
Norwegian	nor	no
Norwegian Bokmål; Bokmål, Norwegian	nob	nb
Norwegian Nynorsk; Nynorsk, Norwegian	nno	nn
Nubian languages	nub	
Nyamwezi	nym	
Nyanja; Chichewa; Chewa	nya	ny
Nyankole	nyn	
Nynorsk, Norwegian; Norwegian Nynorsk	nno	nn
Nyoro	nyo	
Nzima	nzi	
Occitan (post 1500); Provençal	oci	oc
Ojibwa	oji	oj

Old Bulgarian; Old Slavonic; Church Slavonic; Church Slavic; Old Church Slavonic	chu	cu
Old Church Slavonic; Old Slavonic; Church Slavonic; Old Bulgarian; Church Slavic	chu	cu
Old Slavonic; Church Slavonic; Old Bulgarian; Church Slavic; Old Church Slavonic	chu	cu
Oriya	ori	or
Oromo	orm	om
Osage	osa	
Ossetian; Ossetic	oss	os
Ossetic; Ossetian	oss	os
Otomian languages	oto	
Pahlavi	pal	
Palauan	pau	
Pali	pli	pi
Pampanga	pam	
Pangasinan	pag	
Panjabi	pan	pa
Papiamento	pap	
Papuan (Other)	paa	
Persian	per/fas*	fa
Persian, Old (ca.600-400 B.C.)	peo	
Philippine (Other)	phi	
Phoenician	phn	
Pohnpeian	pon	
Polish	pol	pl
Portuguese	por	pt
Prakrit languages	pra	
Provençal; Occitan (post 1500)	oci	oc
Provençal, Old (to 1500)	pro	
Pushto	pus	ps

Quechua	que	qu
Raeto-Romance	roh	rm
Rajasthani	raj	
Rapanui	rap	
Rarotongan	rar	
Reserved for local use	qaa-qtz	
Romance (Other)	roa	
Romanian	rum/ron*	ro
Romany	rom	
Rundi	run	rn
Russian	rus	ru
Salishan languages	sal	
Samaritan Aramaic	sam	
Sami languages (Other)	smi	
Samoan	smo	sm
Sandawe	sad	
Sango	sag	sg
Sanskrit	san	sa
Santali	sat	
Sardinian	srd	sc
Sasak	sas	
Saxon, Low; German, Low; Low Saxon; Low German	nds	
Scots	sco	
Scottish Gaelic; Gaelic	gla	gd
Selkup	sel	
Semitic (Other)	sem	
Serbian	scc/srp*	sr
Serer	srr	
Shan	shn	

Shona	sna	sn
Sidamo	sid	
Sign languages	sgn	
Siksika	bla	
Sindhi	snd	sd
Sinhalese	sin	si
Sino-Tibetan (Other)	sit	
Siouan languages	sio	
Skolt Sami	sms	
Slave (Athapascan)	den	
Slavic (Other)	sla	
Slovak	slo/slk*	sk
Slovenian	slv	sl
Sogdian	sog	
Somali	som	so
Songhai	son	
Soninke	snk	
Sorbian languages	wen	
Sotho, Northern	nso	
Sotho, Southern	sot	st
South American Indian (Other)	sai	
Southern Sami	sma	
South Ndebele	nbl	nr
Spanish; Castilian	spa	es
Sukuma	suk	
Sumerian	sux	
Sundanese	sun	su
Susu	sus	
Swahili	swa	sw

Swati	ssw	ss
Swedish	swe	sv
Syriac	syr	
Tagalog	tgl	tl
Tahitian	tah	ty
Tai (Other)	tai	
Tajik	tgk	tg
Tamashek	tmh	
Tamil	tam	ta
Tatar	tat	tt
Telugu	tel	te
Tereno	ter	
Tetum	tet	
Thai	tha	th
Tibetan	tib/bod*	bo
Tigre	tig	
Tigrinya	tir	ti
Timne	tem	
Tiv	tiv	
Tlingit	tli	
Tok Pisin	tpi	
Tokelau	tkl	
Tonga (Nyasa)	tog	
Tonga (Tonga Islands)	ton	to
Tsimshian	tsi	
Tsonga	tso	ts
Tswana	tsn	tn
Tumbuka	tum	
Tupi languages	tup	
Turkish	tur	tr

Turkish, Ottoman (1500-1928)	ota	
Turkmen	tuk	tk
Tuvalu	tvl	
Tuvinian	tyv	
Twi	twi	tw
Ugaritic	uga	
Uighur	uig	ug
Ukrainian	ukr	uk
Umbundu	umb	
Undetermined	und	
Urdu	urd	ur
Uzbek	uzb	uz
Vai	vai	
Venda	ven	ve
Vietnamese	vie	vi
Volapük	vol	vo
Votic	vot	
Wakashan languages	wak	
Walamo	wal	
Walloon	wln	wa
Waray	war	
Washo	was	
Welsh	wel/ cym*	cy
Wolof	wol	wo
Xhosa	xho	xh
Yakut	sah	
Yao	yao	
Yapese	yap	
Yiddish	yid	yi

Yoruba	yor	yo
Yupik languages	ypk	
Zande	znd	
Zapotec	zap	
Zenaga	zen	
Zhuang; Chuang	zha	za
Zulu	zul	zu
Zuni	zun	

Reason codes

Reason Code (ID)	**Reason Description**
1005	Above Floor Limit (Phone Auth)
1006	Below Floor Limit

Financial message specification overview

All financial messages should contain the Originator and MessageType. A financial message can be either a request or response message type.

The top level structure of the financial request message is:

```
<FinancialMessage>
   <Originator>
   </Originator>

   <MessageType>
   </MessageType>

   <Request>
   </Request>
</FinancialMessage>
```

And for the financial response message is:

```
<FinancialMessage>
   <Originator>
   </Originator>

   <MessageType>
   </MessageType>

   <Request>
   </Request>

   <Response>
   </Response>
</FinancialMessage>
```

Note:

- All the elements that are optional may not be included. They will be required for some transaction types only.
- The Originator element and MessageType must always be present in the request as well as the response elements
- The Request element will be present in the request as well as response messages

Financial request message

The following is a generic financial request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain> </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>
```

```
    </MessageType>

    <Request >
        <TransactionNumber>  </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <CardDetail> </CardDetail>
        <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
        <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]

        <PinData> </PinData> [Optional]
        <SecurityControlInfo></SecurityControlInfo> [Optional]
        <MAC> </MAC> [Optional]

        <Plan> </Plan> [Optional]
        <ProductCodes></ProductCodes> [Optional]
        <DownPayment> {Amount Aggregate} </DownPayment> [Optional]

        <OriginalMessage> [Optional]
        <TransactionNumber>  </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime></TransactionDateTime>
        <CardDetail>  </CardDetail>
        <RequestedAmount> {Amount Aggregate} </RequestedAmount> [Optional]

        <ApprovalCode> </ApprovalCode> [Optional]
        <ExternalReferenceNumber> </ExternalReferenceNumber> [Optional]
        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount> [Optional]
        </OriginalMessage>

        <NetAmount> {Amount Aggregate} </NetAmount>

        </TenderTotal>

    </Request>
</FinancialMessage >
```

Note:

- The Originator, MessageType and Request elements are mandatory for a financial request message.
- OriginalMessage can hold all the elements from the request as well as response. Only the most frequently used elements are shown above. The elements not declared as optional must be provided.
- PinData, SecurityControlInfo and MAC contain binary data. Please see the section on Binary Data Types as to how binary data types are represented.

Financial response message

The following is a generic financial response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain> </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   </MessageType>

   </Request>

   <Response>
      <ActionCode ID="" Desc=""/>
      <ApprovalCode> </ApprovalCode>
      <ExternalReferenceNumber> </ExternalReferenceNumber>

      <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
      <BalanceAmount> {Amount Aggregate} </BalanceAmount> [Optional]
      <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]

      <ExpiryDate > </ExpiryDate> [Optional]
      <EffectiveDate> </EffectiveDate> [Optional]

      <Receipt> </Receipt> [Optional]
      <SettlementData> </Settlement>ntData> [Optional]

   </Response>
</FinancialMessage >
```

Note:

- The Originator, MessageType, Request, and Response elements are mandatory for a financial response message.
- The Balance will be present only if applicable.
- The Receipt and SettlementData information will be sent only if applicable.
- If ExpiryDate is provided in the response, then this expiry date should be provided in the original message request.
- The CashBackAmount should be included in the RequestedAmount and ApprovedAmount.

Financial reconciliation request message

The following is a financial reconciliation request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain> </Chain>
      <Store>  </Store>
```

```
        <Register>  </Register>
    </Originator>

    </MessageType>

    <Request>
        <TransactionNumber>  </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime></TransactionDateTime>

        <NetAmount> {Amount Aggregate} </NetAmount>

        </TenderTotal>

    </Request>
</FinancialMessage>
```

Financial reconciliation response message

The following is a financial reconciliation response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain> </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    </MessageType>

    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>
        <ExternalReferenceNumber> </ExternalReferenceNumber>


        <NetAmount> {Amount Aggregate} </NetAmount>

        </TenderTotal>

    </Response>
</FinancialMessage>
```

Financial settlement request message

The following is a financial settlement request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain> </Chain>
```

```
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    </MessageType>

    <Request>
        <TransactionNumber>  </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime></TransactionDateTime>

        <NetAmount> {Amount Aggregate} </NetAmount>

        </TenderTotal>

    </Request>
</FinancialMessage>
```

Financial settlement response message

The following is a financial settlement response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain> </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    </MessageType>

    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>
        <ExternalReferenceNumber> </ExternalReferenceNumber>


        <NetAmount> {Amount Aggregate} </NetAmount>

        </TenderTotal>

    </Response>
</FinancialMessage>
```

Note: For reconciliation and settlement messages, there should be a separate </TenderTotal> element for each tender. If the totals represent all tenders, specify the tender description as "ALL".

Financial sign on request message

The following is a generic financial sign-on message containing all required elements:

```
<FinancialMessage>
   <Originator>
      <Chain> </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType>None</AccountType>
      <Function>SignOn </Function>
      <Modifier>Original </Modifier>
      <StoreAndForward> False <StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>

   <Request>
   </Request>

</FinancialMessage >
```

Note: The Originator and MessageType elements are mandatory for a financial sign on request message.

Financial sign on response message

The following is a generic financial sign-on response message containing all required elements:

```
<FinancialMessage>
   <Originator>
      <Chain> </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   </MessageType>

   </Request>

   <Response>
      <MerchantInfo>
         <CreditName></CreditName>
         <DebitName></DebitName>
         <PinPadKeyexchangeRequired></PinPadKeyExchangeRequired>
         [Optional]
         <PinPadFormattedDataExchangeRequired>
         </PinPadFormattedDataExchangeRequired> [Optional]
      </MerchantInfo>
```

```
    </Response>
</FinancialMessage >
```

Note: The Originator, MessageType and Response elements are mandatory for a financial sign on response message.

Message format types

The following are the types of message formats:

- Regular type messages
- Reference type messages
- SAF type messages
- Reversal type messages

Regular type messages

Regular message types are used for regular type of transactions such as authorizations (for purchase), or returns. These types of messages will have the Modifier set to Original.

The following is an example of a regular type authorization (purchase) message.

```
<FinancialMessage>
  <Originator>
    <Chain></Chain>
    <Store></Store>
    <Register></Register>
  </Originator>
  <MessageType>
    <Function>Authorization</Function>
    <Modifier>Original</Modifier>
    <AccountType>CreditCard</AccountType>
    <StoreAndForward> false </StoreAndForward>
    <AdditionalModifier>Original</AdditionalModifier>
  </MessageType>
  <Request>
    <TransactionNumber></TransactionNumber>
    <SequenceNumber></SequenceNumber>
    <TransactionDateTime></TransactionDateTime>
    <CardDetail>
      <InputMode> </InputMode>
      <CardPresent> </CardPresent>
      <AccountNumber></AccountNumber>
      <ExpiryDate> </ExpiryDate>
    </CardDetail>
    <RequestedAmount Value="" CurrencyCode=" " />
  </Request>
</FinancialMessage>
```

Reference type messages

Reference type messages refer to regular type messages for voids or post authorization (completion) messages. These types of messages will have the OriginalMessage element present in the request that refer to the original regular type of message. The OriginalMessage element will contain both request and response elements. If the original message has been approved offline (SAF), then it will not contain the response element but will contain the OfflineResponse in the Request element.

Void messages will have the Modifier set to Void.

Below are two examples of void messages. The first message had the original message approved online and the second message has original message approved offline.

Example 1 - Original message approved online

```
<FinancialMessage>
     <Originator>
          <Chain></Chain>
          <Store></Store>
          <Register></Register>
     </Originator>
     <MessageType>
          <Function>Authorization</Function>
          <Modifier>Void</Modifier>
          <AccountType>CreditCard</AccountType>
          <StoreAndForward> false </StoreAndForward>
          <AdditionalModifier>Original</AdditionalModifier>
     </MessageType>
   <Request>
          <TransactionNumber></TransactionNumber>
          <SequenceNumber></SequenceNumber>
          <TransactionDateTime></TransactionDateTime>
          <OriginalMessage>
   <Request>
      <TransactionNumber> </TransactionNumber>
      <SequenceNumber> </SequenceNumber>
      <TransactionDateTime></TransactionDateTime>
      <CardDetail>
         <InputMode> </InputMode>
         <CardPresent> </CardPresent>
         <AccountNumber></AccountNumber>
         <ExpiryDate></ExpiryDate>
      </CardDetail>
      <RequestedAmount Value="" CurrencyCode="" />
   </Request>
   <Response>
      <ExternalReferenceNumber> </ExternalReferenceNumber>
      <ApprovalCode></ApprovalCode >
      <ApprovedAmount Value="" CurrencyCode="" />
      </Response>
      </OriginalMessage>
   </Request>
```

```
</FinancialMessage>
```

Example 2 - Original message approved offline

```
<FinancialMessage>
   <Originator>
      <Chain></Chain>
      <Store></Store>
      <Register></Register>
   </Originator>
   <MessageType>
      <Function>Authorization</Function>
      <Modifier>Void</Modifier>
      <AccountType>CreditCard</AccountType>
      <StoreAndForward> false </StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber></TransactionNumber>
      <SequenceNumber></SequenceNumber>
      <TransactionDateTime></TransactionDateTime>
      <OriginalMessage>
         <Request>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber> </SequenceNumber>
            <TransactionDateTime></TransactionDateTime>
            <CardDetail>
               <InputMode> </InputMode>
               <CardPresent> </CardPresent>
               <AccountNumber></AccountNumber>
               <ExpiryDate></ExpiryDate>
            </CardDetail>
            <RequestedAmount Value="" CurrencyCode="" />
            <OfflineResponse>
               <ExternalReferenceNumber> </ExternalReferenceNumber>
               <ApprovalCode> </ApprovalCode >
               <ApprovedAmount Value="" CurrencyCode="" />
            </OfflineResponse>
         </Request>
      </OriginalMessage>
   </Request>
</FinancialMessage>
```

SAF type messages

Both Regular and Reference type messages can be approved offline manually or by phone. These are known as SAF type messages. The SAF messages are similar to Regular and Reference type message except that the StoreAndForward flag is set to true, and it has the OfflineResponse element in the Request.

Note that whenever SAF messages are referenced, they always have the OfflineResponse in the Request. An example of this is a void for an SAF original message. The following are examples of SAF messages. These messages are similar to previous examples except that they have been approved offline.

Example 1

```
<FinancialMessage>
   <Originator>
      <Chain></Chain>
      <Store></Store>
      <Register></Register>
   </Originator>
   <MessageType>
      <Function>Authorization</Function>
      <Modifier>Original</Modifier>
      <AccountType>CreditCard</AccountType>
      <StoreAndForward> true </StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber></TransactionNumber>
      <SequenceNumber></SequenceNumber>
      <TransactionDateTime></TransactionDateTime>
      <CardDetail>
         <InputMode> </InputMode>
         <CardPresent> </CardPresent>
         <AccountNumber></AccountNumber>
         <ExpiryDate> </ExpiryDate>
      </CardDetail>
      <RequestedAmount Value="" CurrencyCode=" " />

      <OfflineResponse>
         <ExternalReferenceNumber> </ExternalReferenceNumber>
         <ApprovalCode> </ApprovalCode >
         <ApprovedAmount Value="" CurrencyCode="" />
      </OfflineResponse>
   </Request>
</FinancialMessage>
```

Example 2

```
<FinancialMessage>
   <Originator>
      <Chain></Chain>
      <Store></Store>
      <Register></Register>
   </Originator>
   <MessageType>
       <Function>Authorization</Function>
       <Modifier>Void</Modifier>
```

```
          <AccountType>CreditCard</AccountType>
          <StoreAndForward> true </StoreAndForward>
          <AdditionalModifier>Original</AdditionalModifier>
        </MessageType>
        <Request>
          <TransactionNumber></TransactionNumber>
          <SequenceNumber></SequenceNumber>
          <TransactionDateTime></TransactionDateTime>
          <OriginalMessage>
              <Request>
                  <TransactionNumber> </TransactionNumber>
                  <SequenceNumber> </SequenceNumber>
                  <TransactionDateTime></TransactionDateTime>
                      <CardDetail>
                          <InputMode> </InputMode>
                          <CardPresent> </CardPresent>
                          <AccountNumber></AccountNumber>
                          <ExpiryDate></ExpiryDate>
                      </CardDetail>
                      <RequestedAmount Value="" CurrencyCode="" />
                  </Request>
                  <Response>
                      <ExternalReferenceNumber> </ExternalReferenceNumber>
                      <ApprovalCode></ApprovalCode >
                      <ApprovedAmount Value="" CurrencyCode="" />
                  </Response>
              </OriginalMessage>

          <OfflineResponse>
                      <ExternalReferenceNumber> </ExternalReferenceNumber>
                      <ApprovalCode> </ApprovalCode >
                      <ApprovedAmount Value="" CurrencyCode="" />
          </OfflineResponse>
        </Request>
</FinancialMessage>
```

Example 3

```
<FinancialMessage>
   <Originator>
      <Chain></Chain>
      <Store></Store>
      <Register></Register>
   </Originator>
   <MessageType>
      <Function>Authorization</Function>
      <Modifier>Void</Modifier>
      <AccountType>CreditCard</AccountType>
      <StoreAndForward> true </StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
```

```
    </MessageType>
    <Request>
        <TransactionNumber></TransactionNumber>
        <SequenceNumber></SequenceNumber>
        <TransactionDateTime></TransactionDateTime>
        <OriginalMessage>
            <Request>
                <TransactionNumber> </TransactionNumber>
                <SequenceNumber> </SequenceNumber>
                <TransactionDateTime></TransactionDateTime>
                <CardDetail>
                    <InputMode> </InputMode>
                    <CardPresent> </CardPresent>
                    <AccountNumber></AccountNumber>
                    <ExpiryDate></ExpiryDate>
                </CardDetail>
                <RequestedAmount Value="" CurrencyCode="" />
                <OfflineResponse>
                    <ExternalReferenceNumber> </ExternalReferenceNumber>
                    <ApprovalCode> </ApprovalCode >
                    <ApprovedAmount Value="" CurrencyCode="" />
                </OfflineResponse>
            </Request>
        </OriginalMessage>

        <OfflineResponse>
            <ExternalReferenceNumber> </ExternalReferenceNumber>
            <ApprovalCode> </ApprovalCode >
            <ApprovedAmount Value="" CurrencyCode="" />
        </OfflineResponse>
    </Request>
</FinancialMessage>
```

Reversal type messages

Any type of message can be reversed including Regular, Reference and SAF type messages.

A reversal message will have the OriginalMessage element which refers to the original message to be reversed. The OriginalMessage element will only contain the Request element and not the Response element or the OfflineResponse element.

The Modifier will be set to Reversal for all types of messages except for voids which will be set to ReversalVoid. If a SAF message is being reversed then the StoreAndForward message will be set to true.

```
<FinancialMessage>
    <Originator>
        <Chain></Chain>
        <Store></Store>
        <Register></Register>
    </Originator>
    <MessageType>
```

```
            <Function>Authorization</Function>
            <Modifier>Reversal {or ReversalVoid}</Modifier>
            <AccountType>CreditCard</AccountType>
            <StoreAndForward> false </StoreAndForward>
            <AdditionalModifier>Original</AdditionalModifier>
        </MessageType>
        <Request>
               <TransactionNumber> </TransactionNumber>
               <SequenceNumber> </SequenceNumber>
               <TransactionDateTime></TransactionDateTime>
               <OriginalMessage>
                  <Request>
                     <TransactionNumber> </TransactionNumber>
                     <SequenceNumber></SequenceNumber>
                     <TransactionDateTime></TransactionDateTime>
                     <CardDetail>
                        <InputMode> </InputMode>
                        <CardPresent> </CardPresent>
                        <AccountNumber></AccountNumber>
                        <ExpiryDate></ExpiryDate>
                     </CardDetail>
                     <RequestedAmount Value="" CurrencyCode="" />
                  </Request>
               </OriginalMessage>
        </Request>
    </FinancialMessage>
```

Stored value card-related messages

The following stored value card transactions types are supported:

- Activate
- Deactivate
- Redeem
- Return
- Reload
- Balance Inquiry
- Replacement
- Cashout
- PreAuthorization
- PostAuthorization
- StoreAndForward
- Void
- Reversal

Activate request message

The following is a stored value card activate request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store> </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> StoredValueCard </AccountType>
      <Function> Activate  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward >  False </StoreAndForward>
   </MessageType>

   <Request >
      <TransactionNumber> </TransactionNumber>
      <SequenceNumber>  </SequenceNumber>
      <TransactionDateTime> </TransactionDateTime>

      <CardDetail> </CardDetail>
      <RequestedAmount>  {Amount Aggregate} </RequestedAmount> [Optional]

   </Request>

</FinancialMessage >
```

Activate response message

The following is a stored value card activate response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   </MessageType>
      <AccountType> StoredValueCard </AccountType>
      <Function> Activate  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request> </Request>

   <Response>
      <ActionCode ID=”” Desc=””/>
```

```
            <ApprovalCode> </ApprovalCode>

            <ExternalReferenceNumber> </ExternalReferenceNumber>

            <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
            <BalanceAmount> {Amount Aggregate} </BalanceAmount>

            <ExpiryDate > </ExpiryDate> [Optional]
            <EffectiveDate> </EffectiveDate> [Optional]

      </Response>
</FinancialMessage >
```

Redeem request message

The following is a stored value card redeem request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> StoredValueCard </AccountType>
      <Function> Authorization  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request >
      <TransactionNumber> </TransactionNumber>
      <SequenceNumber>  </SequenceNumber>
      <TransactionDateTime> </TransactionDateTime>

      <CardDetail> </CardDetail>
      <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
   </Request>
</FinancialMessage >
```

Redeem response message

The following is a stored value card redeem response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>
```

```
    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Authorization  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request> </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <BalanceAmount> {Amount Aggregate} </BalanceAmount>

        <ExpiryDate > </ExpiryDate> [Optional]

    </Response>

</FinancialMessage >
```

Note: Approved amount should be checked with the requested amount as in some cases they could differ. When voiding this transaction, the authorized amount should be sent and not the requested amount.

Return request message

The following is a stored value card return request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Return  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>
```

```
        <CardDetail> </CardDetail>
        <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
    </Request>
</FinancialMessage >
```

Return response message

The following is a stored value card return response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Return  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request> </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <BalanceAmount> {Amount Aggregate} </BalanceAmount>

        <ExpiryDate > </ExpiryDate> [Optional]

    </Response>

</FinancialMessage >
```

Reload request message

The following is a stored value card reload request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>
```

```
    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Payment </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <TransactionNumber>  </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <CardDetail> </CardDetail>
        <RequestedAmount>  {Amount Aggregate} </RequestedAmount>

    </Request>

</FinancialMessage >
```

Reload response message

The following is a stored value card reload response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain> </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Payment </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request> </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <BalanceAmount> {Amount Aggregate} </BalanceAmount>
        <ExpiryDate > </ExpiryDate> [Optional]

    </Response>
```

```
</FinancialMessage >
```

Balance Inquiry request message

The following is a stored value card balance inquiry request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> StoredValueCard </AccountType>
      <Function> BalanceInquiry </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request >
      <TransactionNumber> </TransactionNumber>
      <SequenceNumber>  </SequenceNumber>
      <TransactionDateTime> </TransactionDateTime>

      <CardDetail> </CardDetail>

   </Request>

</FinancialMessage >
```

Balance Inquiry response message

The following is a stored value card balance inquiry response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> StoredValueCard </AccountType>
      <Function> BalanceInquiry </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request> </Request>

   <Response>
```

```
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <BalanceAmount> {Amount Aggregate} </BalanceAmount>
        <ExpiryDate > </ExpiryDate> [Optional]

    </Response>

</FinancialMessage >
```

Replacement request message

The following is a stored value card replacement request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Replace </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <TransactionNumber>  </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <CardDetail Name="New"/>
        <CardDetail Name="Old"/>

    </Request>

</FinancialMessage >
```

Replacement response message

The following is a stored value card replacement response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>
```

```
    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Replace </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request> </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <BalanceAmount> {Amount Aggregate} </BalanceAmount>

        <CardDetail Name="New"/>
        <ExpiryDate > </ExpiryDate> [Optional]

    </Response>

</FinancialMessage >
```

Note: Balance and expiry date will be of the new card.

Cash-out request message

The following is a stored value card cash-out request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> CashOut </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>
```

```
        <CardDetail> </CardDetail>

    </Request>
</FinancialMessage >
```

Cash-out response message

The following is a stored value card cash-out response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> CashOut </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request> </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <BalanceAmount> {Amount Aggregate} </BalanceAmount>
        <ExpiryDate > </ExpiryDate> [Optional]

    </Response>

</FinancialMessage >
```

Notes:

- Approved Amount will contain the cashed out amount.
- Balance would be 0.00.

Void request message

Void requests can be sent only for the following transaction types:

- Activate
- Deactivate
- Redeem
- Return

- Reload
- CashOut
- PostAuthorization

The following is a stored value card void request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> StoredValueCard </AccountType>
      <Function> Activate / Authorization / Payment / CashOut </Function>
      <Modifier> Void </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request >
      <TransactionNumber> </TransactionNumber>
      <SequenceNumber>  </SequenceNumber>
      <TransactionDateTime> </TransactionDateTime>

      <OriginalMessage>
         <TransactionNumber> </TransactionNumber>
         <SequenceNumber>  </SequenceNumber>
         <TransactionDateTime> </TransactionDateTime>
         <CardDetail>  </CardDetail>

         <ExternalReferenceNumber> </ExternalReferenceNumber>
         <ApprovalCode> </ approvalCode >
         <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
      </OriginalMessage>


   </Request>
</FinancialMessage >
```

Note: All data from the original approved transaction request should be in the OriginalMessage element. ExternalReferenceNumber is from the original approved transaction response.

Void response message

The following is a stored value card void response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
```

```
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Activate / Authorization / Payment / CashOut </Function>
        <Modifier> Void </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <BalanceAmount> {Amount Aggregate} </BalanceAmount>
        <ExpiryDate > </ExpiryDate> [Optional]

    </Response>

</FinancialMessage >
```

Reversal request message

Only the following types of financial transactions can be reversed:

- Activate
- Deactivate
- Redeem
- Return
- Reload
- Replacement
- Cashout
- PreAuthorization
- PostAuthorization
- StoreAndForward
- Void

The following is a stored value card reversal request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
```

```
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Activate / Redeem / Reload / CashOut </Function>
        <Modifier> Reversal </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <OriginalMessage>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>
            <CardDetail>  </CardDetail>
            <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
            <ExternalReferenceNumber> </ExternalReferenceNumber>
        </OriginalMessage>
    </Request>

</FinancialMessage >
```

Note: All data should be that of the original approved transaction request.

Reversal response message

The following is a stored value card reversal response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Activate / Redeem / Reload / CashOut </Function>
        <Modifier> Reversal </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request>
    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>
```

```
        <ApprovedAmount> {Amount Aggregate}   </ApprovedAmount>
        <BalanceAmount> {Amount Aggregate} </BalanceAmount>
        <ExpiryDate > </ExpiryDate> [Optional]

    </Response>
</FinancialMessage >
```

Store and Forward request message

Store and forward messages are used for locally (force) approved and voice approved transactions. The following is a stored value card store and forward request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>
    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Authorization </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  True </StoreAndForward>
        <AdditionalModifier>Original</AdditionalModifier>
    </MessageType>
    <Request>
        <!-- format similar to regular txn types -->
        <OfflineResponse>
            {Response Aggregate}
        </OfflineResponse>
    </Request>
</FinancialMessage>
```

Store and Forward response message

Store and forward messages are used for locally (force) approved and voice approved transactions. The following is a stored value card store and forward response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>
    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Authorization </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  True </StoreAndForward>
        <AdditionalModifier>Original</AdditionalModifier>
```

```
    </MessageType>
    <Request>
        <!-- format similar to regular txn types -->
        <OfflineResponse>
            {Response Aggregate}
        </OfflineResponse>
    </Request>
    <Response>
    </Response>
</FinancialMessage>
```

Void Store and Forward request with online original request

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>
    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> Activate </Function>
        <Modifier> Void </Modifier>
        <StoreAndForward>  False </StoreAndForward>
        <AdditionalModifier>Original</AdditionalModifier>
    </MessageType>
    <Request>
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>
        <OriginalMessage>
            <Request>
                <TransactionNumber> </TransactionNumber>
                <SequenceNumber>  </SequenceNumber>
                <TransactionDateTime> </TransactionDateTime>
                <CardDetail>  </CardDetail>
            </Request>
            <Response>
                </ExternalReferenceNumber>
                </ApprovalCode>
                </ApprovedAmount>
            </Response>
        </OriginalMessage>
        <OfflineResponse>
            {Response Aggregate}
        </OfflineResponse>
    </Request>
</FinancialMessage >
```

Void Store and Forward response with online original request

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>
   <MessageType>
      <AccountType> StoredValueCard </AccountType>
      <Function> Activate </Function>
      <Modifier> Void </Modifier>
      <StoreAndForward>  False </StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber> </TransactionNumber>
      <SequenceNumber>  </SequenceNumber>
      <TransactionDateTime> </TransactionDateTime>
      <OriginalMessage>
         <Request>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>
            <CardDetail>  </CardDetail>
         </Request>
         <Response>
            </ExternalReferenceNumber>
            </ApprovalCode>
            </ApprovedAmount>
         </Response>
      </OriginalMessage>
      <OfflineResponse>
         {Response Aggregate}
      </OfflineResponse>
   </Request>
   <Response>
   </Response>
</FinancialMessage >
```

Void Store and Forward request with offline original request

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>
   <MessageType>
      <AccountType> StoredValueCard </AccountType>
```

```
            <Function> Activate </Function>
            <Modifier> Void </Modifier>
            <StoreAndForward>  False </StoreAndForward>
            <AdditionalModifier>Original</AdditionalModifier>
        </MessageType>
        <Request>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>
            <OriginalMessage>
                <Request>
                    <TransactionNumber> </TransactionNumber>
                    <SequenceNumber>  </SequenceNumber>
                    <TransactionDateTime> </TransactionDateTime>
                    <CardDetail>  </CardDetail>
                    <OfflineResponse>
                    </OfflineResponse>
                </Request>
                <Response>
                </Respoonse>
            </OriginalMessage>
            <OfflineResponse>
                {Response Aggregate}
            </OfflineResponse>
        </Request>
    </FinancialMessage >
```

Void Store and Forward response with offline original request

```
    <FinancialMessage>
        <Originator>
            <Chain>  </Chain>
            <Store>  </Store>
            <Register>  </Register>
        </Originator>
        <MessageType>
            <AccountType> StoredValueCard </AccountType>
            <Function> Activate </Function>
            <Modifier> Void </Modifier>
            <StoreAndForward>  False </StoreAndForward>
            <AdditionalModifier>Original</AdditionalModifier>
        </MessageType>
        <Request>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>
            <OriginalMessage>
                <Request>
                    <TransactionNumber> </TransactionNumber>
                    <SequenceNumber>  </SequenceNumber>
```

```
                <TransactionDateTime> </TransactionDateTime>
                <CardDetail>  </CardDetail>
                <OfflineResponse>
                </OfflineResponse>
            </Request>
            <Response>
            </Respoonse>
        </OriginalMessage>
        <OfflineResponse>
        </OfflineResponse>
    </Request>
    <Response>
    </Response>
</FinancialMessage >
```

Pre Authorization request message

The following is a stored value card pre authorization request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> PreAuthorization  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <CardDetail> </CardDetail>
        <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
    </Request>
</FinancialMessage >
```

Pre-authorization response message

The following is a stored value card pre authorization response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
```

```
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> PreAuthorization  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request>
    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <BalanceAmount> {Amount Aggregate} </BalanceAmount>

        <ExpiryDate > </ExpiryDate> [Optional]

    </Response>
</FinancialMessage >
```

Post authorization request message

Also called as Pre-Authorization Completion request message. The following is a stored value card post authorization request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> PostAuthorization </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request>
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <RequestedAmount> </RequestedAmount>
```

```
        <OriginalMessage>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>
            <CardDetail>  </CardDetail>

            <ExternalReferenceNumber> </ExternalReferenceNumber>
            <ApprovalCode> </ApprovalCode>
            <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        </OriginalMessage>
    </Request>
</FinancialMessage >
```

Note: All data from the original approved transaction request should be in the OriginalMessage element. ExternalReferenceNumber is from the original approved transaction response

Post authorization response message

Also called as Pre Authorization Completion response message. The following is a stored value card post authorization response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> StoredValueCard </AccountType>
        <Function> PostAuthorization </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request>
    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <BalanceAmount> {Amount Aggregate} </BalanceAmount>
        <ExpiryDate > </ExpiryDate> [Optional]
    </Response>
</FinancialMessage >
```

Deactivate request message

The following is a stored value card deactivate request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store> </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> StoredValueCard </AccountType>
      <Function> Deactivate  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward >  False </StoreAndForward>
   </MessageType>

   <Request >
      <TransactionNumber> </TransactionNumber>
      <SequenceNumber>  </SequenceNumber>
      <TransactionDateTime> </TransactionDateTime>

      <CardDetail> </CardDetail>
      <RequestedAmount>  {Amount Aggregate} </RequestedAmount> [Optional]

   </Request>
</FinancialMessage >
```

Deactivate response message

The following is a stored value card deactivate response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   </MessageType>
      <AccountType> StoredValueCard </AccountType>
      <Function> Deactivate  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request> </Request>

   <Response>
      <ActionCode ID="" Desc=""/>
      <ApprovalCode> </ApprovalCode>
```

```
        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <BalanceAmount> {Amount Aggregate} </BalanceAmount>
    </Response>

</FinancialMessage >
```

Credit card-related messages

The following credit card transaction types are supported:

- Sale
- Return
- PreAuthorization
- PostAuthorization
- StoreAndForward
- Void
- Reversal

Sale request message

The following is a credit card sale request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> CreditCard </AccountType>
        <Function> Authorization  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <CardDetail> </CardDetail>
        <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
    </Request>
</FinancialMessage >
```

Sale response message

The following is a credit card sale response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> CreditCard </AccountType>
      <Function> Authorization  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request> </Request>

   <Response>
      <ActionCode ID="" Desc=""/>
      <ApprovalCode> </ApprovalCode>

      <ExternalReferenceNumber> </ExternalReferenceNumber>

      <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>

      <ExpiryDate > </ExpiryDate> [Optional]

   </Response>

</FinancialMessage >
```

Note: Approved amount should be checked with the requested amount as in some cases they could differ. When voiding this transaction, the authorized amount should be sent and not the requested amount.

Return request message

The following is a credit card return request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> CreditCard </AccountType>
```

```
        <Function> Return  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <CardDetail> </CardDetail>
        <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
    </Request>
</FinancialMessage >
```

Return response message

The following is a credit card return response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> CreditCard </AccountType>
        <Function> Return  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request> </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>

        <ExpiryDate > </ExpiryDate> [Optional]

    </Response>
</FinancialMessage >
```

Void request message

Void requests can be sent only for the following transaction types:

- Sale
- Return
- PostAuthorization

The following is a credit card void request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> CreditCard </AccountType>
      <Function> Authorization / Return / PostAuthorizaion </Function>
      <Modifier> Void </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request >
      <TransactionNumber> </TransactionNumber>
      <SequenceNumber>  </SequenceNumber>
      <TransactionDateTime> </TransactionDateTime>

      <OriginalMessage>
         <TransactionNumber> </TransactionNumber>
         <SequenceNumber>  </SequenceNumber>
         <TransactionDateTime> </TransactionDateTime>
         <CardDetail>  </CardDetail>

         <ExternalReferenceNumber> </ExternalReferenceNumber>
         <ApprovalCode> </ APprovalCode >
         <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
      </OriginalMessage>
   </Request>
</FinancialMessage >
```

Note: All data from the original approved transaction request should be in the OriginalMessage element. The ExternalReferenceNumber is from the original approved transaction response.

Void response message

The following is a credit card void response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>
```

```
    <MessageType>
        <AccountType> CreditCard </AccountType>
        <Function> Authorization/ Return / PostAuthorizaion </Function>
        <Modifier> Void </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <ExpiryDate > </ExpiryDate> [Optional]
    </Response>
</FinancialMessage >
```

Reversal request message

Only the following types of financial transactions can be reversed:

- Sale
- Return
- PreAuthorization
- PostAuthorization
- StoreAndForward
- Void

The following is a credit card reversal request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> CreditCard </AccountType>
        <Function> [see above] </Function>
        <Modifier> Reversal </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <OriginalMessage>
            <TransactionNumber> </TransactionNumber>
```

```
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>
            <CardDetail>  </CardDetail>
            <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
            <ExternalReferenceNumber> </ExternalReferenceNumber>
         </OriginalMessage>
   </Request>
</FinancialMessage >
```

Note: All data should be that of the original approved transaction request.

Reversal response message

The following is a credit card reversal response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> CreditCard </AccountType>
      <Function> [See above] </Function>
      <Modifier> Reversal </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request>
   </Request>

   <Response>
      <ActionCode ID="" Desc=""/>
      <ApprovalCode> </ApprovalCode>

      <ExternalReferenceNumber> </ExternalReferenceNumber>

      <ApprovedAmount> {Amount Aggregate}   </ApprovedAmount>
      <ExpiryDate > </ExpiryDate> [Optional]

   </Response>
</FinancialMessage >
```

Store and forward request message

Store and forward messages are used for locally (force) approved and voice approved transactions. The following is a credit card store and forward request message containing all the elements:

```
<FinancialMessage>
   <Originator>
```

```
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> CreditCard </AccountType>
        <Function> Authorization </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  True </StoreAndForward>
    </MessageType>

    <Request >
        <OriginalMessage>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>
            <CardDetail>  </CardDetail>
            <RequestedAmount> </RequestedAmount>

            <ExternalReferenceNumber> </ExternalReferenceNumber>
            <ApprovalCode> </ApprovalCode>
            <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        </OriginalMessage>
    </Request>
</FinancialMessage >
```

Note: All data from the original approved transaction request should be in the OriginalMessage element. ExternalReferenceNumber is from the original approved transaction response

Store and forward response message

Store and forward messages are used for locally (force) approved and voice approved transactions. The following is a credit card store and forward response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> CreditCard </AccountType>
        <Function> Authorization </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  true </StoreAndForward>
    </MessageType>

    <Request>
```

```
    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>
        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <ExpiryDate > </ExpiryDate> [Optional]
    </Response>
</FinancialMessage >
```

Pre-authorization request message

The following is a credit card pre authorization request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> CreditCard </AccountType>
        <Function> PreAuthorization  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <CardDetail> </CardDetail>
        <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
    </Request>
</FinancialMessage >
```

Pre-authorization response message

The following is a credit card pre authorization response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
```

```
        <AccountType> CreditCard </AccountType>
        <Function> PreAuthorization  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request>
    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>
        <ExternalReferenceNumber> </ExternalReferenceNumber>
        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <ExpiryDate > </ExpiryDate> [Optional]
    </Response>
</FinancialMessage >
```

Post authorization request message

Also called as Pre-authorization Completion request message. The following is a credit card post authorization request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> CreditCard </AccountType>
        <Function> PostAuthorization </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request>
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <RequestedAmount> </RequestedAmount>

        <OriginalMessage>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>
            <CardDetail>  </CardDetail>

            <ExternalReferenceNumber> </ExternalReferenceNumber>
```

```
            <ApprovalCode> </ApprovalCode>
            <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        </OriginalMessage>
    </Request>
</FinancialMessage >
```

Note: All data from the original approved transaction request should be in the OriginalMessage element. ExternalReferenceNumber is from the original approved transaction response

Post authorization response message

Also called as Pre-authorization Completion response message. The following is a credit card post authorization response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> CreditCard </AccountType>
        <Function> PostAuthorization </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request>
    </Request>

    <Response>
    <ActionCode ID="" Desc=""/>
    <ApprovalCode> </ApprovalCode>

    <ExternalReferenceNumber> </ExternalReferenceNumber>

    <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
    <ExpiryDate > </ExpiryDate> [Optional]
    </Response>
</FinancialMessage >
```

Debit card-related messages

The following credit card transaction types are supported:

- Sale
- Return
- Balance Inquiry
- Void

- Reversal

Sale request message

The following is a debit card sale request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
</Originator>

<MessageType>
      <AccountType> DebitCard </AccountType>
      <Function> Authorization  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request >
      <TransactionNumber> </TransactionNumber>
      <SequenceNumber>  </SequenceNumber>
      <TransactionDateTime> </TransactionDateTime>

      <CardDetail> </CardDetail>
      <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
      <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]

      <PinData> </PinData>
      <SecurityControlInfo></SecurityControlInfo>
      <MAC> </MAC> [Optional]
   </Request>
</FinancialMessage >
```

Sale response message

The following is a debit card sale response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> DebitCard </AccountType>
      <Function> Authorization  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
```

```
    </MessageType>

    <Request> </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]
    </Response>
</FinancialMessage >
```

Return request message

The following is a debit card return request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> DebitCard </AccountType>
        <Function> Return  </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <CardDetail> </CardDetail>
        <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
        <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]

        <PinData> </PinData>
        <SecurityControlInfo></SecurityControlInfo>
        <MAC> </MAC> [Optional]
    </Request>
</FinancialMessage >
```

Return response message

The following is a debit card return response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> DebitCard </AccountType>
      <Function> Return  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request> </Request>

   <Response>
      <ActionCode ID="" Desc=""/>
      <ApprovalCode> </ApprovalCode>

      <ExternalReferenceNumber> </ExternalReferenceNumber>

      <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
      <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]
   </Response>
</FinancialMessage >
```

Balance inquiry request message

The following is a debit card balance inquiry request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> DebitCard </AccountType>
      <Function> BalanceInquiry </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request>
      <TransactionNumber> </TransactionNumber>
      <SequenceNumber>  </SequenceNumber>
      <TransactionDateTime> </TransactionDateTime>

      <CardDetail> </CardDetail>
```

```
        <PinData> </PinData>
        <SecurityControlInfo></SecurityControlInfo>
        <MAC> </MAC> [Optional]
    </Request>
</FinancialMessage >
```

Balance inquiry response message

The following is a debit card balance inquiry response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> DebitCard </AccountType>
        <Function> BalanceInquiry </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request> </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <BalanceAmount> {Amount Aggregate} </BalanceAmount>
    </Response>
</FinancialMessage >
```

Void request message

Void requests can be sent only for the following transaction types:

- Sale
- Return

The following is a debit card void request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>
```

```
    <MessageType>
        <AccountType> DebitCard </AccountType>
        <Function> Authorization / Return / PostAuthorizaion </Function>
        <Modifier> Void </Modifier>
        <StoreAndForward>  False </StoreAndForward>
    </MessageType>

    <Request >
        <TransactionNumber> </TransactionNumber>
        <SequenceNumber>  </SequenceNumber>
        <TransactionDateTime> </TransactionDateTime>

        <PinData> </PinData>
        <SecurityControlInfo></SecurityControlInfo>
        <MAC> </MAC> [Optional]

        <OriginalMessage>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>
            <CardDetail>  </CardDetail>
            <CashBackAmount> {Amount Aggregate} </CashBackAmount>
            [Optional]

            <ExternalReferenceNumber> </ExternalReferenceNumber>
            <ApprovalCode> </ ApprovalCode >
            <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        </OriginalMessage>
    </Request>
</FinancialMessage >
```

Note: All data from the original approved transaction request should be in the OriginalMessage element. ExternalReferenceNumber is from the original approved transaction response.

Void response message

The following is a debit card void response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> DebitCard </AccountType>
        <Function> Authorization / Return / PostAuthorizaion </Function>
        <Modifier> Void </Modifier>
        <StoreAndForward>  False </StoreAndForward>
```

```
    </MessageType>

    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <ExpiryDate > </ExpiryDate> [Optional]
        <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]
    </Response>
</FinancialMessage >
```

Reversal request message

Only the following types of financial transactions can be reversed:

- Sale
- Return
- StoreAndForward
- Void

The following is a debit card reversal request message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> DebitCard </AccountType>
        <Function> [see above] </Function>
        <Modifier> Reversal </Modifier>
        <StoreAndForward> False </StoreAndForward>
    </MessageType>

    <Request>
        <OriginalMessage>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>

            <CardDetail>  </CardDetail>
            <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
            <CashBackAmount> {Amount Aggregate} </CashBackAmount>
            [Optional]
```

```
            <PinData> </PinData>
            <SecurityControlInfo></SecurityControlInfo>
            <MAC> </MAC> [Optional]

            <ExternalReferenceNumber> </ExternalReferenceNumber>
         </OriginalMessage>
      </Request>
</FinancialMessage >
```

Note: All data should be that of the original approved transaction request.

Reversal response message

The following is a debit card reversal response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> DebitCard </AccountType>
      <Function> [See above] </Function>
      <Modifier> Reversal </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request>
   </Request>

   <Response>
      <ActionCode ID="" Desc=""/>
      <ApprovalCode> </ApprovalCode>

      <ExternalReferenceNumber> </ExternalReferenceNumber>

      <ApprovedAmount> {Amount Aggregate}   </ApprovedAmount>
      <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]

   </Response>
</FinancialMessage >
```

Store and forward request message

Store and forward messages are used for locally (force) approved and voice approved transactions. The following is a debit card store and forward request message containing all the elements:

```
<FinancialMessage>
```

```
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> DebitCard </AccountType>
        <Function> Authorization </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  True </StoreAndForward>
    </MessageType>

    <Request >
        <OriginalMessage>
            <TransactionNumber> </TransactionNumber>
            <SequenceNumber>  </SequenceNumber>
            <TransactionDateTime> </TransactionDateTime>
            <CardDetail>  </CardDetail>
            <RequestedAmount> </RequestedAmount>

            <CashBackAmount> {Amount Aggregate} </CashBackAmount>
            [Optional]
            <PinData> </PinData>
            <SecurityControlInfo></SecurityControlInfo>
            <MAC> </MAC> [Optional]

            <ExternalReferenceNumber> </ExternalReferenceNumber>
            <ApprovalCode> </ApprovalCode>
            <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        </OriginalMessage>
    </Request>
</FinancialMessage >
```

Note: All data from the original approved transaction request should be in the OriginalMessage element. The ExternalReferenceNumber is from the original approved transaction response

Store and forward response message

Store and forward messages are used for locally (force) approved and voice approved transactions. The following is a debit card store and forward response message containing all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
```

```
        <AccountType> DebitCard </AccountType>
        <Function> Authorization </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  true </StoreAndForward>
    </MessageType>

    <Request>
    </Request>

    <Response>
        <ActionCode ID="" Desc=""/>
        <ApprovalCode> </ApprovalCode>

        <ExternalReferenceNumber> </ExternalReferenceNumber>

        <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
        <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]
        <ExpiryDate > </ExpiryDate> [Optional]
    </Response>
</FinancialMessage >
```

PinPad key exchange request message

PinPad key exchange request message contains all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> DebitCard </AccountType>
        <Function> PinPadKeyExchange </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  false </StoreAndForward>
        <AdditionalModifier> Original/PinPadFormattedDataExchange
        </AdditionalModifier>
    </MessageType>

    <Request>
        <PinPadInfo></PinPadInfo>
    </Request>

</FinancialMessage >
```

PinPad key exchange response message

PinPad key exchange response message contains all the elements:

```
<FinancialMessage>
```

```
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> DebitCard </AccountType>
        <Function> PinPadKeyExchange </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  false </StoreAndForward>
        <AdditionalModifier> Original/PinPadFormattedDataExchange
        </AdditionalModifier>
    </MessageType>

    <Request>
        <PinPadInfo></PinPadInfo>
    </Request>

    <Response>
        <PinPadInfo></PinPadInfo>
    </Response>

</FinancialMessage >
```

PinPad close batch request message

PinPad close batch request message contains all the elements:

```
<FinancialMessage>
    <Originator>
        <Chain>  </Chain>
        <Store>  </Store>
        <Register>  </Register>
    </Originator>

    <MessageType>
        <AccountType> DebitCard </AccountType>
        <Function> PinPadCloseBatch </Function>
        <Modifier> Original </Modifier>
        <StoreAndForward>  false </StoreAndForward>
        <AdditionalModifier> Original/PinPadFormattedDataExchange
        </AdditionalModifier>
    </MessageType>

    <Request>
        <PinPadInfo></PinPadInfo>
    </Request>

</FinancialMessage >
```

PinPad close batch response message

PinPad close batch request message contains all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>
   <MessageType>
      <AccountType> DebitCard </AccountType>
      <Function> PinPadCloseBatch </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  false </StoreAndForward>
      <AdditionalModifier> Original/PinPadFormattedDataExchange
      </AdditionalModifier>
   </MessageType>
   <Request>
      <PinPadInfo></PinPadInfo>
   </Request>

   <Response>
      <PinPadInfo></PinPadInfo>
   </Response>

</FinancialMessage >
```

Check verification/guarantee-related messages

The following check transaction types are supported:

- Authorization (Verification / Guarantee)

Authorization request message

The following is a Check authorization request message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> Cheque </AccountType>
      <Function> Authorization  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>
```

```
<Request >
    <TransactionNumber> </TransactionNumber>
    <SequenceNumber>  </SequenceNumber>
    <TransactionDateTime> </TransactionDateTime>

    <CardDetail>
        <InputMode> MICR </InputMode>
        <CardPresent> true </CardPresent>

        <AccountNumber></AccountNumber>
        <ExpiryDate> 200202 </ExpiryDate>
        <ChequeNumber> </ChequeNumber>
        <BankNumber> </BankNumber>
        <MICRNumber> </MICRNumber> [Optional]

        <VerificationMethod Type="">
            <Number> </Number>
            <State> </State> [Optional]
            <DateOfBirth> </DateOfBirth> [Optional]
            <ZipCode> </ZipCode> [Optional]
        </VerificationMethod

        <EffectiveDate> </EffectiveDate> [Optional]
    </CardDetail>

    <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
    <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]
</Request>
</FinancialMessage >
```

Notes:

- The Effective Date is the date on the check (optional)
- ExpiryDate is a mandatory field but will not be used for check authorizations. A valid date must be entered and it's suggested that it be set to the current date
- Transnet will determine if check verification or check guarantee is to be done and will also determine the check processor
- The CardDetail will hold the MICR information
- VerificationMethod Element must be present and as much information as possible should be provided
- The following are the verification method types:
  - Drivers License
  - SSN
  - PlasticCard
  - MICR
- The CardDetail must always contain the AccountNumber, ChequeNumber and BankNumber information of the check. The optional MICRNumber is the complete MICR number.

Authorization response message

The following is a check authorization response message containing all the elements:

```
<FinancialMessage>
   <Originator>
      <Chain>  </Chain>
      <Store>  </Store>
      <Register>  </Register>
   </Originator>

   <MessageType>
      <AccountType> Cheque </AccountType>
      <Function> Authorization  </Function>
      <Modifier> Original </Modifier>
      <StoreAndForward>  False </StoreAndForward>
   </MessageType>

   <Request> </Request>

   <Response>
      <ActionCode ID="" Desc=""/>
      <ApprovalCode> </ApprovalCode>

      <ExternalReferenceNumber> </ExternalReferenceNumber>

      <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
      <CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]
   </Response>
</FinancialMessage >
```

Cheque Transfer request message

The following is a Cheque transfer request message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> Cheque </AccountType>
<Function> Transfer  </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request >
<TransactionNumber> </TransactionNumber>
```

```
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>

<CardDetail Name=”To”>
<CardDetail Name=”From”>


<RequestedAmount>  {Amount Aggregate} </RequestedAmount>
<CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]

<LanguageCode><LanguageCode> [Optional]

</Request>
</FinancialMessage >
```

Cheque Transfer response message

The following is a Cheque transfer response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> Cheque </AccountType>
<Function> Transfer  </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request> </Request>

<Response>
<ActionCode ID=”” Desc=””/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>

<ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
<CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]

<LanguageCode><LanguageCode> [Optional]
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]
</Response>
```

```
</FinancialMessage >
```

Electronic Benefits Transfer (EBT)-related messages

Purchase Only request message

The following is a EBT purchase request message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> Authorization  </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request >
<TransactionNumber> </TransactionNumber>
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>

<CardDetail> </CardDetail>
<RequestedAmount>  {Amount Aggregate} </RequestedAmount>

<PinData> </PinData>
<SecurityControlInfo></SecurityControlInfo>
<MAC> </MAC> [Optional]
<PinPadInfo></PinPadInfo> [Optional]

<LanguageCode><LanguageCode> [Optional]
</Request>
</FinancialMessage >
```

Purchase Only response message

The following is a EBT purchase response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>
```

```
<MessageType>
<AccountType> EBT </AccountType>
<Function> Authorization  </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request> </Request>

<Response>
<ActionCode ID="" Desc=""/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>

<ApprovedAmount> {Amount Aggregate} </ApprovedAmount>

<BalanceAmount> {AccountBalanceAmount Aggregate} </BalanceAmount>
[Optional, Multiple]

<PinPadInfo></PinPadInfo> [Optional]
<LanguageCode><LanguageCode> [Optional]
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]
</Response>

</FinancialMessage >
```

Purchase With CashBack request message

The following is a EBT purchase with cashback request message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> Authorization  </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request >
<TransactionNumber> </TransactionNumber>
```

```
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>

<CardDetail> </CardDetail>
<RequestedAmount>  {Amount Aggregate} </RequestedAmount>
<CashBackAmount> {Amount Aggregate} </CashBackAmount>

<PinData> </PinData>
<SecurityControlInfo></SecurityControlInfo>
<MAC> </MAC> [Optional]
<PinPadInfo></PinPadInfo> [Optional]

<LanguageCode><LanguageCode> [Optional]
</Request>
</FinancialMessage >
```

Purchase With CashBack response message

The following is a EBT purchase sale with cashback response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> Authorization  </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request> </Request>

<Response>
<ActionCode ID="" Desc=""/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>

<ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
<CashBackAmount> {Amount Aggregate} </CashBackAmount>

<BalanceAmount> {AccountBalanceAmount Aggregate} </BalanceAmount>
[Optional, Multiple]

<PinPadInfo></PinPadInfo> [Optional]
<LanguageCode><LanguageCode> [Optional]
```

```
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]
</Response>

</FinancialMessage >
```

WithdrawAll request message

The following is a EBT withdrawAll request message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> CashOut </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>

</MessageType>

<Request >
<TransactionNumber> </TransactionNumber>
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>

<CardDetail> </CardDetail>

<PinData> </PinData>
<SecurityControlInfo></SecurityControlInfo>
<MAC> </MAC> [Optional]
<PinPadInfo></PinPadInfo> [Optional]

<LanguageCode><LanguageCode> [Optional]

</Request>
</FinancialMessage >
```

WithdrawAll response message

The following is a EBT withdrawAll response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
```

```
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> CashOut </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>

</MessageType>

<Request> </Request>

<Response>
<ActionCode ID="" Desc=""/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>

<ApprovedAmount> {Amount Aggregate} </ApprovedAmount>

<BalanceAmount> {AccountBalanceAmount Aggregate} </BalanceAmount>
[Optional, Multiple]

<PinPadInfo></PinPadInfo> [Optional]
<LanguageCode><LanguageCode> [Optional]
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]
</Response>

</FinancialMessage >
```

Note: Approved Amount will contain all the withdrawn amount.

Return request message

The following is a EBT return request message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> Return  </Function>
```

```
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request >
<TransactionNumber> </TransactionNumber>
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>

<CardDetail> </CardDetail>
<RequestedAmount>  {Amount Aggregate} </RequestedAmount>

<PinData> </PinData>
<SecurityControlInfo></SecurityControlInfo>
<MAC> </MAC> [Optional]
<PinPadInfo></PinPadInfo> [Optional]

<LanguageCode><LanguageCode> [Optional]
</Request>
</FinancialMessage >
```

Return response message

The following is a EBT return response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> Return  </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request> </Request>

<Response>
<ActionCode ID="" Desc=""/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>

<ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
```

```
<BalanceAmount> {AccountBalanceAmount Aggregate} </BalanceAmount>
[Optional, Multiple]

<PinPadInfo></PinPadInfo> [Optional]
<LanguageCode><LanguageCode> [Optional]
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]
</Response>

</FinancialMessage >
```

Balance Inquiry request message

The following is a EBT balance inquiry request message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> BalanceInquiry </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request>
<TransactionNumber> </TransactionNumber>
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>

<CardDetail> </CardDetail>

<PinData> </PinData>
<SecurityControlInfo></SecurityControlInfo>
<MAC> </MAC> [Optional]
<PinPadInfo></PinPadInfo> [Optional]

<LanguageCode><LanguageCode> [Optional]

</Request>

</FinancialMessage >
```

Balance Inquiry response message

The following is a EBT balance inquiry response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> BalanceInquiry </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request> </Request>

<Response>
<ActionCode ID="" Desc=""/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>

<BalanceAmount> {AccountBalanceAmount Aggregate} </BalanceAmount>
[Optional, Multiple]

<PinPadInfo></PinPadInfo> [Optional]
<LanguageCode><LanguageCode> [Optional]
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]
</Response>

</FinancialMessage >
```

Void request message

Void requests can be sent only for the following transaction types:

- Sale
- Return

The following is a EBT void request message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
```

```
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> Authorization / Return / PostAuthorizaion </Function>
<Modifier> Void </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request >
<TransactionNumber> </TransactionNumber>
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>

<PinData> </PinData>
<SecurityControlInfo></SecurityControlInfo>
<MAC> </MAC> [Optional]
<PinPadInfo></PinPadInfo> [Optional]


<OriginalMessage>
       <Request>
  <TransactionNumber> </TransactionNumber>
  <SequenceNumber>  </SequenceNumber>
  <TransactionDateTime> </TransactionDateTime>
  <CardDetail>  </CardDetail>
  <CashBackAmount> {Amount Aggregate} </CashBackAmount>   [Optional]
</Request>
       <Response>
  <ExternalReferenceNumber> </ExternalReferenceNumber>
  <ApprovalCode> </ approvalCode >
  <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
</Response>
</OriginalMessage>

<LanguageCode><LanguageCode> [Optional]
</Request>
</FinancialMessage >
```

Note: All data from the original approved transaction request should be in the OriginalMessage element. ExternalReferenceNumber is from the original approved transaction response.

Void response message

The following is a EBT void response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
```

```
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> Authorization / Return / PostAuthorizaion </Function>
<Modifier> Void </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

</Request>

<Response>
<ActionCode ID="" Desc=""/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>

<ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
<CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]

<BalanceAmount> {AccountBalanceAmount Aggregate} </BalanceAmount>
[Optional, Multiple]

<PinPadInfo></PinPadInfo> [Optional]
<LanguageCode><LanguageCode> [Optional]
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]
</Response>

</FinancialMessage >
```

Reversal request message

Only the following types of financial transactions can be reversed:

- Sale
- Return
- StoreAndForward
- Void

The following is a EBT reversal request message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
```

```
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> [see above] </Function>
<Modifier> Reversal </Modifier>
<StoreAndForward> False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request>
<TransactionNumber> </TransactionNumber>
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>

<OriginalMessage>
       <Request>
  <TransactionNumber> </TransactionNumber>
  <SequenceNumber>  </SequenceNumber>
  <TransactionDateTime> </TransactionDateTime>

  <CardDetail>  </CardDetail>
  <RequestedAmount>  {Amount Aggregate} </RequestedAmount>
  <CashBackAmount> {Amount Aggregate} </CashBackAmount>  [Optional]


  <PinData> </PinData>
  <SecurityControlInfo></SecurityControlInfo>
  <MAC> </MAC> [Optional]
  <PinPadInfo></PinPadInfo> [Optional]

  <LanguageCode><LanguageCode> [Optional]

  <ExternalReferenceNumber> </ExternalReferenceNumber>
</Request>
</OriginalMessage>
</Request>

</FinancialMessage >
```

Note: All data should be that of the original approved transaction request.

Reversal response message

The following is a EBT reversal response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
```

```
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> [See above] </Function>
<Modifier> Reversal </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request>
</Request>

<Response>
<ActionCode ID="" Desc=""/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>

<ApprovedAmount> {Amount Aggregate}   </ApprovedAmount>
<CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]

<BalanceAmount> {AccountBalanceAmount Aggregate} </BalanceAmount>
[Optional, Multiple]

<PinPadInfo></PinPadInfo> [Optional]
<LanguageCode><LanguageCode> [Optional]
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]
</Response>
</FinancialMessage >
```

Store And Forward request message

Store and forward messages are used for locally (force) approved and voice approved transactions. The following is a EBT store and forward request message containing all the elements:

```
<FinancialMessage>
   <Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> Authorization </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  True </StoreAndForward>
```

```
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request >
<TransactionNumber> </TransactionNumber>
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>
<CardDetail>  </CardDetail>
<RequestedAmount> </RequestedAmount>

<CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]
<PinData> </PinData>
<SecurityControlInfo></SecurityControlInfo>
<MAC> </MAC> [Optional]
<PinPadInfo></PinPadInfo> [Optional]

<LanguageCode></LanguageCode> [Optional]

<DocumentReferenceNumber> </DocumentReferenceNumber> [Optional]

<OfflineResponse>
<ExternalReferenceNumber> </ExternalReferenceNumber>
<ApprovalCode> </ApprovalCode>
<ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
       </OfflineResponse>
</Request>
</FinancialMessage >
```

Note: All data from the original approved transaction request should be in the OriginalMessage element. ExternalReferenceNumber is from the original approved transaction response

Store And Forward response message

Store and forward messages are used for locally (force) approved and voice approved transactions. The following is a EBT store and forward response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> Authorization </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  true </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>
```

```
   <Request>
</Request>

<Response>
<ActionCode ID="" Desc=""/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>

<ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
<CashBackAmount> {Amount Aggregate} </CashBackAmount> [Optional]

<BalanceAmount> {AccountBalanceAmount Aggregate} </BalanceAmount>
[Optional, Multiple]

<PinPadInfo></PinPadInfo> [Optional]
<LanguageCode><LanguageCode> [Optional]
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]
</Response>
</FinancialMessage >
```

Pre Authorization request message

The following is a EBT pre authorization request message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> PreAuthorization  </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request >
<TransactionNumber> </TransactionNumber>
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>

<CardDetail> </CardDetail>
<RequestedAmount>  {Amount Aggregate} </RequestedAmount>
```

```
<PinData> </PinData>
<SecurityControlInfo></SecurityControlInfo>
<MAC> </MAC> [Optional]
<PinPadInfo></PinPadInfo> [Optional]

<LanguageCode><LanguageCode> [Optional]
</Request>
</FinancialMessage >
```

Pre Authorization response message

The following is a EBT pre authorization response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> PreAuthorization  </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request>
</Request>

<Response>
<ActionCode ID="" Desc=""/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>

<ApprovedAmount> {Amount Aggregate} </ApprovedAmount>

<BalanceAmount> {AccountBalanceAmount Aggregate} </BalanceAmount>
[Optional, Multiple]

<PinPadInfo></PinPadInfo> [Optional]
<LanguageCode><LanguageCode> [Optional]
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]

</Response>
</FinancialMessage >
```

Post Authorization request message

Also called as Pre Authorization Completion request message. The following is a EBT post authorization request message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> PostAuthorization </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request>
<TransactionNumber> </TransactionNumber>
<SequenceNumber>  </SequenceNumber>
<TransactionDateTime> </TransactionDateTime>

<RequestedAmount> </RequestedAmount>

<OriginalMessage>
          <Request>
  <TransactionNumber> </TransactionNumber>
  <SequenceNumber>  </SequenceNumber>
  <TransactionDateTime> </TransactionDateTime>
  <CardDetail>  </CardDetail>
          </Request>
          <Response>
  <ExternalReferenceNumber> </ExternalReferenceNumber>
  <ApprovalCode> </ApprovalCode>
  <ApprovedAmount> {Amount Aggregate} </ApprovedAmount>
</Response>
</OriginalMessage>

<PinData> </PinData>
<SecurityControlInfo></SecurityControlInfo>
<MAC> </MAC> [Optional]
<PinPadInfo></PinPadInfo> [Optional]

<LanguageCode><LanguageCode> [Optional]
</Request>
</FinancialMessage >
```

Note: All data from the original approved transaction request should be in the OriginalMessage element. ExternalReferenceNumber is from the original approved transaction response

Post Authorization response message

Also called as Pre Authorization Completion response message. The following is a EBT post authorization response message containing all the elements:

```
<FinancialMessage>
<Originator>
<Chain>  </Chain>
<Store>  </Store>
<Register>  </Register>
</Originator>

<MessageType>
<AccountType> EBT </AccountType>
<Function> PostAuthorization </Function>
<Modifier> Original </Modifier>
<StoreAndForward>  False </StoreAndForward>
<AdditionalModifier>Original</AdditionalModifier>
</MessageType>

<Request>
</Request>

<Response>
<ActionCode ID="" Desc=""/>
<ApprovalCode> </ApprovalCode>

<ExternalReferenceNumber> </ExternalReferenceNumber>
<ApprovedAmount> {Amount Aggregate} </ApprovedAmount>

<BalanceAmount> {AccountBalanceAmount Aggregate} </BalanceAmount>
[Optional, Multiple]

<PinPadInfo></PinPadInfo> [Optional]
<LanguageCode><LanguageCode> [Optional]
<ExternalSequenceNumber></ExternalSequenceNumber> [Optional]
<DisplayInfo></DisplayInfo> [Optional]
<AcknowledgementRequired></AcknowledgementRequirement> [Optional]
</Response>
</FinancialMessage >
```

Sample FinancialMessage XMLs

Credit card authorization (manual)

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>CreditCard</AccountType>
      <Function>Authorization</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>183</TransactionNumber>
      <SequenceNumber>20</SequenceNumber>
      <TransactionDateTime>20041102161623.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Keyed</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber>5454545454545454</AccountNumber>
         <ExpiryDate>200505</ExpiryDate>
      </CardDetail>
      <RequestedAmount Value="11.00" CurrencyCode="USD"/>
      <PinPadInfo/>
   </Request>
</FinancialMessage>
```

Credit authorization reversal (manual)

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>CreditCard</AccountType>
      <Function>Authorization</Function>
      <Modifier>Reversal</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>181</TransactionNumber>
      <SequenceNumber>254</SequenceNumber>
```

```
            <TransactionDateTime>20041102160716.000</TransactionDateTime>
            <OriginalMessage>
               <Request>
                  <TransactionNumber>181</TransactionNumber>
                  <SequenceNumber>9</SequenceNumber>
                  <TransactionDateTime>20041102160716.000</TransactionDateTime>
                  <CardDetail>
                     <InputMode>Keyed</InputMode>
                     <CardPresent>True</CardPresent>
                     <AccountNumber>4055011111111111</AccountNumber>
                     <ExpiryDate>200505</ExpiryDate>
                  </CardDetail>
                  <RequestedAmount Value="47.70" CurrencyCode="USD"/>
                  <PinPadInfo/>
               </Request>
            </OriginalMessage>
            <PinPadInfo/>
         </Request>
      </FinancialMessage>
```

Credit authorization (swiped)

```
      <FinancialMessage>
         <Originator>
            <Chain>1</Chain>
            <Store>1</Store>
            <Register>1</Register>
         </Originator>
         <MessageType>
            <AccountType>CreditCard</AccountType>
            <Function>Authorization</Function>
            <Modifier>Original</Modifier>
            <StoreAndForward>False</StoreAndForward>
            <AdditionalModifier>Original</AdditionalModifier>
         </MessageType>
         <Request>
            <TransactionNumber>211</TransactionNumber>
            <SequenceNumber>35</SequenceNumber>
            <TransactionDateTime>20041103122302.000</TransactionDateTime>
            <CardDetail>
               <InputMode>Swiped</InputMode>
               <CardPresent>True</CardPresent>
               <AccountNumber>3566002020140006</AccountNumber>
               <ExpiryDate>200612</ExpiryDate>
               <Track2>3566002020140006=06121015432112345678</Track2>
            </CardDetail>
            <RequestedAmount Value="1.26" CurrencyCode="USD"/>
            <TotalTaxAmount Value="0.07" CurrencyCode="USD"/>
            <PinPadInfo/>
         </Request>
```

```
</FinancialMessage>
```

Credit void authorization

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>CreditCard</AccountType>
      <Function>Authorization</Function>
      <Modifier>Void</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>344</TransactionNumber>
      <SequenceNumber>7</SequenceNumber>
      <TransactionDateTime>20041104163525.000</TransactionDateTime>
      <OriginalMessage>
         <Request>
            <TransactionNumber>341</TransactionNumber>
            <SequenceNumber>1</SequenceNumber>
            <TransactionDateTime>20041104163302.000</TransactionDateTime>
            <CardDetail>
               <InputMode>Keyed</InputMode>
               <CardPresent>True</CardPresent>
               <AccountNumber>4788250000028291</AccountNumber>
               <ExpiryDate>200512</ExpiryDate>
            </CardDetail>
            <RequestedAmount Value="10.00" CurrencyCode="USD"/>
            <PinPadInfo/>
         </Request>
         <Response>
            <ActionCode Id="000" Desc=""/>
            <ApprovalCode>092649</ApprovalCode>
            <ApprovedAmount Value="10.00" CurrencyCode="USD"/>
         </Response>
      </OriginalMessage>
      <PinPadInfo/>
   </Request>
</FinancialMessage>
```

Credit return

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
```

```
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>CreditCard</AccountType>
      <Function>Return</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>223</TransactionNumber>
      <SequenceNumber>77</SequenceNumber>
      <TransactionDateTime>20041103123713.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Swiped</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber>4788250000028291</AccountNumber>
         <ExpiryDate>200512</ExpiryDate>
         <Track2>4788250000028291=05121015432112345678</Track2>
      </CardDetail>
      <RequestedAmount Value="32.00" CurrencyCode="USD"/>
      <PinPadInfo/>
   </Request>
</FinancialMessage>
```

Debit authorization

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>DebitCard</AccountType>
      <Function>Authorization</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>349</TransactionNumber>
      <SequenceNumber>30</SequenceNumber>
      <TransactionDateTime>20041104164456.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Swiped</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber>9999999800002773</AccountNumber>
         <ExpiryDate>200612</ExpiryDate>
         <Track2>9999999800002773=06121015432112345678</Track2>
```

```
            <AffectedAccountName>Checking</AffectedAccountName>
         </CardDetail>
         <RequestedAmount Value="20.00" CurrencyCode="USD"/>
         <CashBackAmount Value="5.00" CurrencyCode="USD"/>
         <PinPadInfo>
            <PinData PinLength="16">DF87F1E2E5CD5CB1</PinData>
            <SecurityControlInfo>F0001081020108000000A</SecurityControlInfo>
         </PinPadInfo>
      </Request>
   </FinancialMessage>
```

Debit return

```
   <FinancialMessage>
      <Originator>
         <Chain>1</Chain>
         <Store>1</Store>
         <Register>1</Register>
      </Originator>
      <MessageType>
         <AccountType>DebitCard</AccountType>
         <Function>Return</Function>
         <Modifier>Original</Modifier>
         <StoreAndForward>False</StoreAndForward>
         <AdditionalModifier>Original</AdditionalModifier>
      </MessageType>
      <Request>
         <TransactionNumber>777</TransactionNumber>
         <SequenceNumber>128</SequenceNumber>
         <TransactionDateTime>20041115154243.000</TransactionDateTime>
         <CardDetail>
            <InputMode>Swiped</InputMode>
            <CardPresent>True</CardPresent>
            <AccountNumber>9999999800002773</AccountNumber>
            <ExpiryDate>200612</ExpiryDate>
            <Track2>9999999800002773=06121015432112345678</Track2>
         </CardDetail>
         <RequestedAmount Value="20.00" CurrencyCode="USD"/>
         <PinPadInfo>
            <PinData PinLength="16">CD495A9D231C0E4D</PinData>
            <SecurityControlInfo>FFFF000000005B000012</SecurityControlInfo>
         </PinPadInfo>
      </Request>
   </FinancialMessage>
```

Check authorization (manually entered)

```
   <FinancialMessage>
      <Originator>
         <Chain>1</Chain>
         <Store>1</Store>
```

```
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>Cheque</AccountType>
      <Function>Authorization</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>188</TransactionNumber>
      <SequenceNumber>29</SequenceNumber>
      <TransactionDateTime>20041102162030.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Keyed</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber>12345678</AccountNumber>
         <ExpiryDate>200411</ExpiryDate>
         <ChequeNumber>1234</ChequeNumber>
         <BankNumber>1234</BankNumber>
         <MICRNumber>1234 12345678 1234</MICRNumber>
         <VerificationMethod Type="MICR">
            <Number>1234</Number>
         </VerificationMethod>
      </CardDetail>
      <RequestedAmount Value="15.00" CurrencyCode="USD"/>
      <PinPadInfo/>
   </Request>
</FinancialMessage>
```

Check authorization (scanned)

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>Cheque</AccountType>
      <Function>Authorization</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>783</TransactionNumber>
      <SequenceNumber>1</SequenceNumber>
      <TransactionDateTime>20041115154752.000</TransactionDateTime>
      <CardDetail>
```

```
            <InputMode>MICR</InputMode>
            <CardPresent>True</CardPresent>
            <AccountNumber>1712026</AccountNumber>
            <ExpiryDate>200411</ExpiryDate>
            <ChequeNumber>412</ChequeNumber>
            <BankNumber>041000124</BankNumber>
            <MICRNumber>041000124 1712026 0412</MICRNumber>
            <VerificationMethod Type="MICR">
               <Number>041000124</Number>
            </VerificationMethod>
         </CardDetail>
         <RequestedAmount Value="16.00" CurrencyCode="USD"/>
         <PinPadInfo/>
      </Request>
   </FinancialMessage>
```

EBT (Cash) authorization

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>EBT</AccountType>
      <Function>Authorization</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>349</TransactionNumber>
      <SequenceNumber>27</SequenceNumber>
      <TransactionDateTime>20041104164406.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Swiped</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber>9999999800002773</AccountNumber>
         <ExpiryDate>200612</ExpiryDate>
         <Track2>9999999800002773=06121015432112345678</Track2>
         <AffectedAccountName>Cash</AffectedAccountName>
      </CardDetail>
      <RequestedAmount Value="15.00" CurrencyCode="USD"/>
      <PinPadInfo>
         <PinData PinLength="16">B7BD1DF8FC8AC0B6</PinData>
         <SecurityControlInfo>F0001081020108000009</SecurityControlInfo>
      </PinPadInfo>
   </Request>
</FinancialMessage>
```

EBT (Foodstamp) authorization

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>EBT</AccountType>
      <Function>Authorization</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>349</TransactionNumber>
      <SequenceNumber>33</SequenceNumber>
      <TransactionDateTime>20041104164540.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Swiped</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber>9999999800002773</AccountNumber>
         <ExpiryDate>200612</ExpiryDate>
         <Track2>9999999800002773=06121015432112345678</Track2>
         <AffectedAccountName>FoodStamp</AffectedAccountName>
      </CardDetail>
      <RequestedAmount Value="15.00" CurrencyCode="USD"/>
      <PinPadInfo>
         <PinData PinLength="16">61710F18B0C26829</PinData>
         <SecurityControlInfo>F0001081020108000000B</SecurityControlInfo>
      </PinPadInfo>
   </Request>
</FinancialMessage>
```

EBT (Foodstamp) Balance inquiry

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>EBT</AccountType>
      <Function>BalanceInquiry</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
```

```
            <TransactionNumber>761</TransactionNumber>
            <SequenceNumber>45</SequenceNumber>
            <TransactionDateTime>20041115150030.000</TransactionDateTime>
            <CardDetail>
               <InputMode>Swiped</InputMode>
               <CardPresent>True</CardPresent>
               <AccountNumber>9999999800002773</AccountNumber>
               <ExpiryDate>200612</ExpiryDate>
               <Track2>9999999800002773=06121015432112345678</Track2>
               <AffectedAccountName>FoodStamp</AffectedAccountName>
            </CardDetail>
            <PinPadInfo>
               <PinData PinLength="16">678F1687748D4850</PinData>
               <SecurityControlInfo>FFFF000000005B00000E</SecurityControlInfo>
            </PinPadInfo>
         </Request>
      </FinancialMessage>
```

EBT (Foodstamp) Return

```
      <FinancialMessage>
         <Originator>
            <Chain>1</Chain>
            <Store>1</Store>
            <Register>1</Register>
         </Originator>
         <MessageType>
            <AccountType>EBT</AccountType>
            <Function>Return</Function>
            <Modifier>Original</Modifier>
            <StoreAndForward>False</StoreAndForward>
            <AdditionalModifier>Original</AdditionalModifier>
         </MessageType>
         <Request>
            <TransactionNumber>772</TransactionNumber>
            <SequenceNumber>64</SequenceNumber>
            <TransactionDateTime>20041115152413.000</TransactionDateTime>
            <CardDetail>
               <InputMode>Swiped</InputMode>
               <CardPresent>True</CardPresent>
               <AccountNumber>9999999800002773</AccountNumber>
               <ExpiryDate>200612</ExpiryDate>
               <Track2>9999999800002773=06121015432112345678</Track2>
               <AffectedAccountName>FoodStamp</AffectedAccountName>
            </CardDetail>
            <RequestedAmount Value="25.00" CurrencyCode="USD"/>
            <PinPadInfo>
               <PinData PinLength="16">592AC8543474F68B</PinData>
               <SecurityControlInfo>FFFF000000005B000011</SecurityControlInfo>
            </PinPadInfo>
```

```
   </Request>
</FinancialMessage>
```

Stored Value Card (Gift Card) Issue

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>StoredValueCard</AccountType>
      <Function>Activate</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>778</TransactionNumber>
      <SequenceNumber>131</SequenceNumber>
      <TransactionDateTime>20041115154331.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Swiped</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber>6035718888880552519</AccountNumber>
         <ExpiryDate>204912</ExpiryDate>
         <Track2>6035718888880552519=4912</Track2>
      </CardDetail>
      <RequestedAmount Value="100.00" CurrencyCode="USD"/>
      <PinPadInfo/>
   </Request>
</FinancialMessage>
```

Stored Value Card (Gift Card) payment (reload)

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>StoredValueCard</AccountType>
      <Function>Payment</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>346</TransactionNumber>
```

```
      <SequenceNumber>19</SequenceNumber>
      <TransactionDateTime>20041104163913.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Swiped</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber/>
         <ExpiryDate>204912</ExpiryDate>
         <Track2>6035718888880552527=4912</Track2>
      </CardDetail>
      <RequestedAmount Value="25.00" CurrencyCode="USD"/>
      <PinPadInfo/>
   </Request>
</FinancialMessage>
```

Stored Value Card (Gift Card) Authorization (redemption)

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>StoredValueCard</AccountType>
      <Function>Authorization</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>348</TransactionNumber>
      <SequenceNumber>22</SequenceNumber>
      <TransactionDateTime>20041104164206.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Keyed</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber>6035718888880552527</AccountNumber>
         <ExpiryDate>200511</ExpiryDate>
      </CardDetail>
      <RequestedAmount Value="12.00" CurrencyCode="USD"/>
      <PinPadInfo/>
   </Request>
</FinancialMessage>
```

Stored Value Card (Gift Card) post-authorization

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
```

```
   </Originator>
   <MessageType>
      <AccountType>StoredValueCard</AccountType>
      <Function>Authorization</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>True</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>521</TransactionNumber>
      <SequenceNumber>54088</SequenceNumber>
      <TransactionDateTime>20041105191653.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Swiped</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber>6035718888880552535</AccountNumber>
         <ExpiryDate>204912</ExpiryDate>
         <Track2>6035718888880552535=4912</Track2>
      </CardDetail>
      <RequestedAmount Value="21.19" CurrencyCode="USD"/>
      <TotalTaxAmount Value="1.20" CurrencyCode="USD"/>
      <PinPadInfo/>
      <OfflineResponse>
         <ActionCode Id="000" Desc=""/>
         <ApprovalCode>333333</ApprovalCode>
         <ApprovedAmount Value="21.19" CurrencyCode="USD"/>
         <Reason ID="1005" Desc="AboveFloorLimit"/>
      </OfflineResponse>
   </Request>
</FinancialMessage>
```

Stored Value Card (Gift Card) Balance Inquiry

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>StoredValueCard</AccountType>
      <Function>BalanceInquiry</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>208</TransactionNumber>
      <SequenceNumber>1</SequenceNumber>
      <TransactionDateTime>20041103121925.000</TransactionDateTime>
```

```
      <CardDetail>
         <InputMode>Swiped</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber/>
         <ExpiryDate>204912</ExpiryDate>
         <Track2>6035718888880552527=4912</Track2>
      </CardDetail>
      <PinPadInfo/>
   </Request>
</FinancialMessage>
```

Stored Value Card (Gift Card) Cash Out

```
<FinancialMessage>
   <Originator>
      <Chain>1</Chain>
      <Store>1</Store>
      <Register>1</Register>
   </Originator>
   <MessageType>
      <AccountType>StoredValueCard</AccountType>
      <Function>CashOut</Function>
      <Modifier>Original</Modifier>
      <StoreAndForward>False</StoreAndForward>
      <AdditionalModifier>Original</AdditionalModifier>
   </MessageType>
   <Request>
      <TransactionNumber>194</TransactionNumber>
      <SequenceNumber>73</SequenceNumber>
      <TransactionDateTime>20041102164836.000</TransactionDateTime>
      <CardDetail>
         <InputMode>Keyed</InputMode>
         <CardPresent>True</CardPresent>
         <AccountNumber>6035718888880552527</AccountNumber>
         <ExpiryDate>200510</ExpiryDate>
      </CardDetail>
      <PinPadInfo/>
   </Request>
</FinancialMessage>
```

About the APM Adapter API

5

This section provides information about creating a custom Application Processing Module, or APM, for use within Centralized EFT or File Transfer. It covers the following topics:

- Writing a custom APM adapter (see page 159)
- Configuring a custom APM adapter (see page 168)

Writing a custom APM adapter

Centralized EFT allows any number of applications to attach to a simple programming interface, and send messages to a transaction server. The main purpose of the transaction server is to receive a transaction from an application on one machine, deliver it to an Application Processing Module (APM), wait for the APM to respond, and then return the response to the application. The APM might, for example, forward the transaction to an outside party for an authorization, or simply perform a central database query. The Centralized EFT server may be connected to any number of Application Processing Modules. These modules can include a range of functionality from credit card authorization to shared database lookup.

Each APM contains a unique adapter set within an adapter context which communicates with the rest of the APM and the application via an integrated adapter interface. To produce a custom APM, you need to specify a name for your APM and adapter context, and define a unique adapter based on the type of messages you want to process. An Adapter API (Application Programming Interface) is provided for this purpose. This API consists of five methods, which you must implement so that the message type you wish to connect to can be understood by the adapter interface layer. See “Adapter API” on page 160 for information on the API methods. For more information about the adapter interface layer, see “About the Adapter Context” on page 161.

Note: While you must implement all of the listed methods, your APM does not have to process all of the actions indicated by these methods. Simply set your APM to not process certain message types.

Adapter API

Centralized EFT/File Transfer use this interface to process requests within an APM. These requests originate at the client and the serve delivers the messages to the APM.

Note: This section is intended for technical personnel with a comprehensive knowledge of Java programming. All code must be written using Java 1.3 or greater.

Methods

doText

```
public java.lang.String doText(java.lang.String text)
```

This method processes text messages and returns a text message response.

Parameters	Returns
`text` - A text message to be processed	A text response

doBytes

```
public byte[] doBytes(byte[] bytes)
```

This method processes a byte array message and returns a byte array response.

Parameters	Returns
`bytes` - A byte message to be processed	A byte array response

doDocument

```
public org.jdom.Document doDocument(org.jdom.Document doc)
```

This method processes a given XML document and returns another document in response.

Parameters	Returns
`doc` - The document as the message to be processed	A document response

setContext

```
public void setContext(AdapterContext context)
```

This method specifies the context in which this adapter will work (that is, it tells the adapter where it is located). Note that you can have multiple adapters, but you must specify a single context for each adapter (you can have multiple instances of the same class definition).

Parameters	Returns
`context` - The context for this adapter	The adapter context

getName

```
public java.lang.String getName()
```

This method gets the name of the adapter (that is, whatever name you give it).

Parameters	Returns
	The name of the adapter implemented

About the Adapter Context

The adapter context acts as an interface layer between an APM and the core components of the Centralized EFT. It defines an interface to be implemented by a class containing an adapter (for example, a custom APM). It reads the requests processed through the APM (for example, a credit authorization request), and specifies the actions to be taken. A description of the methods employed follows.

Note: This code is set and is not provided as part of the API. The following information is provided for your information only.

Methods

onSend

Arguments	Parameters
`public void onSend(Adapter adapter)` This method notifies the container that a message has been sent. It also causes the context to update the meter accordingly.	`adapter` - The reporting adapter object
`public void onSend(Adapter adapter, java.lang.String message)` This method notifies the container that a message has been sent. It also causes the context to update the meter accordingly and logs the message.	`adapter` - The reporting adapter object `message` - The message that was sent

onReceive

Arguments	Parameters
`public void onReceive(Adapter adapter)` This method notifies the container that a message has been received. It also causes the context to update the meter accordingly.	`adapter` - The reporting adapter object
`public void onReceive(Adapter adapter, java.lang.String message)` This method notifies the container that a message has been received. It also causes the context to update the meter accordingly and logs the message.	`adapter` - The reporting adapter object `message` - The message that was received

log

Arguments	Parameters
public void log(Adapter adapter), java.lang.String message This method provides logging for the context.	`adapter` - The logging source `message` - The message to be logged
`public void log(Adapter adapter, byte[] message)` This method provides logging for the context.	`adapter` - The logging source `message` - The message to be logged Hex values will be converted to ASCII digits before logging

getConfig

```
public java.util.Properties getConfig(Adapter adapter)
```

This method retrieves the configuration information specific to the adapter from the adapter properties files. The adapter specified is read from the **getName** adapter argument (see page 161).

Parameters	Returns
`adapter` - Identifies the source of the configuration request	The adapter configurations as a properties object

ExpressReturns adapter

ExpressReturns is a SAP product designed to manage the processing of merchandise returns across a business entity. A facility exists within ExpressReturns which allows a register (for example, Transactionware GM) to send formatted messages and receive replies. This Adapter allows these messages to be routed through Centralized EFT/File Transfer to the ExpressReturns server.

The Adapter does not modify the messages except to extract the actual ExpressReturns message from the XML message. It also does not interpret the results. It exists only as a message switch.

How the adapter works

Transactionware GM sends ExpressReturns messages formatted in a methodCall XML.

The ExpressReturns adapter extracts the ExpressReturns message from the methodCall XML document and sends it to the ExpressReturns server. The adapter opens a connection to the server, sends the message, waits for a response and reformats the response into a methodResponse XML document.

Connection to the ExpressReturns server is through a TCP/IP socket. The connection is opened when the message is extracted successfully from the methodCall XML document, and is closed after the response is received. Note that before the adapter closes the socket it sends the message "Done:" to the ExpressReturns server to indicate the socket is to be closed.

The methodResponse XML document includes a Result element, which contains the response either from ExpressReturns or from Centralized EFT/File Transfer in case of an error. The result element may contain:

- A response from ExpressReturns
- Invalid message format— This text is sent back when the Adapter cannot find the params/param/value part of the methodCall XML document.
- Communications error— This text is sent back when an error occurs in the communications, usually that the connection could not be established to the ExpressReturns server.

Sample message formats

The following is a sample of a message sent from Transactionware GM:

```
<?xml version="1.0"?>
<methodCall>
<methodName>returnRequest</methodName>
<params>
```

```
<param>
<value>TT=00;TS=0001;TR=02;TN=1234;RD=123004;RH=100402;CN=0001;EF=FILLER;
MN=12345678901;MS=S;MX=0000;OD=1224;OS=0001;OR=02;ON=1235;OZ=S;FI=;ST=1;S
S=PA;SI=19545126;IT=00;IS=1;IQ=1;IP=398:</value>
</param>
</params>
</methodCall>
```

The following is an example of a response message for an error:

```
<MethodResponse>
<Result>Communications error</Result>
</MethodResponse>
```

The following is an example of a response message for a successful response:

```
<MethodResponse>
<Result>RT=00;TT=00;TS=0001;TR=02;TN=1234;EF=FILLER;AC=00;AN=0000;AD=DISP
LAY1   DISPLAY2            ;AJ=THIS MESSAGE TO THE JOURNAL
;ST=0;SS=00;TY=01;MO=0;IS=1;IQ=1;IP=398:></Result>
</MethodResponse>
```

High availability feature

Originally the ExpressReturns Adapter for routing messages to and from the ExpressReturns server was designed without provision for failover if the main connection to the ExpressReturns Server failed. The ExpressReturns adapter can be configured for multiple connections such that the if adapter fails to make a connnection at first it automatically goes to the next connection. The following code sample illustrates the changes you need to make to the transnet.xml file to enable this feature.

Sample for a single connection in the Transnet.xml file

```
    <Connection
          name="ExpressReturns"

class="com.triversity.transnet.core.tms.connection.TNMessageConsumerConne
ction" >
          <adapter class =
"com.triversity.transnet.xtension.xreturn.ExpressReturnsAdapter"
                encoding = "UTF-16"

connectionClassName="com.triversity.transnet.xtension.xreturn.ExpressRetu
rnsAPMConnection"
                serverURL="69.4.2.156:5005"
                connectionTimeoutMillis="30000"/>
     </Connection>
```

Sample for multiple connections in the Transnet.xml file

```
    <Connection
          name="ExpressReturns"

class="com.triversity.transnet.core.tms.connection.TNMessageConsumerConne
ction" >
          <adapter class =
"com.triversity.transnet.xtension.xreturn.ExpressReturnsAdapter"
                encoding = "UTF-16"
```

```
connectionClassName="com.triversity.transnet.xtension.xreturn.ExpressRetu
rnsAPMConnection">
           <connections>
       <connection id="1"   serverURL="69.4.2.156:5005"
connectionTimeoutMillis="30000"/>
       <connection id="2"   serverURL="44.25.202.44:5005"
connectionTimeoutMillis="30000"/>
           </connections>
</adapter>


     </Connection>
```

Trickle load feature

The ExpressReturns adaptor allows File Transfer to load “trickled transactions” to SAP’s ExpressReturns on a frequent, near real time basis. The trickle load function means that ExpressReturns can validate returned items the same day as purchase. The trickle load adapter routes the TLog message to the ExpressReturn server, one transaction per message.

A new option "routeToAll" allows user to configure the behavior of the File Transfer Routing Rule. By default, it is false, i.e., the original behavior is preserved. If "routeToAll" is set to true, the sending process sends message to all routes listed. If one of the routes failed, the process is failed. The following code sample illustrates the changes you need to make to the transnet.xml file to enable this feature.

Note: ExpressReturns Adaptor runs on IBM WebSphere Application Server version 6. IBM's Client for JMS on J2SE has to be installed and include the jar files: sibc.jms.jar, sibc.jndi.jar, sibc.orb.jar to use the ExpressReturns Trickle Load Adapter

---

Configuration sample for the Transnet.xml file

```
<Connection name="ExpressReturnTLogUploader"
class="com.triversity.transnet.core.tms.connection.TNMessageConsumerConne
ction">
      <adapter class =
"com.triversity.transnet.xtension.xreturn.ExpressReturnsTLogUploader"
      contextBroker="com.ibm.websphere.naming.WsnInitialContextFactory"
      providerURL="iiop://allegiance3:2809/"
providerEndpoints="allegiance3:7276"
      urlPackage=""
queueBroker="jms/ERConnect"
queue="jms/ERQueue">
      </adapter>
    </Connection>
```

The followings is an example of the routing rule has "rotueToAll" option: In this case, the message TLog is sent to both XXTlogRoute and ERTlogRoute.

```
      <RoutingRule message="TLog" routeToAll="true">
         <Route name="XXTlogRoute" timeoutSeconds="15"/>
         <Route name="ERTlogRoute" timeoutSeconds="15"/>
      </RoutingRule>
```

Messaging compatibility feature

To make the messaging between ExpressReturn and File Transfer function correctly when different versions of the ER software are installed, the following options have been added to the ExpressReturn adaptor:

- useEOT - Sends an End of Transmission (EOT) at the end of the message response.
  The value is: true or false.
- waitForData - Indicates how many milliseconds the system should wait for the subsequence message which is blocked before it considered the respond is finished.
  The value is a positive whole number.

In general, to ensure messaging works correctly, set useEOT to “true”. If, however you are implementing Transnet 1.6 with ExpressReturn with a build number of 233 (ER Service Pack 2) or less, set the useEOT option to “false”.

If you find that long response messages are truncated when implementing Transnet 1.6 with ExpressReturn with a build number of 233 (ER Service Pack 2) or less, you can set the waitForData to prevent this. The number of millisecond for waitForData varies from system to system. Usually, a value between one-tenth of a second (100) to a second (1000) should solve the problem.

Note: When both useEOT and waitForData options are set, the system does not wait for the EOT indefinitely. It stops waiting after the number of milliseconds specified in waitForData.

Sample of the useEOT option

```
      <Connection name="ExpressReturns"
class="com.triversity.transnet.core.tms.connection.TNMessageConsumerConne
ction">
         <adapter
class="com.triversity.transnet.xtension.xreturn.ExpressReturnsAdapter"
            encoding="UTF-16"
connectionClassName="com.triversity.transnet.xtension.xreturn.ExpressRetu
rnsAPMConnection"
            serverURL="127.0.0.1:5350"
            connectionTimeoutMillis="10000"
            useEOT="true"
        />
      </Connection>
```

Sample of the waitForData option

```
      <Connection name="ExpressReturns"
class="com.triversity.transnet.core.tms.connection.TNMessageConsumerConne
ction">
         <adapter
class="com.triversity.transnet.xtension.xreturn.ExpressReturnsAdapter"
            encoding="UTF-16"
connectionClassName="com.triversity.transnet.xtension.xreturn.ExpressRetu
rnsAPMConnection"
            serverURL="127.0.0.1:5350"
```

```
      connectionTimeoutMillis="10000"
      waitForData="250"
  />
</Connection>
```

Configuring a custom APM adapter

After you have created your custom APM using the adapter API provided, you must configure it in your system. There are a number of files you will need to alter in order to do this.

To write and configure a custom APM:

1. Create and compile your java file, using the methods prescribed in the Adapter API (for more information see “Adapter API” on page 160).
2. Do one of the following:
   - Create a .jar (Java Archive Resource) file from the compiled code (named using the name of your APM adapter, for example MyAdapter.jar).

     OR
   - Copy the entire Java directory.
3. Copy the .jar file or Java directory to the location of your installation files (by default on a Windows system these will be located at C:Programs\Transnet\lib).

6 About the Credit APM Adapter API

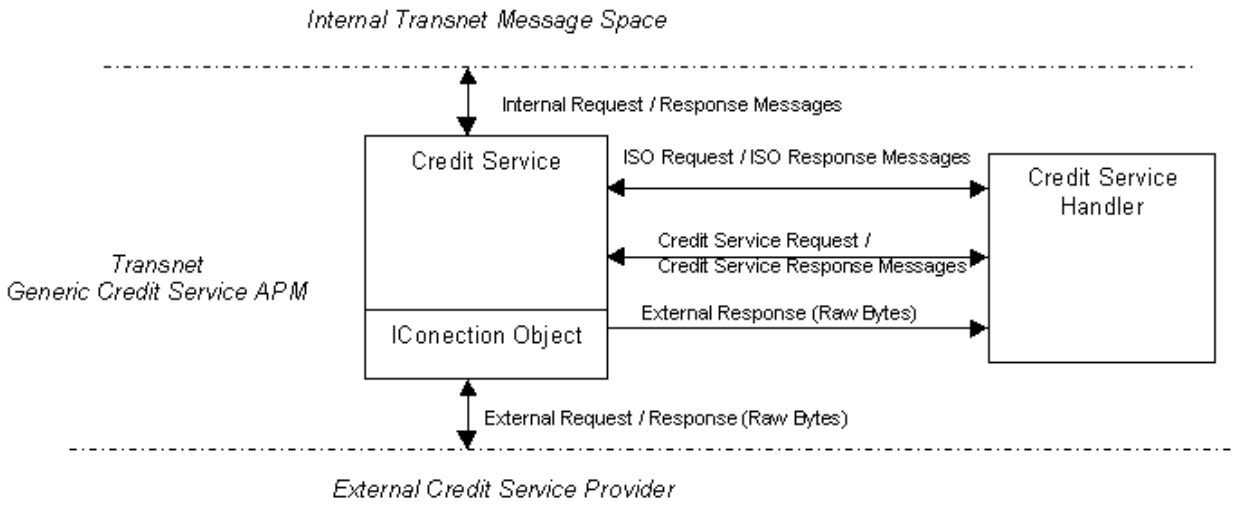
This section provides information about creating a custom Credit APM (Application Processing Module).

Like the basic application processing module (APM) API, the Credit Service APM API is a framework that you can use to implement specific credit services. Containing functionality common to most credit services, it provides a standard interface for handling requests and responses, and can manage logging, metering, and exceptions.

About the Credit Service APM

The Generic Credit Service APM framework is made up of three main components: the Credit Service, Credit Service Handler, and IConnection.

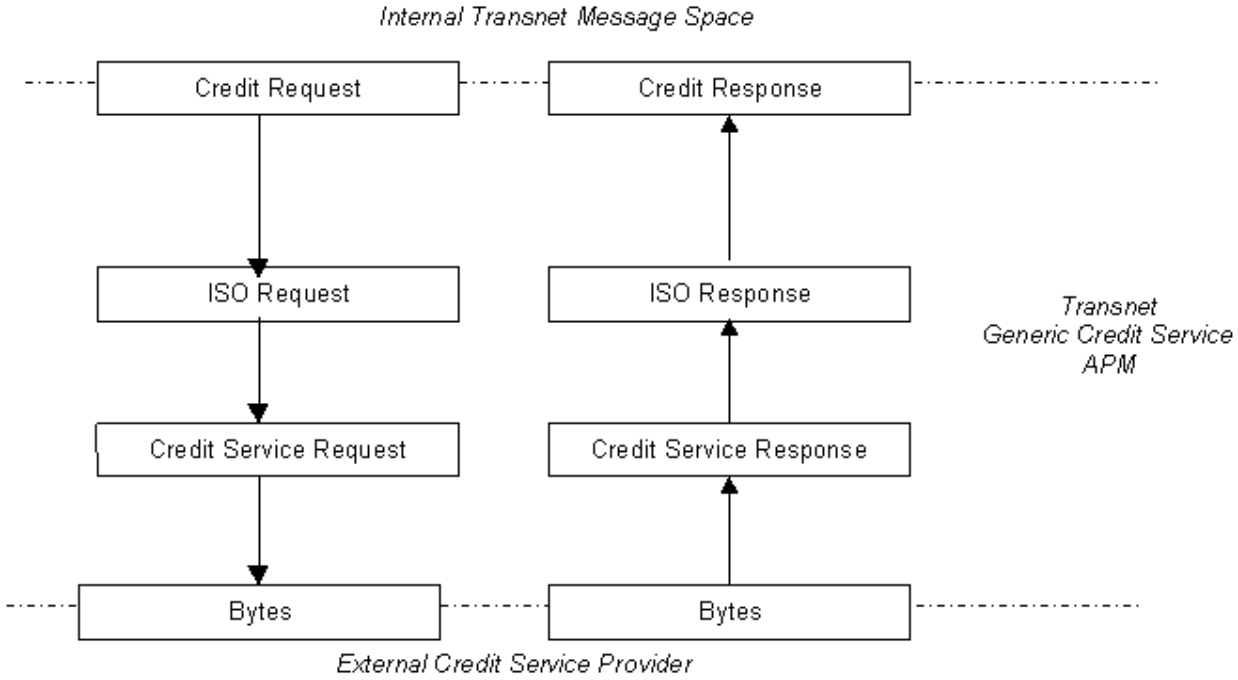
The following diagram outlines the architecture of the system:



The Credit Service object receives a credit request message from the internal message space. It extracts the ISO request message from the credit request message and based on the type of the ISO message calls the appropriate Credit Service Handler method to convert the ISO request message to a Credit Service Request message. The Credit Service object sends the Credit Service Request object to the IConnection object, which will send raw bytes external request to the External Service Provider. The Credit Service Object will wait for a specified time interval for a response. If not, then it will check for requests and then response and so on.

Once the Credit Service gets an external response from the IConnection object, it will send the external response, which is just raw bytes, to the Credit Service Handler to convert the external response to a Credit Service Response object. Then based on the Credit Service response type calls the appropriate Credit Service Handler to convert the Credit Service Response object to a ISO response object. It will then send the credit response message having the ISO response object to the internal message space.

The following diagram represents the Credit Service APM data flow:



In addition to sending requests and receiving responses, it also does the following:

- On startup, if there are any configuration errors, the Credit Service will log the error. It will continue running and if it receives any request, it will send a System Malfunction response.
- If configured, can send echo request at specified time intervals.
- Opens the connection when it receives the first request and closes the connection if it receives a IOException from the IConnection object. Before closing the connection, it will make sure that all responses have been read from the IConnection object. If it receives any request while it is closing the connection, it will send an issuer unavailable response.
- It can pre process a request before sending any request to a credit service provider. For example it pre processes network sign on request by getting your system configurations.
- It handles late responses.
- Logs data received and sent from/to the system message space and the credit service provider.
- Meters the data it receives and sends.

Credit service APM classes

The following are the main credit service APM classes:

- CreditService: The generic credit service APM
- CreditServiceHandler: The interface for converting internal messages to external and vice versa. Also, used for getting the IConection interface object and creating the echo request
- BaseCreditServiceHandler: The concrete implementation of the CreditServiceHandler interface
- CreditServiceMessage: This is the base class for all external message classes, inherited from AbstractPersistentObject. It contains functionality that is common to all external messages that go through the CreditService object. It contains methods that provide information on the type of message being sent or received

  It also has a method to get a unique identifier in a credit service and a method to validate the message
- Credit Service Request: This is the base class for all external requests, inherited from CreditServiceMessage

  This class has one extra method called respond, which returns a corresponding CreditServiceResponse object
- Credit Service Response: This is the base class for all external responses, inherited from the CreditServiceMessage class
- Credit Exception: This is the exception class that is used by the CreditService object. In addition to behaving as a regular exception class, it can hold the ISO response code, which is used by the CreditService object to send the appropriate ISO decline response
- IConnection interface: The CreditService object interacts with the External credit service provider through the IConnection interface object

Other classes

- ExternalSocketConnection: The IConnection interface object that communicates to an external credit service provider through sockets. It is inherited from SocketConnection. In addition to the functionality provided by the SocketConnection, it implements a reader thread which checks for response at regular time intervals and if so, stores the response in an internal queue. When the read method is called, a response is returned from the internal queue instead of directly reading from the socket connection.

  This is an abstract class. Users should provide and implementation of the readExternal method which reads from the input stream and returns a response in the form of bytes.
- RegisterMap: RegisterMap class is used for mapping clients register information with that of credit service provider.

  It contains the chain, store and terminal information of the clients and credit service provider.
- MerchantFactory: MerchantFactory is a singleton factory for producing RegisterMap objects from an XMLMerchant producer. The name of the XML merchant file should be provided.

Implementing a Credit Service APM

The generic credit service APM cannot be used as is and some implementation has to be provided for each credit service provider. The following approach should be followed:

- Create the external message types. See "Creating external message types" on page 172 for more information.
- Create a CreditServiceHandler object inherited from the BaseCreditServiceHandler class. Provide implementation for only those request and response methods that are supported by the credit service provider. See "Implementing the Credit Service Handler interface" on page 172 for more information.
- Create the IConnection object. See "Implementing the IConnection interface" on page 173 for more information.

Note: See "Configuring a Credit Service APM" on page 175 for information about how to configure a credit service APM.

Implementing the Credit Service Handler interface

- Implement the initialize method, if required.
- Implement the getCreditServiceResponse method. This method will be called when Credit Service receives a response from the IConnection object. The response will be in the form of bytes. This method discovers the type of response from the bytes and return an CreditServiceResponse object
- Based on what type of transactions need to be implemented by the credit service provider, only those methods need to be implemented.
- For example if credit authorization transactions are supported by the credit service provider, then only createExternalAuthorizationRequest and getISOAuthorizationResponse methods need to be implemented.
- ISO request and a register map information will be passed to all request methods and will return a CreditServiceRequest object
- Original ISO request, register map information and the Credit Service Response will be passed to all response methods and will return a ISO response.

Note: Please refer to "Transactions supported by the Credit Service APM" on page 177 for a list of transactions supported by the Credit Service APM and its corresponding methods.

Creating external message types

Each external request or response must have a message function, a message class, and a message type defined.

- If it is a request type, it should be inherited from CreditServiceRequest class. The message function is defined as a request function by the CreditServiceRequest class. Also, implement the respond method.
- If it is a response type, it should be inherited from CreditServiceResponse class. The message function is defined as a response function by the CreditServiceRequest class.
- Define the message class and message type for the object.
- Create the external message fields and getter and setter methods for these fields.

- Implement the read and write methods of the AbstractPersistentObject which should read and write in the format required by the credit service provider.
- Implement the getIdentifier method. It should return a identifier that is unique for the credit service being implemented. It is through using this identifier that CreditService gets the original credit request when it receives a response.

Implementing the IConnection interface

If using sockets, then ExternalSocketConnection object can be used. If so, then create a IConnection object inherited from ExternalSocketConnection and provide an implementation for the readExternalConnection method.

If using any other types of connection, then create a IConnection object. The following methods of the IConnection object has to be implemented:

- open(String connectionName)
- close
- writePersistent(IPersistentObject object)
- readObject(long timeoutMilis)

In the Credit Service Handler method getConnection, return the IConnection object just created above.

The following configurations in the credit.properties file are applicable to the IConnection object:

- ConnectionCount
- ConnectionName0
- ConnectionReadTimeOut
- TransactionLifeSpan

See "Configuring a Credit Service APM" on page 175 for details.

Note: The Credit Service opens the connection when it receives the first request and closes the connection if it receives an IOException from the IConnection object. Before closing the connection, it will make sure that all responses have been read from the IConnection object by calling the readObject method. If it returns null, then it assumes that there are no more responses and closes the connection. If it receives any request while it is closing the connection, it will send an issuer unavailable response.

Sending an echo request

- Implement the CreditServicehandler method createEchoRequest
- Set the property EchoTestInterval in the credit.properties file to the desired interval. The Credit Service will send the echo request after the specified interval.
- In the CreditServiceHandler method getCreditServiceResponse, make sure that the echo response is ignored by returning null.

Pre-processing a request

The Credit Service will pre-process only one type of request: the network sign-on request. When it receives a network sign-on request, it sends a response containing the terminal information configured for the credit service provider in the merchants XML file. It sends the credit name, debit name and external store id in a particular format.

Handling settlement and reconciliation requests

If the Credit Service receives a reconciliation / settlement request, it will forward the request to the "Settlement APM". The detail about the Settlement APM is beyond the scope of this document. If required, the Credit Service could be modified to send the reconciliation / settlement request to the Credit Service Handler. But as of now, this feature does not exist.

Handling settlement specific data

If there exists any settlement specific data (for example, any data required during settlement), then the data can be sent by using the setSupplemental method of ISO response.

Metering

The Credit Service logs the time when it sends a request and when it receives a response.

Logging

The Credit Service can log data received and sent, if configured to do so. The following are the objects it logs:

- ISO Message Request/Response objects received/sent from/to the internal message space. Log level =5. To enable this logging, CanLogOnGet and canLogOnPut properties in the credit.properties file will need to be set to true.
- Credit Service Request/ Responses objects received/ sent from/to the credit service provider. Log Level=0. To enable this logging, CanLogOnRead and canLogOnWrite properties in the credit.properties file will need to be set to true.
- Raw Bytes received from the credit service provider. Log Level=8. To enable this logging, CanLogOnRead in the credit.properties file will need to be set to true.
- The log threshold level for the Credit Service is the logThresholdLevel property in the credit.properties file. If the above mentioned log levels are greater that the log threshold level, then the data will not be logged.

Configuring a Credit Service APM

As with any existing APM, to configure a new credit service APM, you must add and configure a connection setting in the transnet.xml (server side) configuration file, and create and configure a merchant.xml file specific to the service.

Note: For more information about the generic APM configuration settings, please see the *Cetntralized EFT/File Transfer Techncial Reference Guide*.

The transnet.xml (server side) configuration file contains basic settings for your Centralized EFT server. This file is located by default at *root*:\Program Files\transnet\config on your Centralized EFT server machine. The following sample shows a sample set of tags for configuring a new credit service APM.

```
<Transnet SystemID="10_1_1">
<!-- SAMPLE SERVER SIDE TRANSNET.XML -->
    <Logging>logging.properties</Logging>
    <!-- The ConnectionManager Service tag set defines all of the
    connections used for your TMT message movement.
    Note that the forcedShutdownWaitTime attribute is measured in
    milliseconds. -->
    <Service name="ConnectionManager" description="Transnet Connection
    Manager"
    class="com.triversity.transnet.core.tms.connection.TNConnectionManager
    " forcedShutdownWaitTime="5000">
        <ActivationInterval value="10" unit="seconds" />

        ...

        <!-- For each generic financial APM or credit service provider you
        are using, you must have a Connection tag set in this file, as with
        the SampleAPMConnection section below.
        In each section, you must specify a name for the connection, as
        well as the general name for the EFT provider.
        You must also specify a Destination ID number (one for a single
        connection; successive numbers for each additional connection), and
        provide the IP address and port number (default 2000X) of the
        service provider. -->
        <Connection name="SampleAPMConnection"
        class="com.triversity.transnet.core.eft.EFTService"
        creditServiceHandlerClassName="com.triversity.transnet.xtension.cr
        edit.nova.NovaHandler" eftProviderName="Sample"
        transactionLifeSpan="10000">
            <APMConnection
            connectionClass="com.triversity.transnet.xtension.credit.nova.N
            ovaSocketConnection">
                <Destination id="1" name="127.0.0.1:20001" />
            </APMConnection>
        </Connection>

    ...

    </Service>

    ...
```

```
</Transnet>
```

A credit service mechant file also needs to be created. The name of the file should be *XXX*Merchant.xml, where *XXX* is the name of the service. For example, if the service provider was the “Associates”, the merchant file should be named AssociatesMerchant.xml.

Transactions supported by the Credit Service APM

The following table lists all transactions currently supported by the Credit Service APM. It also includes the corresponding Credit Service Handler methods that must be implemented to support each transaction, and the ISO message type used for creating the request or response:

Transactions	**Credit Service Handler Request Method** **ISO Request Type**	**Credit Service Handler Response Method** **ISO Response Type**
Authorizations (Credit and PrivateLabelCard)	CreateExternalAuthorizationRequest ISO8583AuthorizationRequest	GetISOAuthorizationResponse ISO8583AuthorizationRequestResponse
Balance Inquiry (PrivateLabelCard) (DEPRECATED see "ISO transaction type Identification" on page 178)	CreateExternalBalanceInquiryRequest ISO8583AdministrativeRequest	GetISOBalanceInquiryResponse ISO8583AdministrativeRequestResponse
Fund Inquiry (DEPRECATED see "ISO transaction type Identification" on page 178)	CreateExternalFundInquiryRequest ISO8583AdministrativeRequest	GetISOFundInquiryResponse ISO8583AdministrativeRequestResponse
Financial (Debit)	CreateExternalFinancialRequest ISO8583FinancialRequest	GetISOFinancialResponse ISO8583FinancialRequestResponse

ISO transaction type Identification

The following table lists the available ISO transaction types, together with all relevant ISO helper, message, processing code, and function code information:

Account type	**Transaction type**	**ISO8583 helper method**	**ISO message**	**Processing code**			**Function code**
				Txn Type	From Account	To Account	
Credit							
	Authorization	IsAuthorizationTxn	1100	00	00	Same as From	100
	Void Authorization	IsVoidTxn	1100	22	00	Same as From	100
	Return	isReturnTxn	1100	20	00	Same as From	100
	Void Return	isReturnVoidTxn	1100	02	00	Same as From	100
	PreAuthorization	isPreAuthorizationTxn	1100	00	00	Same as From	101
	PostAuthorization	isPostAuthorizationTxn	1100	00	00	Same as From	102
	Void PostAuthorization	isPostAuthorizationVoidTxn	1100	22	00	Same as From	102
Debit							
	Authorization	IsAuthorizationTxn	1200	00	10 / 20	Same as From	200
	Void Authorization	IsVoidTxn	1200	22	10 / 20	Same as From	200
	Return	isReturnTxn	1200	20	10 / 20	Same as From	200
	Void Return	isReturnVoidTxn	1200	02	10 / 20	Same as From	200
	Debit Cash back	isAuthorizationWithCashBackTxn	1200	09	10 / 20	Same as From	200
	Debit Second Request (Same as Acknowledgement)	isSecondRequestTxn	1200	17	10 / 20	Same as From	200
Check	Check Authorization	IsAuthorizationTxn	1200	04	00	Same as From	200

Private Label							
	Authorization	IsAuthorizationTxn	1100	00	90	Same as From	100
	Void Authorization	IsVoidTxn	1100	22	90	Same as From	100
	Return	isReturnTxn	1100	20	90	Same as From	100
	Void Return	isReturnVoidTxn	1100	02	90	Same as From	100
	Payment	isPaymentTxn	1100	50	90	Same as From	100
	Void Payment	isPaymentVoidTxn	1100	58	90	Same as From	100
	Balance Inquiry	isBalanceInquiryTxn	1100	31	90	Same as From	100
	PreAuthorization	isPreAuthorizationTxn	1100	00	90	Same as From	101
	PostAuthorization	isPostAuthorizationTxn	1100	00	90	Same as From	102
	Void PostAuthorization	isPostAuthorizationVoidTxn	1100	22	90	Same as From	102
Stored Value							
	Redeem	IsAuthorizationTxn	1100	00	91 / 92	Same as From	100
	Void Redeem	IsVoidTxn	1100	22	91 / 92	Same as From	100
	Return	isReturnTxn	1100	20	91 / 92	Same as From	100
	Void Return	isReturnVoidTxn	1100	02	91 / 92	Same as From	100
	Reload	isPaymentTxn	1100	50	91 / 92	Same as From	100
	Void Reload	isPaymentVoidTxn	1100	58	91 / 92	Same as From	100
	Balance Inquiry	isBalanceInquiryTxn	1100	31	91 / 92	Same as From	100
	Activate	isActivateTxn	1100	90	91 / 92	Same as From	100

	Void Activate	IsActivateVoidTxn	1100	91	91 / 92	Same as From	100
	Deactivate	IsActivateTxn	1100	92	91 / 92	Same as From	100
	Void Deactivate	IsDeActivateVoidTxn	1100	93	91 / 92	Same as From	100
	Replacement	isTransferTxn	1100	40	91 / 92	Same as From	100
	Void Replacement	isTransferVoidTxn	1100	48	91 / 92	Same as From	100
	CashOut	isCashOutTxn	1100	01	91 / 92	Same as From	100
	Void CashOut	isCashOutVoidTxn	1100	28	91 / 92	Same as From	100
	PreAuthorization	isPreAuthorizationTxn	1100	00	91 / 92	Same as From	101
	PostAuthorization	isPostAuthorizationTxn	1100	00	91 / 92	Same as From	102
	Void PostAuthorization	isPostAuthorizationVoidTxn	1100	22	91 / 92	Same as From	102
All	Acknowledgement (Same as Debit CashBack)	isAcknowledgementTxn	1100 / 1200	17	00	Same as From	100
All							
	Reversal Messages		1420	Original	Original	Original	Original
	SAF Messages		1120 / 1220	Original	Original	Original	Original

ISO definitions

```
public class ISODEF {

 //ISO message definition
 public final static short AUTH_REQ                    = 1100; // authorization request
 public final static short AUTH_REQ_REPEAT              = 1101; // authorization request repect
 public final static short AUTH_REQ_RESP          = 1110; // authorization request response
 public final static short AUTH_ADVICE                  = 1120; // authorization advice
 public final static short AUTH_ADVICE_REPEAT           = 1121; // authorization advice repect
 public final static short AUTH_ADVICE_RESP             = 1130; // authorization afvice response
 public final static short AUTH_NOTIFY                  = 1140; // authorization notification

 // financial messages
 public final static short FIN_REQ                = 1200; // financial request
```

```
public final static short FIN_REQ_REPEAT        = 1201; // financial request repect
public final static short FIN_REQ_RESP          = 1210; // financial request response
public final static short FIN_ADVICE                    = 1220; // financial advice
public final static short FIN_ADVICE_REPEAT             = 1221; // financial advice repect
public final static short FIN_ADVICE_RESP       = 1230; // financial advice response
public final static short FIN_NOTIFY                    = 1240; // financial notification

// file action messages
public final static short FILE_REQ              = 1304; // file action request
public final static short FILE_REQ_REPEAT       = 1305; // file action request repect
public final static short FILE_REQ_RESP         = 1314; // file action request response
public final static short FILE_ADVICE                   = 1324; // file action advice
public final static short FILE_ADVICE_REPEAT            = 1325; // file action advice repect
public final static short FILE_ADVICE_RESP      = 1334; // file action advice response
public final static short FILE_NOTIFY                   = 1344; // file action notification

// reversal/chargeback messages
public final static short REV_ADVICE                    = 1420; // reversal advice
public final static short REV_ADVICE_REPEAT             = 1421; // reversal advice repect
public final static short REV_ADVICE_RESP       = 1430; // reversal advice response
public final static short REV_ADVICE_NOTIFY             = 1440; // reversal notification
public final static short CHARGE_ADVICE         = 1422; // chargeback advice
public final static short CHARGE_ADVICE_REPEAT  = 1423; // chargeback advice repect
public final static short CHARGE_ADVICE_RESP            = 1432; // chargeback advice response
public final static short CHARGE_NOTIFY         = 1442; // chargeback advice notification

// reconciliation messages
public final static short ACQ_RECON_REQ         = 1500; // acquirer reconciliation request
public final static short ACQ_RECON_REQ_REPEAT  = 1501; // acquirer reconciliation request repect
public final static short ACQ_RECON_REQ_RESP            = 1510; // acquirer reconciliation request response
public final static short ACQ_RECON_ADVICE              = 1520; // acquirer reconciliation advice
public final static short ACQ_RECON_ADVICE_REPEAT       = 1521; // acquirer reconciliation advice repect
public final static short ACQ_RECON_ADVICE_RESP = 1530; // acquirer reconciliation advice response
public final static short ACQ_RECON_NOTIFY              = 1540; // acquirer reconciliation notification
public final static short ISSR_RECON_REQ        = 1502; // issuer reconciliation request
public final static short ISSR_RECON_REQ_REPEAT = 1503; // issuer reconciliation request repect
public final static short ISSR_RECON_REQ_RESP           = 1512; // issuer reconciliation request response
public final static short ISSR_RECON_ADVICE             = 1522; // issuer reconciliation advice
public final static short ISSR_RECON_ADVICE_REPEAT      = 1523; // issuer reconciliation advice repect
public final static short ISSR_RECON_ADVICE_RESP= 1532; // issuer reconciliation afvice response
public final static short ISSR_RECON_NOTIFY             = 1542; // issuer reconciliation notification

// administrative messages
public final static short ADMIN_REQ                     = 1604; // administrative request
public final static short ADMIN_REQ_REPEAT              = 1605; // administrative request repect
public final static short ADMIN_REQ_RESP        = 1614; // administrative request response
public final static short ADMIN_ADVICE          = 1624; // administrative advice
public final static short ADMIN_ADVICE_REPEAT           = 1625; // administrative advice repect
public final static short ADMIN_ADVICE_RESP             = 1634; // administrative advice response
public final static short ADMIN_NOTIFY          = 1644; // administrative notification

// fee collection messages
public final static short ACQ_FEE_ADVICE        = 1720; // acquirer fee collection advice
public final static short ACQ_FEE_ADVICE_REPEAT = 1721; // acquirer fee collection advice repect
public final static short ACQ_FEE_ADVICE_RESP           = 1730; // acquirer fee collection advice response
public final static short ACQ_FEE_NOTIFY        = 1740; // acquirer fee collection notification
public final static short ISSR_FEE_ADVICE       = 1722; // issuer fee collection advice
public final static short ISSR_FEE_ADVICE_REPEAT = 1723; // issuer fee collection advice repect
```

```
public final static short ISSR_FEE_ADVICE_RESP     = 1732; // issuer fee collection advice response
public final static short ISSR_FEE_NOTIFY          = 1742; // issuer fee collection notification

// network management messages
public final static short NET_MGMT_REQ             = 1804; // network management request
public final static short NET_MGMT_REQ_REPEAT           = 1805; // network management request repect
public final static short NET_MGMT_REQ_RESP             = 1814; // network management request response
public final static short NET_MGMT_ADVICE               = 1824; // network management advice
public final static short NET_NGMT_ADVICE_REPEAT= 1825; // network management advice repect
public final static short NET_MGMT_ADVICE_RESP  = 1834; // network management advice response
public final static short NET_MGMT_NOTIFY           = 1844; // network management notification

//ISO fields definition
public final static short BIT_SECONDARY_BITMAP = 1;
public final static short BIT_PRIMARY_ACCOUNT_NUMBER = 2;
public final static short BIT_PROCESSING_CODE = 3;
public final static short BIT_TRANSACTION_AMOUNT = 4;
public final static short BIT_RECONCILIATION_AMOUNT = 5;
public final static short BIT_BILLING_AMOUNT = 6;
public final static short BIT_TRANSMISSION_DATE_TIME = 7;
public final static short BIT_BILLING_FEE_AMOUNT = 8;
public final static short BIT_RECONCILIATION_CONVERSION_RATE = 9;
public final static short BIT_BILLING_CONVERSION_RATE = 10;
public final static short BIT_SYSTEM_TRACE = 11;
public final static short BIT_LOCAL_DATE_TIME = 12;
public final static short BIT_EFFECTIVE_DATE = 13;
public final static short BIT_EXPIRY_DATE = 14;
public final static short BIT_SETTLEMENT_DATE = 15;
public final static short BIT_CONVERSION_DATE = 16;
public final static short BIT_CAPTURE_DATE = 17;
public final static short BIT_MERCHANT_TYPE = 18;
public final static short BIT_ACQUIRER_COUNTRY_CODE = 19;
public final static short BIT_PAN_COUNTRY_CODE = 20;
public final static short BIT_FORWARD_COUNTRY_CODE = 21;
public final static short BIT_POS_DATA_CODE = 22;
public final static short BIT_CARD_SEQ_NUMBER = 23;
public final static short BIT_FUNCTION_CODE = 24;
public final static short BIT_MSG_REASON_CODE = 25;
public final static short BIT_ACCEPTOR_BUS_CODE = 26;
public final static short BIT_APPROVAL_CODE_LEN = 27;
public final static short BIT_RECONCILIATION_DATE = 28;
public final static short BIT_RECONCILIATION_INDICATOR = 29;
public final static short BIT_ORIGINAL_AMOUNT = 30;
public final static short BIT_ACQUIRER_REFERENCE_DATA = 31;
public final static short BIT_ACQUIRING_ID = 32;
public final static short BIT_FORWARDING_ID = 33;
public final static short BIT_EXTEND_PAN = 34;
public final static short BIT_TRACK2 = 35;
public final static short BIT_TRACK3 = 36;
public final static short BIT_RETRIEVAL_NUMBER = 37;
public final static short BIT_APPROVAL_CODE = 38;
public final static short BIT_ACTION_CODE = 39;
public final static short BIT_SERVICE_CODE = 40;
public final static short BIT_TERMINAL_ID = 41;
public final static short BIT_MERCHANT_ID = 42;
public final static short BIT_MERCHANT_NAME = 43;
public final static short BIT_ADDITIONAL_RESPONSE_DATA = 44;
public final static short BIT_TRACK1 = 45;
public final static short BIT_AMOUNT_FEE = 46;
```

```
public final static short BIT_ADDIDATA_NATIONAL = 47;
public final static short BIT_ADDIDATA_PRIVATE = 48;
public final static short BIT_TXN_CURRENCY_CODE = 49;
public final static short BIT_RECON_CURRENCY_CODE = 50;
public final static short BIT_BILLING_CURRENCY_CODE = 51;
public final static short BIT_ENCRYPTED_PIN = 52;
public final static short BIT_SECURITY_CONTROL_INFO = 53;
public final static short BIT_ADDITIONAL_AMOUNT = 54;
public final static short BIT_IC_CARD_DATA = 55;
public final static short BIT_ORIGINAL_DATA = 56;
public final static short BIT_LIFE_CYCLE_CODE = 57;
public final static short BIT_AUTH_AGENT_CODE = 58;
public final static short BIT_TRANSPORT_DATA = 59;
public final static short BIT_FIELD60 = 60;
public final static short BIT_FIELD61 = 61;
public final static short BIT_FIELD62 = 62;
public final static short BIT_FIELD63 = 63;
public final static short BIT_MAC1 = 64;

public final static short BIT_FIELD65 = 65;
public final static short BIT_ORIGINAL_FEE_AMOUNT = 66;
public final static short BIT_EXTEND_PAYMENT_DATE = 67;
public final static short BIT_RECEIVING_COUNTRY_CODE = 68;
public final static short BIT_SETTLEMENT_COUNTRY_CODE = 69;
public final static short BIT_AUTH_AGENT_COUNTRY_CODE = 70;
public final static short BIT_MESSAGE_NUMBER = 71;
public final static short BIT_DATA_RECORD = 72;
public final static short BIT_ACTION_DATE = 73;
public final static short BIT_CREDIT_NUMBER = 74;
public final static short BIT_CREDIT_REV_NUMBER = 75;
public final static short BIT_DEBIT_NUMBER = 76;
public final static short BIT_DEBIT_REV_NUMBER = 77;
public final static short BIT_TRANSFER_NUMBER = 78;
public final static short BIT_TRANSFER_REV_NUMBER = 79;
public final static short BIT_INQUIEY_NUMBER = 80;
public final static short BIT_AUTHORIZATION_NUMBER = 81;
public final static short BIT_INQUIRY_REV_NUMBER = 82;
public final static short BIT_PAYMENT_NUMBER = 83;
public final static short BIT_PAYMENT_REV_NUMBER = 84;
public final static short BIT_FEE_COLLECTION_NUMBER = 85;
public final static short BIT_CREDIT_AMOUNT = 86;
public final static short BIT_CREDIT_REV_AMOUNT = 87;
public final static short BIT_DEBIT_AMOUNT = 88;
public final static short BIT_DEBIT_REV_AMOUNT = 89;
public final static short BIT_AUTHORIZATION_REV_NUMBER = 90;
public final static short BIT_TXN_DEST_COUNTRY_CODE = 91;
public final static short BIT_TXN_ORIG_COUNTRY_CODE = 92;
public final static short BIT_TXN_DEST_ID = 93;
public final static short BIT_TXN_ORIG_ID = 94;
public final static short BIT_ISSUER_REFER_DATA = 95;
public final static short BIT_KEY_MGMT_DATA = 96;
public final static short BIT_NET_AMOUNT = 97;
public final static short BIT_PAYEE = 98;
public final static short BIT_SETTLEMENT_ID_CODE = 99;
public final static short BIT_RECEIVING_ID_CODE = 100;
public final static short BIT_FILE_NAME = 101;
public final static short BIT_ACCNT_ID1 = 102;
public final static short BIT_ACCNT_ID2 = 103;
public final static short BIT_TXN_DESCRIPTION = 104;
```

```
public final static short BIT_CREDIT_CHARGEBACK_AMOUNT = 105;
public final static short BIT_DEBIT_CHARGEBACK_AMOUNT = 106;
public final static short BIT_CREDIT_CHARGEBACK_NUMBER = 107;
public final static short BIT_DEBIT_CHARGEBACK_NUMBER = 108;
public final static short BIT_CREDIT_FEE_AMOUNT = 109;
public final static short BIT_DEBIT_FEE_AMOUNT = 110;
public final static short BIT_FIELD111 = 111;
public final static short BIT_FIELD112 = 112;
public final static short BIT_FIELD113 = 113;
public final static short BIT_FIELD114 = 114;
public final static short BIT_FIELD115 = 115;
public final static short BIT_FIELD116 = 116;
public final static short BIT_FIELD117 = 117;
public final static short BIT_FIELD118 = 118;
public final static short BIT_FIELD119 = 119;
public final static short BIT_FIELD120 = 120;
public final static short BIT_FIELD121 = 121;
public final static short BIT_FIELD122 = 122;
public final static short BIT_FIELD123 = 123;
public final static short BIT_FIELD124 = 124;
public final static short BIT_FIELD125 = 125;
public final static short BIT_FIELD126 = 126;
public final static short BIT_FIELD127 = 127;
public final static short BIT_MAC2 = 128;

public final static short BIT_PRIMARY_ACCOUNT_NUMBER_LENGTH = 130;//128+original number
public final static short BIT_ACQUIRER_REFERENCE_DATA_LENGTH = 159;
public final static short BIT_ACQUIRING_ID_LENGTH = 160;
public final static short BIT_FORWARDING_ID_LENGTH = 161;
public final static short BIT_EXTEND_PAN_LENGTH = 162;
public final static short BIT_TRACK2_LENGTH = 163;
public final static short BIT_TRACK3_LENGTH = 164;
public final static short BIT_MERCHANT_NAME_LENGTH = 171;
public final static short BIT_ADDITIONAL_RESPONSE_DATA_LENGTH = 172;
public final static short BIT_TRACK1_LENGTH = 173;
public final static short BIT_AMOUNT_FEE_LENGTH = 174;
public final static short BIT_ADDIDATA_NATIONAL_LENGTH = 175;
public final static short BIT_ADDIDATA_PRIVATE_LENGTH = 176;
public final static short BIT_SECURITY_CONTROL_INFO_LENGTH = 181;
public final static short BIT_ADDITIONAL_AMOUNT_LENGTH = 182;
public final static short BIT_IC_CARD_DATA_LENGTH = 183;
public final static short BIT_ORIGINAL_DATA_LENGTH = 184;
public final static short BIT_AUTH_AGENT_CODE_LENGTH = 186;
public final static short BIT_TRANSPORT_DATA_LENGTH = 187;
public final static short BIT_FIELD60_LENGTH = 188;
public final static short BIT_FIELD61_LENGTH = 189;
public final static short BIT_FIELD62_LENGTH = 190;
public final static short BIT_FIELD63_LENGTH = 191;
public final static short BIT_ORIGINAL_FEE_AMOUNT_LENGTH = 194;
public final static short BIT_DATA_RECORD_LENGTH = 200;
public final static short BIT_TXN_DEST_ID_LENGTH = 221;
public final static short BIT_TXN_ORIG_ID_LENGTH = 222;
public final static short BIT_ISSUER_REFER_DATA_LENGTH = 223;
public final static short BIT_KEY_MGMT_DATA_LENGTH = 224;
public final static short BIT_SETTLEMENT_ID_CODE_LENGTH = 227;
public final static short BIT_RECEIVING_ID_CODE_LENGTH = 228;
public final static short BIT_FILE_NAME_LENGTH = 229;
public final static short BIT_ACCNT_ID1_LENGTH = 230;
public final static short BIT_ACCNT_ID2_LENGTH = 231;
```

```
 public final static short BIT_TXN_DESCRIPTION_LENGTH = 232;
 public final static short BIT_CREDIT_FEE_AMOUNT_LENGTH = 237;
 public final static short BIT_DEBIT_FEE_AMOUNT_LENGTH = 238;
 public final static short BIT_FIELD111_LENGTH = 239;
 public final static short BIT_FIELD112_LENGTH = 240;
 public final static short BIT_FIELD113_LENGTH = 241;
 public final static short BIT_FIELD114_LENGTH = 242;
 public final static short BIT_FIELD115_LENGTH = 243;
 public final static short BIT_FIELD116_LENGTH = 244;
 public final static short BIT_FIELD117_LENGTH = 245;
 public final static short BIT_FIELD118_LENGTH = 246;
 public final static short BIT_FIELD119_LENGTH = 247;
 public final static short BIT_FIELD120_LENGTH = 248;
 public final static short BIT_FIELD121_LENGTH = 249;
 public final static short BIT_FIELD122_LENGTH = 250;
 public final static short BIT_FIELD123_LENGTH = 251;
 public final static short BIT_FIELD124_LENGTH = 252;
 public final static short BIT_FIELD125_LENGTH = 253;
 public final static short BIT_FIELD126_LENGTH = 254;
 public final static short BIT_FIELD127_LENGTH = 255;

 // ISO 8583(1993) action code - field 39
 // 000-099 Used in 1110,1120,1121,1140 and 1210,1220,1221 and 1240 messages to indicate
 //        that the transaction has been approved.
 public final static short RC_APPROVED                          = 000; // approved
 public final static short RC_APPROVED_WITH_ID                  = 001; // honour with identification
 public final static short RC_APPROVED_PARTIAL                  = 002; // approved for partial amount
 public final static short RC_APPROVED_VIP              = 003; // approved(VIP)
 public final static short RC_APPROVED_TRACK3                   = 004; // approved; update track 3
 public final static short RC_APPROVED_ACCT_SPEC= 005; // approved, account type specified by card issuer
 public final static short RC_APPROVED_PARTIAL_SPEC         = 006; // approved for partial amount; account type specified
by card issuer
 public final static short RC_APPROVED_ICC              = 007; // approved, update ICC
 public final static short RC_APPROVED_NEED_CNFM       = 80; // approved, but need confirmation(used for CIBC and NOVA
debit card processing mode

 // 100-199 Used in 1110,1120,1121,1140 and 1210,1220,1221 and 1240 messages to indicate
 //        that the transaction has been processed for authorization by or on behalf of
 //        the card issuer and has been denied(not requiring a card pick-up)
 public final static short RC_DECLINED_DO_NOT_HONOUR       = 100; // do not honour
 public final static short RC_DECLINED_EXPIRED_CARD         = 101; // expired card
 public final static short RC_DECLINED_SUSPECTED = 102; // suspected fraud
 public final static short RC_DECLINED_CONTACT_ACQ          = 103; // card acceptor contact acquirer
 public final static short RC_DECLINED_RESTRICTED= 104; // restricted card
 public final static short RC_DECLINED_CALL_ACQ     = 105; // card acceptor call acquirer's security department
 public final static short RC_DECLINED_PIN_EXCEED = 106; // allowable PIN tries exceeded
 public final static short RC_DECLINED_REFER_ISSR = 107; // refer to card issuer
 public final static short RC_DECLINED_REFER_ISSR_COND     = 108; // refer to card issuer's special conditions
 public final static short RC_DECLINED_INVALID_MERCHANT= 109; // invalid merchant
 public final static short RC_DECLINED_INVALID_AMOUNT       = 110; // invalid amount
 public final static short RC_DECLINED_INVALID_CARD         = 111; // invalid card number
 public final static short RC_DECLINED_PIN_REQUIRED= 112; // PIN data required
 public final static short RC_DECLINED_UNACCEPT_FEE         = 113; // unacceptable fee
 public final static short RC_DECLINED_ACCT_REQ    = 114; // no account of type requested
 public final static short RC_DECLINED_FUNC_NOT_SUPPORT= 115; // requested function not supported
 public final static short RC_DECLINED_NOT_SUFF_FUNDS       = 116; // not sufficient funds
 public final static short RC_DECLINED_INCORRECT_PIN        = 117; // incorrect PIN
 public final static short RC_DECLINED_NO_CARD_RECORD     = 118; // no card record
 public final static short RC_DECLINED_NOT_ALLOW_CARDHOLDER       = 119; // transaction not permitted to cardholder
```

```
public final static short RC_DECLINED_NOT_ALLOW_TERMINAL        = 120; // transaction not permitted to terminal
public final static short RC_DECLINED_EXCEED_AMOUNT_LIMIT= 121; // exceeds withdrawal amount limit
public final static short RC_DECLINED_VIOLATION    = 122; // security violation
public final static short RC_DECLINED_EXCEED_FREQ_LIMIT = 123; // exceeds withdrawal frequency limit
public final static short RC_DECLINED_VIOLATION_LAW      = 124; // violation of law
public final static short RC_DECLINED_NOT_EFFECTIVE      = 125; // card not effective
public final static short RC_DECLINED_INVALID_PIN = 126; // invalid PIN block
public final static short RC_DECLINED_PIN_LENGTH = 127; // PIN length error
public final static short RC_DECLINED_PIN_SYNC     = 128; // PIN key synch error
public final static short RC_DECLINED_SUSPECTED_COUNTER= 129; // suspected counterfeit card

// 200-299 Used in 1110,1120,1121,1140 and 1210,1220,1221 and 1240 messages to indicate
//       that the transaction has been processed for authorization by or on behalf of
//       the card issuer and has been denied requiring a card to be pick-up.
public final static short RC_PICKUP_DO_NOT_HONOUR        = 200; // do not honour
public final static short RC_PICKUP_EXPIRED_CARD= 201; // expired card
public final static short RC_PICKUP_SUSPECTED            = 202; // suspected fraud
public final static short RC_PICKUP_CONTACT_ACQ = 203; // card acceptor contact acquirer
public final static short RC_PICKUP_RESTRICTED     = 204; // restricted card
public final static short RC_PICKUP_CALL_ACQ             = 205; // card acceptor call acquirer's security department
public final static short RC_PICKUP_PIN_EXCEED     = 206; // allowable PIN tries exceeded
public final static short RC_PICKUP_SPEC_COND            = 207; // special condition
public final static short RC_PICKUP_LOST          = 208; // lost card
public final static short RC_PICKUP_STOLEN               = 209; // stolen card
public final static short RC_PICKUP_SUSPECTED_COUNTER  = 210; // suspected countfeid card

// 300-399 Used in 1314 1324,1325 and 1344 messages to indicate the result of the file action
public final static short RC_FILE_SUCCESS        = 300; // successful
public final static short RC_FILE_NOT_SUPPORT            = 301; // not supported by receiver
public final static short RC_FILE_UNABLE_LOCATE_RECORD= 302; // unable to locate record on file
public final static short RC_FILE_DUP_REPLACE            = 303; // duplicate record; old record replaced
public final static short RC_FILE_EDIT_ERROR             = 304; // field edit error
public final static short RC_FILE_LOCKED_OUT             = 305; // file locked out
public final static short RC_FILE_NOT_SUCCESS            = 306; // not successful
public final static short RC_FILE_FORMAT_ERROR    = 307; // format error
public final static short RC_FILE_DUP_REJECT             = 308; // duplicate; new record rejected
public final static short RC_FILE_UNKNOWN                = 309; // unknown file

// 400-499 Used in 1430,1432,1440 and 1442 messages to indicate the result of the
//       reversal or chargeback.
public final static short RC_REVERSAL_ACCEPT             = 400; // accepted

// 500-599 Used in 1510,1512,1530 and 1532 messages to indicate the result of a reconciliation.
public final static short RC_RECON_IN_BALANCE            = 500; // reconciled; in balance
public final static short RC_RECON_OUT_BALANCE  = 501; // reconciled; out balance
public final static short RC_RECON_AMOUNT_NOT_RECON     = 502; // amount not reconciled; total provided
public final static short RC_RECON_TOTAL_NOT_AVAILABLE= 503; // totals not avalable
public final static short RC_RECON_NOT_RECON             = 504; // not reconciled; totals provided

// 600-699 Used in 1614;1624;1625 and 1644 messages.
public final static short RC_ADMIN_ACCEPT        = 600; // accepted
public final static short RC_ADMIN_NOT_TRACE_ORIGIN      = 601; // not able to trace back original transaction
public final static short RC_ADMIN_INVALID_REFERENCE     = 602; // invalid reference number
public final static short RC_ADMIN_PAN_INCOMPATIBLE      = 603; // reference number/PAN incompatible
public final static short RC_ADMIN_PHOTO_NOT_AVAILABLE= 604; // POS photograph is not available
public final static short RC_ADMIN_ITEM_SUPP             = 605; // item supplied
public final static short RC_ADMIN_DOC_NOT_SUPP = 606; // request cannot be fulfilled-required/requested documentation
is not available
```

```
  // 700-799 Used in 1720;1721;1740;1722;1723 and 1742 messages.
  public final static short RC_FEE_ACCEPT                = 700; // accepted

  // 800-901 Used in 1814;1824;1825 and 1844 messages.
  public final static short RC_NETWORK_ACCEPT             = 800; // accepted
  public final static short RC_NETWORK_NO_LIABILITY= 900; // advice acknowledged; no financial liability accepted
  public final static short RC_NETWORK_LIABILITY    = 901; // advice acknowledged; financial liability accepted

  // 902-949 Used in request response and advice response messages to indicate transaction
  //        could not be processed.
  public final static short RC_REJECT_INVALID_TXN    = 902; // invalid transaction
  public final static short RC_REJECT_RE_ENTER_TXN= 903; // re-enter transaction
  public final static short RC_REJECT_FORMAT_ERROR        = 904; // format error
  public final static short RC_REJECT_ACQ_NOT_SUPP= 905; // acquirer not supported by switch
  public final static short RC_REJECT_CUTOVER_IN_PROCESS= 906; // cutover in process
  public final static short RC_REJECT_ISSUER_INOPERATIVE= 907; // card issuer or switch inoperative
  public final static short RC_REJECT_DEST_NOT_FOUND      = 908; // transaction destination cannot be found for routing
  public final static short RC_REJECT_SYSTEM_MALFUNCTION= 909; // system malfunction
  public final static short RC_REJECT_ISSUER_SIGNOFF      = 910; // card issuer signed off
  public final static short RC_REJECT_ISSUER_TIMEOUT      = 911; // card issuer timed out
  public final static short RC_REJECT_ISSUER_NOT_AVAILABLE= 912; // card issuer unavailable
  public final static short RC_REJECT_DUP_TRANSMISSION    = 913; // duplicate transmission
  public final static short RC_REJECT_NOT_TRACE_ORIGIN    = 914; // not able to trace back to original transaction
  public final static short RC_REJECT_CHECKPOINT_ERROR    = 915; // reconciliation cutover or checkpoint error
  public final static short RC_REJECT_MAC_ERROR           = 916; // MAC incorrect
  public final static short RC_REJECT_MAC_KEY_SYNC        = 917; // MAC key sync error
  public final static short RC_REJECT_NO_COMM_KEY= 918; // no communication keys available for use
  public final static short RC_REJECT_ENCRYPTION_KEY_SYNC= 919; // encryption key sync error
  public final static short RC_REJECT_SECURITY_ERROR_TRY_AGAIN    = 920; // security software/hardware error - try
again
  public final static short RC_REJECT_SECURITY_ERROR_NO_ACTION    = 921; // security software/hardware error - no
action
  public final static short RC_REJECT_MSGNO_ERROR= 922; // message number out of sequence
  public final static short RC_REJECT_REQ_IN_PROCESS      = 923; // request in progress

  // 950-999 Used in advice response(1x3x) to indicate the reason for rejection of the transfer
  //        of financial liability.
  public final static short RC_REJECT_VIOLATION           = 950; // violation of business arrangement

  public final static String getISOResponseCodeDescription(short isoResponseCode) {
    switch (isoResponseCode) {
      case ISODEF.RC_REJECT_SYSTEM_MALFUNCTION:
        return "System malfunction";
      case ISODEF.RC_DECLINED_FUNC_NOT_SUPPORT:
        return "Function not supported";
      case ISODEF.RC_REJECT_ISSUER_NOT_AVAILABLE:
        return "Issuer not available";
      case ISODEF.RC_REJECT_INVALID_TXN:
        return "Invalid transaction";
      default:
        return "Code="+isoResponseCode;
    }
  }

  // ISO 8583(1993) function code - field 24
  // 000-099 reserved for ISO use
```

```
// 100-199 Used in 1100;1101;1120;1121 and 1140 messages
public final static short FUNC_AUTH_ORIGAUTH_ACCUAMT    = 100; // original authorization - amount accurate
public final static short FUNC_AUTH_ORIGAUTH_ESTIAMT    = 101; // original authorization - amount estimated
public final static short FUNC_AUTH_REPLAUTH_ACCUAMT    = 102; // replacement authorization - amount accurate
public final static short FUNC_AUTH_REPLAUTH_ESTIAMT    = 103; // replacement authorization - amount estimated
public final static short FUNC_AUTH_RESUBM_ACCUAMT      = 104; // resubmission - amount accurate
public final static short FUNC_AUTH_RESUBM_ESTIAMT      = 105; // resubmission - amount estimated
public final static short FUNC_AUTH_SUPMAUTH_ACCUAMT    = 106; // supplementary authorization - amount accurate
public final static short FUNC_AUTH_SUPMAUTH_ESTIAMT    = 107; // supplementary authorization - amount estimated
public final static short FUNC_AUTH_INQUIRY             = 108; // inquiry

// 200-299 Used in 1200;1201,1220,1221 and 1240 message
public final static short FUNC_FIN_ORIGFIN_TXN      = 200; // original financial request/advice
public final static short FUNC_FIN_PREVAUTH_SAMEAMT     = 201; // previously approved authorization - amount same
public final static short FUNC_FIN_PREVAUTH_DIFFAMT     = 202; // previously approved authorization - amount differs
public final static short FUNC_FIN_RESUB_PREVFIN_DENY   = 203; // resubmission of a previously denied financial
request
public final static short FUNC_FIN_RESUB_PREVFIN_REVER= 204; // resubmission of a previously reversed financial trans-
action
public final static short FUNC_FIN_FIRST_REPRE      = 205; // first representment
public final static short FUNC_FIN_SECOND_REPRE = 206; // second representment
public final static short FUNC_FIN_THIRD_REPRE      = 207; // third or subsequent representment

// 300-399 Used in 1304,1305,1324,1325 and 1344 messages
public final static short FUNC_FILE_UNASSI          = 300; // unassigned
public final static short FUNC_FILE_ADD_RECORD      = 301; // addd record
public final static short FUNC_FILE_CHG_RECORD      = 302; // change record
public final static short FUNC_FLLE_DEL_RECORD      = 303; // delete record
public final static short FUNC_FILE_REP_RECORD      = 304; // replace record
public final static short FUNC_FILE_INQ             = 305; // inquiry
public final static short FUNC_FILE_REP_FILE            = 306; // replace file
public final static short FUNC_FILE_ADD_FILE            = 307; // add file
public final static short FUNC_FILE_DEL_FILE            = 308; // delete file
public final static short FUNC_FILE_CARD_ADMIN      = 309; // card administration

// 400-449 Used in 1420,1421 and 1440 messages to indicate the function of the reversal
public final static short FUNC_REV_FULL             = 400; // full reversal; transaction did not complete as approved
public final static short FUNC_REV_PART             = 401; // partial reversal; transaction did not complete for full amount

// 450-499 Used in 1422,1423 and 1442 messages to indicate the function of the chargeback
public final static short FUNC_CHARG_FIRST_FULL  = 450; // first chargeback; full
public final static short FUNC_CHARG_SECOND_FULL        = 451; // second chargeback; full
public final static short FUNC_CHARG_THIRD_FULL  = 452; // third or subsequent chargeback; full
public final static short FUNC_CHARG_FIRST_PART  = 453; // first chargeback; partial
public final static short FUNC_CHARGE_SECOND_PART       = 454; // second chargeback; partial
public final static short FUNC_CHARGE_THIRD_PART= 455; // third or subsequent; partial

// 500-599 Used in 1500,1501,1502,1503,1520,1521,1522,1523,1540 and 1542 messages
public final static short FUNC_RECON_FINAL              = 500; // final reconciliation
public final static short FUNC_RECON_CHK_POINT   = 501; // checkpoint reconciliation
public final static short FUNC_RECON_FINAL_CURR = 502; // final reconciliation in a specified currency
public final static short FUNC_RECON_CHK_POINT_CURR     = 503; // checkpoint reconciliation in a specified currency
public final static short FUNC_RECON_REQ            = 504; // request for reconciliation totals
public final static short FUNC_RECON_SETTLE             = 570; // request to settle
// 600-649 Used in 1604,1605,1624,1625 and 1644 messages for retrievals.
public final static short FUNC_ADMIN_ORIGRECP_RETRREQ  = 600; // original receipt, retrieval request
public final static short FUNC_ADMIN_ORIGRECP_RETRREQ_REPEAT  = 601; // original receipt, repeat retrieval request
public final static short FUNC_ADMIN_ORIGRECP_FULFILL     = 602; // original receipt, fulfillment
```

```
  public final static short FUNC_ADMIN_COPY_RETRREQ          = 603; // copy, retrieval request
  public final static short FUNC_ADMIN_COPY_RETRREQ_REPEAT          = 604; // copy, repeat retrieval request
  public final static short FUNC_ADMIN_COPY_FULFILL= 605; // copyn; fulfillment
  public final static short FUNC_ADMIN_VEHICLE                      = 606; // vehicle rental agreement
  public final static short FUNC_ADMIN_HOTEL                        = 607; // hotel charge detail
  public final static short FUNC_ADMIN_POS              = 608; // POS photograph
  public final static short FUNC_ADMIN_DEVY                         = 609; // proof of delivery
  public final static short FUNC_ADMIN_IMPRINT                      = 610; // imprint

  // 650-699 Used in 1604,1605,1624,1625 and 1644 messages for administrative messages.
  public final static short FUNC_ADMIN_NOT_PARSE   = 650; // unable to parse message

  // 700-799 Used in 1720,1721,1740,1722,1723 and 1742 messages.
  public final static short FUNC_FEE_COLL               = 700; // fee collection message
  public final static short FUNC_FEE_COLL_CANCEL   = 701; // fee collection cancellation, full/partial

  // 800-899 Used in 1804,1805,1824,1825 and 1844 messages.
  public final static short FUNC_NETWORK_SIGNON                     = 801; // system condition/sign-on
  public final static short FUNC_NETWORK_SIGNOFF  = 802; // system condition/sign-off
  public final static short FUNC_NETWORK_UNAVAIL   = 803; // system condition/target system unavailable
  public final static short FUNC_NETWORK_BACKUP                     = 804; // system condition/message originator's system in
backup
  public final static short FUNC_NETWORK_SPECIAL   = 805; // system condition/special instruction
  public final static short FUNC_NETWORK_ROUTE                      = 806; // system condition/initate alternate routing

  public final static short FUNC_NETWORK_KEYCHANGE          = 811; // system security/key change
  public final static short FUNC_NETWORK_ALERT                      = 812; // system security/security alert
  public final static short FUNC_NETWORK_PASSWDCHANGE   = 813; // system security/password change
  public final static short FUNC_NETWORK_DEVICE_AUTH        = 814; // system security/device authentication

  public final static short FUNC_NETWORK_CUTOVER = 821; // system accounting/cutover
  public final static short FUNC_NETWORK_CHK_POINT= 822; // system accounting/checkpoint

  public final static short FUNC_NETWORK_ECHO                       = 831; // system audit control/echo test

  // ISO 8583(1993) amount type codes - field ... ...
  // 00-19 account related balances
  // 00 reserved for ISO use
  public final static short AMT_LEDGER_BALANCE                      = 01; // account ledger balance
  public final static short AMT_AVAIL_BALANCE                       = 02; // account available balance
  public final static short AMT_OWING                               = 03; // account owning
  public final static short AMT_DUE                         = 04; // account due
  public final static short AMT_AVAIL_CREDIT                        = 05; // account available credit
  // 20-39 card related amounts
  public final static short AMT_REMAINING               = 20; // amount remaining this cycle
  // 40-59 transaction related amounts
  public final static short AMT_CASH                                = 40; // amount cash
  public final static short AMT_GOODS                               = 41; // amount goods and services

  // ISO 8583(1993) processing code - field 3(12)
  // 00-19 debits
  public final static short TPC_DB_GOODS_AND_SERVICE        = 00; // goods and service
  public final static short TPC_DB_CASH                             = 01; // cash
  public final static short TPC_DB_ADJUSTMENT                       = 02; // adjustment
  public final static short TPC_DB_CHEQUE_GUAR                      = 03; // cheque guarantee(funds guaranteed)
  public final static short TPC_DB_CHEQUE_VERI                      = 04; // cheque verification(funds available but not quaran-
teed)
```

```
public final static short TPC_DB_EURO_CHEQUE                = 05; // eurocheque
public final static short TPC_DB_TRAVEL_CHEQUE   = 06; // traveller cheque
public final static short TPC_DB_LETTER_CREDIT     = 07; // letter of credit
public final static short TPC_DB_GIRO                        = 8; // giro(postal banking)
public final static short TPC_DB_DISBURSE            = 9; // goods and services with cash disbursement
public final static short TPC_DB_NON_CASH                    = 10; // non-cash financial instrument(e.g.wire transfer)
public final static short TPC_DB_QUASI                = 11; // quasi-cash and scrip
public final static short TPC_DB_SECOND_REQUEST      = 17; // Tender Retail second request message
// 20-29 credits
public final static short TPC_CR_RETURE               = 20; // returns
public final static short TPC_CR_DEPOSIT              = 21; // deposits
public final static short TPC_CR_ADJUSTMENT                  = 22; // adjustment
public final static short TPC_CR_CHEQUE_GUAR                 = 23; // cheque deposit guarantee
public final static short TPC_CR_CHEQUE               = 24; // cheque deposit
public final static short TPC_CASH_ADJUSTMENT                = 28; // cash adjustment
// 30-39 inquiry services
public final static short TPC_FUND_INQUIRY                   = 30; // available funds inquiry
public final static short TPC_BALANCE_INQUIRY                = 31; // balance inquiry
// 40-49 transfer services
public final static short TPC_TRANSFER                = 40; // cardholder accounts transfer
public final static short TPC_TRANSFER_ADJUSTMENT        = 48; // void cardholder accounts transfer
// 50-59 payment services
public final static short TPC_PAYMENT                        = 50; // payment
public final static short TPC_PAYMENT_ADJUSTMENT         = 58; // payment adjustment
// 90-99 reserved for private use
public final static short TPC_ACTIVATE            = 90; // account activation
public final static short TPC_ACTIVATE_ADJUSTMENT         = 91; // void account activation
public final static short TPC_DEACTIVATE        = 92; // account deactivation
public final static short TPC_DEACTIVATE_ADJUSTMENT      = 93; // void account deactivation


// ISO 8583 (1993) account type - field 3(34/56)
public final static short ACCNT_DEFAULT              = 00; // default - unspecified
public final static short ACCNT_SAVING               = 10; // saving account - default
public final static short ACCNT_CHEQUE               = 20; // cheque account - default
public final static short ACCNT_CREDIT               = 30; // credit facility - default
public final static short ACCNT_UNIVERSAL                    = 40; // universal account - default
public final static short ACCNT_INVESTMENT                   = 50; // investment account - default
public final static short ACCNT_PRIVATELABEL      = 90; // private label card
public final static short ACCNT_STOREVALUE        = 91; // store value card
public final static short ACCNT_GIFT           = 92; // gift card

// ISO Card Data Input Capability
public final static short  CARD_CAPTURE_UNKNOWN     = (short)'0'; //
public final static short  CARD_CAPTURE_MANUAL      = (short)'1';
public final static short  CARD_CAPTURE_SWIPED      = (short)'2';
public final static short  CARD_CAPTURE_SCANNED     = (short)'3';
public final static short  CARD_CAPTURE_OCR         = (short)'4';
public final static short  CARD_CAPTURE_ICC         = (short)'5';
public final static short  CARD_CAPTURE_KEYED       = (short)'6';
public final static short  CARD_CAPTURE_RESERVED    = (short)'7';

}
```

7 Configuring the APM XML Files

This section provides instructions for configuring the Application Processing Modules (APMs) XML files. Topics in this section include:

Generic configuration file settings

For every APM, a series of similar settings must be configured. This section outlines the configurable portions of the XML configuration files required to set up each APM. Note that only the settings specifically described in this section should be altered. No other settings should be changed from their default values without first consulting your SAP representative. For information regarding settings specific to each APM, see the relevant sections in "About the individual APMs" on page 199.

All XML configuration files are located (by default) at X:\Program Files\transnet\config\ where X is the drive on which you installed your Centralized EFT server. The generic files described are as follows:

Note: These files should generally be configured in the order in which they are listed above, with any additional APM-specific files following.

---

transnet.xml — Centralized EFT server version

A section of a transnet.xml file must be configured at your Centralized EFT server. This file is located by default at *root*:\Program Files\transnet\config on your Centralized EFT server machine.

Note: The generic settings shown here apply to the following APMs: FNMS, Nova, WildCard. All other APMs have specific settings, described in the relevant section under "About the individual APMs" on page 199.

In this file you must configure a Connection section for each APM or service provider you have installed. (Note that the default information this file contains is dependent upon the information and settings you selected when installing your components.)

Look for a tag section in this file similar to the following:

```
<Transnet SystemID="10_1_1">
   <Logging>logging.properties</Logging>
   <Service name="ConnectionManager" description="Transnet Connection
   Manager"
   class="com.triversity.transnet.core.tms.connection.TNConnectionManager
   " forcedShutdownWaitTime="5000">
      <ActivationInterval value="10" unit="seconds" />


      ...


      <!-- For each APM or service provider you are using, you must have
      a Connection tag set in this file.
      In each section, you must specify a name for the connection, as
      well as the general name for the EFT provider.
      You must also specify a Destination ID number (one for a single
      connection; successive numbers for each additional connection), and
      provide the IP address and port number (default 2000X) of the
      service provider. -->
      <Connection name="NovaAPMConnection"
      class="com.triversity.transnet.core.eft.EFTService"
      creditServiceHandlerClassName="com.triversity.transnet.xtension.cr
      edit.nova.NovaHandler" eftProviderName="Nova"
      transactionLifeSpan="10000">
         <APMConnection
         connectionClass="com.triversity.transnet.xtension.credit.nova.N
         ovaSocketConnection">
            <Destination id="1" name="127.0.0.1:20001" />
         </APMConnection>
      </Connection>
```

Generic transnet.xml settings

This section describes each generic tag setting used in the APM settings sections of the transnet.xml server side configuration file.

Note: If you require assistance with these settings, please contact your SAP representative.

Tag	Attributes	Set to	Notes
<Connection>			Depending on which APMs you have installed, the appropriate Connection sections should appear in your transnet.xml server side configuration file.If no relevant section exists, you must add it. Contact your SAP representative for assistance
	Name	The name of your APM connection. For example, NovaAPMConnection	Generally set to *XXX*APMConnection, where *XXX* is the name of the service provider. Leave default value
	Class	For example, com.Triversity.transnet.core.eft.EFTService	Leave default value
	creditServerhandlerClassName	For example, com.triversity.transnet.xtension.credit.nova.NovaHandler	Leave default value
	eftProviderName	For example, Nova	Generally set to the name of the service provider. Leave default value
	transactionLifeSpan	For example, 10000	
<APMConnection> This subtag is located within a specific APM <Connection> tag section			
	connectionClass	For example, com.triversity.transnet.xtension.credit.nova.NovaSocketConnection	Leave default value
	<Destination> This subtag is located within a specific APM <APMConnection> tag section		Defines the destination for all messages sent to this service provider through this connection

Tag	Attributes	Set to	Notes
	id	The identification code for the service provider. For example, 1.	
	name	The IP address and port for the service provider. For example, 127.0.0.1:20001	The default port used for connecting to the various service providers is 2000*X* (for example, 20001, 20002, and so forth, for each additional APM connection).

transnet.xml — Tomcat version

A section of a transnet.xml file must be configured at your Web server (Tomcat). This file is located by default at *root*:\Program Files\transnet\tomcat\webapps\Transnet on any machine on which you have installed the Web server applications (Tomcat).

In this file you must simply add, edit, or verify the correct names for your APM Merchant.xml files. (Note that the default information this file contains is dependent upon the information and settings you selected when installing your components.) Look for a tag section in this file similar to the following:

```
<?xml version="1.0" ?>
<TransnetServerConfiguration>
    <MeterTable Name="transnet_meter" />
    <!-- The MerchantFile tags specify the APMs or service providers you
    are using, and indicate the naming conventions for each corresponding
    XXXMerchant.xml configuration file. The names of your XXXMerchant.xml
    files must correspond to those listed here exactly.-->
    <MerchantFile>name="NovaMerchant.xml" service="Nova"</MerchantFile>
    <MerchantFile>name="WildCardMerchant.xml" service="WildCard"</
    MerchantFile>
    <MerchantFile>name="FNMSMerchant.xml" service="FNMS"</MerchantFile>
```

If no **<MerchantFile>** tag set appears for the APM you are using, add one to the file, and save your changes. If you require assistance, consult your SAP representative.

Note: This file is also used to set Mismatch settlement and reconciliation reporting information for your system. For more information, consult the *Centralized EFT User Guide*.

Merchant.xml

A Merchant.xml file must be configured to run any of the APM-supported financial services (for example, credit, debit, settlement, check verification, and so forth), with the exception of the Tender Retail-based services (such as GPS Canada and GPS USA) which use a different file set for the same information. (See "About the individual APMs" on page 199 for detailed information.)

The Merchant.XML file contains basic information pertaining to each service you use as it relates to each chain, store, and register (or terminal) in your retail hierarchy. You must enter information at each of three tag set levels (to correspond with your retail chain, store, and register levels) in this XML file, so that there is an entry for each chain/store/register combination.

Note that this process can be lengthy, but need be performed only once, when you are first setting up your system. After that, you will modify this file only when adding a new chain, store, or register, or to edit information for an existing entity.

For each service you use (each financial service APM you have installed), you must have a separate Merchant.xml file. To achieve this, when beginning your configuration, you must copy the generic Merchant.xml file and rename it for each service provider with the service name preceding the original name (for example, the merchant file for a service provider named "ABC Credit" would be named ABCCreditMerchant.xml). You can then proceed to configure each custom file as required.

Merchant.xml settings

This section describes each tag setting used in the Merchant.xml configuration file.

Note: Much of the information required for the settings in this file must be obtained from your credit service provider prior to beginning the APM configuration process. Such information is flagged in the following table. If you require assistance, please contact your SAP representative.

Tag set/hierarchy level	Tag	Set to	Notes
<Chain>	<Chain ID>	Chain Number For example, 1	Numeric data — value may be from 1 to 999 Used for online transaction processing
	<Name>	Chain Name For example, OurChain	
	<Contact>	Principle contact information for this chain For example, John Smith, OurChain, 41 Main St., Newville, Tel 416-234-5678	
	<BusinessCutoverTime>	Time to reset to new business day, in HHMMSS format For example, 000000	The default is midnight

Tag set/hierarchy level	Tag	Set to	Notes
<Store> A Store tag set must be created for each store within the above defined chain.	<StoreId>	Store identification number For example, 1	Numeric data — value may be from 1 to 99999 Used for online transaction processing
	<ExtStoreId>	Service Provider external store identification number For example, 0001	Used to identify a store to the service provider. Obtain this information from your service provider's contact.
	<ServiceType>	Service Provider service type For example, PS2000	Used to identify which service package this store uses. For example: Normal PS2000 VISA PS2000 Obtain this information from your service provider's contact.
	<Name>	Unique name for this store For example, Newville store	General information used for data settlement.
	<City>	City name for location of this store For example, Newville	General information used for data settlement.
	<State>	State or province name for location of this store For example, New York	General information used for data settlement.
	<ZIPCode>	Zip code or postal code for location of this store For example, 34001	General information used for data settlement.
	<Account>	Service Provider account number For example, 87891236666	Assigned by your service provider. Obtain this information from your service provider's contact. Used for data settlement
	<SecurityCode>	Service Provider security code For example, 5555	Assigned by your service provider. Obtain this information from your service provider's contact. Used for data settlement.

Tag set/hierarchy level	Tag	Set to	Notes
	<ClearingCode>	Service Provider clearing code For example, 6666	Assigned by your service provider. Obtain this information from your service provider's contact. Used for data settlement.
	<ExtAccount>	Service Provider external account number For example, 666641238789	Assigned by your service provider. Obtain this information from your service provider's contact. Used for data settlement.
	<BINNumber>	Service Provider BIN number For example, 4012	Assigned by your service provider. Obtain this information from your service provider's contact. Used for data settlement.
	<ICANumber>	Service Provider ICA number For example, 2343	Assigned by your service provider. Obtain this information from your service provider's contact. Used for data settlement.
	<CategoryCode>	ISO category code For example, 5412	This is a four-digit number assigned by ISO, and used to identify the industry to which the store belongs. For example, the code 5814 indicates that the store is a fast food restaurant.
	<CheckProcessor>	Check processor ID For example, 000000000001	If this store provides check verification services, this parameter identifies which check processor is used. For example: TELECHECK EQUIFAX NPCJBS ETC
	<Contact>	Principle contact information for this store For example, Jenny West, Newville store, 99 Union St, Newville, Tel 416-791-7100	

Tag set/hierarchy level	Tag	Set to	Notes
	<BusinessCutoverTime>	Time to reset to new business day, in HHMMSS format For example, 235959	The Chain level setting is used if no setting is defined at the Store level The default is midnight
<Register> A Register tag set must be created for each register or terminal within the previously-defined store.	<RegisterId>	The number of the register or terminal, as defined by the service provider For example, 88	This number must match the register number defined in your POS product. Obtain this information from your service provider contact.
	<CreditName>	The name of the register or terminal for credit service use, as defined by the service provider For example, ABCCredit1	Obtain this information from your service provider's contact.
	<DebitName>	The name of the register or terminal for debit service use, as defined by the service provider For example, ABCCredit12	Obtain this information from your service provider's contact.

About the individual APMs

This section provides specific information and configuration requirements for each of the APMs included with a standard install of the Centralized EFT/File Transfer. Topics in this section include:

ADS

The ADS APM provides a TCP/IP inteface to the ADS provider. It supports credit and debit but no settlement.

Configuration

To configure the ADS APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)

ADS specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the ADS APM.

```
- <Connection name="ADS"
class="com.triversity.transnet.core.eft.EFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.ad
siso.ADSISOHandler" eftProviderName="ADS">
- <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.adsiso.ADSISOSoc
ketConnection" echoTestIntervalSeconds="300000"
delegateWriteToConnection="true">
  <Destination id="1" name="Connection ip:port" connectTimeoutSeconds="25"
/>
  </APMConnection>
  </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

Allegiance 1-to-1 APM

The Allegiance 1-to-1 APM provides the connectivity required to perform centralized loyalty program tracking and authorization using the SAP Allegiance CRM system via Centralized EFT.

Configuration

To configure the Allegiance 1-to-1 APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT client and server version (specific settings, outlined below)

Allegiance 1-to-1- specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the Allegiance 1-to-1 APM.

At Centralized EFT Client

At Centralized EFT Client, the config/transnet.xml should contain the followings:

- The message descriptor for A1to1 message:

```
<MessageDescriptor name="A1to1Message" root="methodCall">
 <Check xmlType="text" name="methodName" dataType="text" substring="0,5">
  <DataText value="a1to1"/>
 </Check>
</MessageDescriptor>
```

- The routing rule for Allegiance1-to-1 messages:

```
<RoutingRule message="A1to1Message">
 <Route name="A1to1MessageRoute" timeoutSeconds="30"/>
</RoutingRule>
```

- The route for Allegiance1-to-1 messages:

```
<Route name="A1to1MessageRoute" connectionName="TPSClient"/>
```

At Centralized EFT Server

At Centralized EFt Server, the config/transnet.xml should contain the followings:

- A connection to the Allegiance1-to-1 server:

```
<Connection name="Allegiance1To1"

class="com.triversity.transnet.core.tms.connection.TNMessageConsumerConne
ction" >
   <adapter
class="com.triversity.transnet.xtension.allegiance.AllegianceAdapter"
       encoding="UTF-8"
       serverName="//YourAllegiance1to1/a1to1XML">
       <Pool minimumSize="1"
           maximumSize="1000"
           poolCleaningIntervalMillis="1000"
           allowedIdleTime="60000">
       </Pool>
   </adapter>

</Connection>
```

Note: Replace **YourAllegiance1to1** in the above example with the IP or hostname of the Allegiance 1-to-1 server in your environment.
Also, the line **encoding=** may have the value of **UTF-8** or **UTF=16**.

- The message descriptor for Allegiance1-to-1 message:

```
<MessageDescriptor name="A1to1Message" root="methodCall">
 <Check xmlType="text" name="methodName" dataType="text" substring="0,5">
  <DataText value="a1to1"/>
 </Check>
</MessageDescriptor>
```

Note: The message descriptor is the same as the Centralized EFT Client.

- The routing rule for Allegiance1-to-1 messages:

```
<RoutingRule message="A1to1Message">
 <Route name="A1to1MessageRoute" timeoutSeconds="30"/>
</RoutingRule>
```

Note: The routing rule is the same as the Centralized EFT Client.

- The route for A1to1 messages:

```
<Route name="A1to1MessageRoute" connectionName="Allegiance1To1"/>
```

Note: The route is NOT the same as the Centralized EFT Client, it connects to "**Allegiance1To1**" which was the connection defined above.

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

AMEX

The American Express (AMEX) APM provides a dial-up interface to the American Express provider. It supports only authorization of American Express credit cards. It is designed for use on an in-store Centralized EFT server configuration. It currently uses ModemConnection as its dial-up interface, but should be ported to SequentialAPMConnection for future use.

Configuration

To configure the AMEX APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

AMEX specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the AMEX APM.

```
- <Connection name="Amex"
class="com.triversity.transnet.core.apm.APMConnectionHandler">
- <APM name="Amex" maximumCount="1" minimumCount="1"
class="com.triversity.transnet.core.credit.CreditService"
connectionClass="com.triversity.transnet.xtension.credit.amex.AmexModemCo
```

```
nnection"
CreditServiceHandlerClassName="com.triversity.transnet.xtension.credit.am
ex.AmexHandler" TransactionLifeSpan="10000">
  <AllowdIdleTime value="1" unit="hour" MerchantsFile="AmexMerchant" />
  <Connection id="2" name="Amex" PhoneNumber="9,18002281082"
ModemConfig="config\modem_dialer.xml" />
  </APM>
  </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

Certegy

The Certegy APM provides a dial-up interface to the Certegy provider. It supports only check verification. It currently uses SequentialAPMConnection and is for use on an in-store Centralized EFT server configuration.

Configuration

To configure the Certegy APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

Certegy specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the Certegy APM.

```
- <Connection name="Certegy"
class="com.triversity.transnet.core.eft.SynchronousEFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.ce
rtegy.CertegyHandler" eftProviderName="Certegy"
transactionLifeSpan="95000" reuseConnection="false">
- <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.modem.ModemConne
ction">
- <Device
class="com.triversity.transnet.xtension.credit.modem.ModemDevice">
  <PhoneNumber>9,18004161282</PhoneNumber>
  <ModemConfig>config\modem_dialer.xml</ModemConfig>
  </Device>
  <Protocol
class="com.triversity.transnet.xtension.credit.certegy.CertegyModemProtoc
ol" />
  <Destination id="1" name="Certegy" />
  <Connection id="1" name="Certegy Authorization" />
  </APMConnection>
  </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

ConcordEFS

The ConcordEFS APM provides TCP/IP and dial-up intefaces to the Concord provider. It supports credit, debit, EBT, gift card and checks support but no settlement. It supports authorizations, returns,

voids and reversals for credit, debit and EBT, activation for gift cards and balance inquiries for EBT and gift cards.

Configuration

To configure the ConcordEFS APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

ConcordEFS specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the ConcordEFS APM.

```
- <Connection name="ConcordEFSAPM"
class="com.triversity.transnet.core.apm.APMConnectionHandler">
- <APM name="ConcordEFS" maximumCount="1" minimumCount="1"
class="com.triversity.transnet.core.credit.CreditService"
connectionClass="com.triversity.transnet.xtension.credit.concordEFS.Conco
rdEFSSocketConnection"
CreditServiceHandlerClassName="com.triversity.transnet.xtension.credit.co
ncordEFS.ConcordEFSHandler" TransactionLifeSpan="15000"
ConnectionReadTimeOut="2000">
  <AllowdIdleTime value="1" unit="hour" MerchantsFile="ConcordMerchant" />
  <Connection id="2" name="127.0.0.1:5350" />
  </APM>
  </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

Concord Services APM

The Concord Services APM provides the connectivity required to perform credit authorization and settlement services from the Concord service provider through Centralized EFT.

The services the Concord Services APM provide include:

Online transactions

- Credit Card Authorization: Supported cards include VISA, MasterCard, AMEX, Diners Club, Discover, JCB, and Carte Blanche. Possible credit card transactions include sales and authorizations, returns, and voids.

Settlement

- Provides credit card authorization data settlement with Concord Services. The system generates a batch settlement file in accordance with the daily online transactions, and transmits it to the Concord Services settlement center for completion.

Configuration

The Concord Services APM is based upon the Generic Credit APM. As such, to configure the Concord APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- transnet.xml — Tomcat version
- Merchant.xml

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

Concord-specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the Concord APM.

```
<Transnet SystemID="10_1_1">
   <Logging>logging.properties</Logging>
   <Service name="ConnectionManager" description="Transnet Connection
   Manager"
   class="com.triversity.transnet.core.tms.connection.TNConnectionManager
   " forcedShutdownWaitTime="5000">
      <ActivationInterval value="10" unit="seconds" />


      ...


      <Connection name="ConcordAPMConnection"
      class="com.triversity.transnet.core.finance.xml.XMLFinancialMessag
      eHandler" />
```

Datamark

The Datamark APM provides HTTP and dial-up interface to the Datamark provider. It supports only gift cards and no settlement. It supports activations, redemptions, balance inquiries and voids.

Configuration

To configure the Datamark APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

Datamark specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the Datamark APM.

```
<!-- Datamark APM using dial-up connection -->
      <Connection name="Datamark"
class="com.triversity.transnet.core.eft.SynchronousEFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.Da
tamark.DatamarkHandler" eftProviderName="Datamark"
transactionLifeSpan="10000" reuseConnection="false">
         <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.modem.ModemConne
ction">
            <Device
class="com.triversity.transnet.xtension.credit.modem.ModemDevice">
               <PhoneNumber>phone number</PhoneNumber>
               <PhoneNumber>alternate phone number</PhoneNumber>
               <ModemConfig>config\modem_dialer.xml</ModemConfig>
            </Device>
            <Protocol
class="com.triversity.transnet.xtension.credit.modem.ModemProtocol"/>
            <Destination id="1" name="Datamark"/>
```

```
            <Connection id="1" name="Datamark Authorization"/>
         </APMConnection>
      </Connection>


   <!-- Datamark APM using HTTP connection -->
      <Connection name="Datamark"
class="com.triversity.transnet.core.eft.SynchronousEFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.Da
tamark.DatamarkHandler" eftProviderName="Datamark"
transactionLifeSpan="10000" reuseConnection="false">
         <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.Datamark.Datamar
kURLConnection">
            <Destination id="1" name="url connection address"
connectTimeoutSeconds="25" userName="username" password="password"/>
         </APMConnection>
      </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

Discover

The Discover APM provides a dial-up interface to the Discover provider. It supports only Authorization of Discover credit cards. It is designed for use on an in-store Centralized EFT server configuration. It currently uses ModemConnection as its dial-up interface, but should be ported to SequentialAPMConnection for future use.

Configuration

To configure the Discover APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

Discover specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the Discover APM.

```
- <Connection name="Discover"
class="com.triversity.transnet.core.apm.APMConnectionHandler">
- <APM name="Discover" maximumCount="1" minimumCount="1"
class="com.triversity.transnet.core.credit.CreditService"
connectionClass="com.triversity.transnet.xtension.credit.modem.ModemConne
ction"
CreditServiceHandlerClassName="com.triversity.transnet.xtension.credit.Di
scover.DiscoverHandler" TransactionLifeSpan="10000">
  <AllowdIdleTime value="1" unit="hour" MerchantsFile="DiscoverMerchant" /
>
  <Connection id="2" name="Amex" PhoneNumber="9,18002281082"
ModemConfig="config\modem_dialer.xml" />
  </APM>
  </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

EFunds

The Efunds APM provides TCP/IP interface for check verifications.

Configuration

To configure the Efunds APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

Efunds specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the Efunds APM.

```
- <Connection name="eFunds"
class="com.triversity.transnet.core.eft.EFTService"
recordAllTransactions="true"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.eF
unds.eFundsHandler" replyListName="eFunds" replyTimeoutValue="60000"
eftProviderName="eFunds">
- <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.eFunds.eFundsSoc
ketConnection" echoTestIntervalSeconds="300000"
delegateWriteToConnection="true">
  <Destination id="1" name="connection ip:port" connectTimeoutSeconds="25"
/>
  </APMConnection>
  </Connection>
```

For more information about the generic APM configuration files, see “Generic configuration file settings” on page 191.

FDMS

The FDMS APM provides TCP/IP and dial-up interfaces for credit-cards and check verification. Store-based batch settlement is provided through Centralized EFT.

Configuration

To configure the FDMS APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

FDMS specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the FDMS APM.

```
   <!-- fdms socket connection -->
      <Connection name="FDMSSocketAPM"
class="com.triversity.transnet.core.eft.EFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.fd
ms.FdmsHandler" eftProviderName="FDMS" transactionLifeSpan="40000"
sequential="true">
         <APMConnection
connectionClass="com.triversity.transnet.core.apm.SequentialAPMConnection
" Name="Conn1">
           <Device class="com.triversity.transnet.core.apm.SocketDevice">
              <Destination host="host IP address" port="host port"
connectTimeout="1000"/>
            </Device>
            <Protocol
class="com.triversity.transnet.core.apm.SocketProtocol" MaxTrxTime="35"/>
         </APMConnection>
      </Connection>


   <!-- fdms dialup connection -->
      <Connection name="FDMSModemAPM"
class="com.triversity.transnet.core.eft.EFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.fd
ms.FdmsHandler" eftProviderName="FDMS" transactionLifeSpan="80000"
sequential="true">
         <APMConnection
connectionClass="com.triversity.transnet.core.apm.SequentialAPMConnection
" Name="Conn2">
            <Device class="com.triversity.transnet.core.apm.ModemDevice">
               <PhoneNumber retry="1">phone number</PhoneNumber>
              <PhoneNumber retry="1">alternate phone number</PhoneNumber>
            </Device>
            <Protocol
class="com.triversity.transnet.xtension.credit.fdms.FdmsModemProtocol"
MaxTrxTime="75"/>
         </APMConnection>
      </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

First Data North APM

The First Data North APM provides the connectivity required to perform credit authorization and settlement services from the First Data North service provider through Centralized EFT.

The services the First Data North APM provides include:

Online transactions

- Credit Card Authorization: Supported cards include VISA, MasterCard, AMEX, Diners Club, Discover, JCB, and Carte Blanche. Possible credit card transactions include sales and authorizations, returns, and voids.
- Debit Card Purchase: Service allows all debit cards supported by the First Data North services network. Possible debit card transactions include sales and authorizations, returns, and voids.
- Check Verification: The APM provides a service interface with three major check verification/check guarantee processors: NPC (JBS), Equifax/TeleCredit, and TeleCheck. ETC/SCAN check verification is also supported. Possible transactions include check verification and check guarantee.

Settlement

- Provides credit card authorization data settlement with First Data North. The system generates a batch settlement file in accordance with the daily online transactions, and transmits it to the First Data North settlement center for completion.

Configuration

The First Data North APM is based upon the Generic Credit APM. As such, to configure the First Data North APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- transnet.xml — Tomcat version
- Merchant.xml

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

FD North-specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the First Data North APM.

```
<Transnet SystemID="10_1_1">
   <Logging>logging.properties</Logging>
   <Service name="ConnectionManager" description="Transnet Connection
   Manager"
   class="com.triversity.transnet.core.tms.connection.TNConnectionManager
   " forcedShutdownWaitTime="5000">
      <ActivationInterval value="10" unit="seconds" />


      ...


      <!-- In the FDNorth APM Connection settings, the
      transactionLifeSpan setting (measured in milliseconds) indicates
      the duration of time after which a request is returned to Transnet
      if no response is generated at the service provider. This setting
      should be slightly less (-1000 milliseconds) than the corresponding
```

```
setting configured at the client, which in turn should be less than
that configured in the Route Definition, then the Routing Rule,
then the Client.conf and ClientAPI.conf files, and finally the
Transactionware GM Xpress server POS.ini file.

The minimumCount and maximumCount attributes denote the number of
connections; this should be determined by the size of your system
and the volume of data expected. -->
<Connection name="FDNorth"
class="com.triversity.transnet.core.apm.APMConnectionHandler">
    <APM name="FDNorth" minimumCount="1" maximumCount="1"
    intervalMillis="4"
    class="com.triversity.transnet.xtension.credit.firstdata.north.
    FDNorthCreditThread"
    connectionClass="com.triversity.transnet.core.apm.SocketAPMConn
    ection" transactionLifeSpan="10000">
        <AllowdIdleTime value="1" unit="hour" />
        <Connection id="1" name="127.0.0.1:20002" />
    </APM>
</Connection>
```

FDSouth

The FDSouth APM provides a dial-up interface for gift cards with no settlement. It supports activations, redemptions, voids and balance inquiries.

Configuration

To configure the FDSouth APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

FDSouth specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the FDSouth APM.

```
- <Connection name="FDSouth"
class="com.triversity.transnet.core.apm.APMConnectionHandler">
- <APM name="FDSouth" maximumCount="1" minimumCount="1"
intervalMillis="250"
class="com.triversity.transnet.core.credit.CreditService"
connectionClass="com.triversity.transnet.xtension.credit.modem.ModemConne
ction"
CreditServiceHandlerClassName="com.triversity.transnet.xtension.credit.FD
South.FDSouthHandler" TransactionLifeSpan="95000"
ConnectionReadTimeOut="250">
  <AllowdIdleTime value="1" unit="hour" MerchantsFile="FDSouthMerchant" />
- <Connection id="2" Name="FDSouth">
- <Device
class="com.triversity.transnet.xtension.credit.modem.ModemDevice">
  <PhoneNumber>18008841111</PhoneNumber>
  <ModemConfig>config\modem_dialer.xml</ModemConfig>
  </Device>
  <Protocol MaxTrxTime="95"
class="com.triversity.transnet.xtension.credit.modem.ModemProtocol" />
  </Connection>
  </APM>
```

```
</Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

First Data Value Link APM

First Data Value Link APM provides a mechanism to load First Data ValueLink settlement files in a format that allows data comparison between Store and Credit provider.

Configuration

The First Data Value Link APM is based on the Generic Credit APM. As such, to configure the First Data Value Link APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (generic settings)
- transnet.xml — Tomcat version
- Merchant.xml

First Data Value Link specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the First Data Value Link APM.

```
<Connection name="ValueLinkLoader"

class="com.triversity.transnet.core.tms.file.DirectoryConsumerMonitor"
       interval="1000"
       target="data\\valueLink"
       working_directory="c:\\temp\\valueLink">



       <adapter
class="com.triversity.transnet.core.reconciliation.ReconciliationFileCons
umer" >
       <Reconciliation  chain="10"
                service="ValueLink"
                tableName="PaymentDetail_Bank"
                                archiveFolder="C:\program
files\Transnet\logs\ valueLink \archive"
                merchantFileName="ValueLink Merchant.xml">
<Parser name=" ValueLinkFileParser"
class="com.triversity.transnet.xtension.credit.firstdata.valueLink.settle
ment.ValueLinkFileParser "/>
</Reconciliation>


</adapter>
</Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

First National Merchant Services APM

The First National Merchant Services (*FNMS*) APM provides the connectivity required to perform credit authorization and settlement services from the FNMS service provider through Centralized EFT.

The services the FNMS APM provides include:

Online transactions

- Credit Card Authorization: Supported cards include VISA, MasterCard, and AMEX. Possible credit card transactions include sales and authorizations, returns, and voids.
- Debit Card Purchase: Service allows all debit cards supported by the FNMS services network. Possible debit card transactions include sales and authorizations, returns, and voids.

Configuration

The FNMS APM is based upon the Generic Credit APM. As such, to configure the FNMS APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (generic settings)
- transnet.xml — Tomcat version
- Merchant.xml

FNMS specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the FNMS APM.

```
- <Connection name="FNMS"
class="com.triversity.transnet.core.eft.EFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.fn
ms.FNMSHandler" eftProviderName="FNMS" transactionLifeSpan="10000">
- <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.fnms.FNMSSocketC
onnection">
  <Destination id="1" name="host IP address:host port" />
  </APMConnection>
  </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191. For samples of each of the APM configuration files, see Chapter A, "APM Configuration File Reference" on page 251.

GPS Canada (CIBC) APM

The Global Payments Canada (CIBC) APM provides centralized authorization and settlement with GPS Canada using the Tender Retail service. Clients may be individual stores or the head offices of complete chains. The GPS Canada APM initiates contact with and receives responses from GPS Canada through Tender Retail services. Transaction services available through this APM include credit purchases, credit purchase corrections, credit refunds, debit purchases, debit purchase corrections, and debit refunds, and settlement services.

Configurable Parameters for the GPS Canada Credit APM

The following are the configurable parameters for the APM:

- Number of minimum and maximum TenderRetailCredit instances at all times
- Timeout duration for the TenderRetailCredit application
- Credit Service IP address of Tender Retail Service for external connection
- Messages to log

Configurable Parameters for the GPS Canada Settlement APM

In addition to the generic settings, the following are the configurable parameters for the APM:

- Storage medium (database or flat files)
- Credit service IP address for external connection
- Client proxy IP addresses
- Flat file encryption

Configuration

The GPS Canada APM is a Tender Retail Services-based APM. As such, to configure the GPS Canada APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- transnet.xml — Tomcat version
- TRTerminals.xml (see "Configuring GPS Canada client terminal IDs" on page 215)

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

GPS Canada specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the GPS Canada APM.

```
<Transnet SystemID="10_1_1">
   <Logging>logging.properties</Logging>
   <Service name="ConnectionManager" description="Transnet Connection
   Manager"
   class="com.triversity.transnet.core.tms.connection.TNConnectionManager
   " forcedShutdownWaitTime="5000">
      <ActivationInterval value="10" unit="seconds" />


      ...


      <Connection name="GPSCA"
      class="com.triversity.transnet.core.apm.APMConnectionHandler">
         <APM name="GPSCA" minimumCount="1" maximumCount="1"
         intervalMillis="4"
         class="com.triversity.transnet.xtension.credit.tenderretail.TRR
         eaderWriter"
         connectionClass="com.triversity.transnet.xtension.credit.tender
         retail.TRSocket" TransactionLifeSpan="10000"
         ReadObjectTimeout="30000" UseDebitAccountType="true"
         SettlementRetries="3" ReconcileRetries="4"
         ReconcileFileName="TRReconcileReport.txt"
         SettleFileName="TRSettleReport.txt">
            <ErrorReporter
            class="com.triversity.transnet.core.util.FileReporter"
            fileame="TNReport.txt" reportTime="18:00" />
```

```
            <AllowdIdleTime value="1" unit="hour" />
            <Connection id="1" name="127.0.0.1:3858" />
        </APM>
    </Connection>
```

Configuring GPS Canada client terminal IDs

Each client terminal (for example, each SAP Transactionware POS terminal) must be assigned a unique terminal identification name for use by the GPS Canada transaction processing services. Centralized EFT identifies a client by its chain, store, and register numbers, but allows a single client to have separate terminal identification names to denote them as credit and debit service terminals. You must set this information through the TRTerminals.xml file.

For example, in the following example (a small tag section sample from a TRTerminals.xml file) the terminal identification names OURSTOREC3 and OURSTORED3 have both been assigned to register number 3, in store number 1 of chain number 10, to denote this as both a credit and debit service terminal, respectively.

```
<Terminal>
    <Name>OURSTOREC3</Name>
    <Chain>10</Chain>
    <Store>1</Store>
    <Register>3</Register>
    <Debit>false</Debit>
</Terminal>
<Terminal>
    <Name>OURSTORED3</Name>
    <Chain>10</Chain>
    <Store>1</Store>
    <Register>3</Register>
    <Debit>true</Debit>
</Terminal>
```

GPS Canada TRTerminals.xml settings

A terminal entry must be created for each of your GPS Canada client terminals (POS terminals) together with each chain number, store, and register you use, and for every combination thereof. This may take some time during the initial setup of your system, but seldom needs to be altered thereafter. The following tags must be configured before you can use the GPS Canada APM:

Tag	Set to	Notes
<Name>	Name of the terminal For example, OurStore1	You may assign multiple terminal names to a single register to denote different services supported. For example, you might attach both the names OurStoreC1 and OurStoreD1 to a single register to identify it as a credit terminal and a debit terminal.

<Chain>	Chain Number For example, 1	
<Store>	Store Number For example, 1	
<Register>	Register Number For example, 1	
<Debit>	Indicates whether you want to perform debit transactions through this service True or False	

GPS USA (NDC) APM

The Global Payments USA (NDC) APM provides centralized authorization and settlement with GPS USA using the Tender Retail service. Clients may be individual stores or the head offices of complete chains. The GPS USA APM initiates contact with and receives responses from GPS/NDC through Tender Retail services. Transaction services available through this APM include credit purchases, credit purchase corrections, credit refunds, debit purchases, debit purchase corrections, and debit refunds, and settlement services.

Configurable Parameters for the GPS USA Credit APM

The following are the configurable parameters for the APM:

- Number of minimum and maximum TenderRetailCredit instances at all times
- Time out duration for the TenderRetailCredit application
- Credit Service IP address of Tender Retail Service for external connection
- Messages to log

Configurable Parameters for the GPS USA Settlement APM

On top of the generic settings, the following are the configurable parameters for the APM:

- Storage medium (database or flat files)
- Credit service IP address for external connection
- Client proxy IP addresses
- Flat file encryption

Configuration

The GPS USA APM is a Tender Retail Services based APM. As such, to configure the GPS USA APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- transnet.xml — Tomcat version
- TRTerminals.xml (see "Configuring GPS USA client terminal IDs" on page 217)

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

GPS USA specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the GPS USA APM.

```
<Transnet SystemID="10_1_1">
    <Logging>logging.properties</Logging>
    <Service name="ConnectionManager" description="Transnet Connection
    Manager"
    class="com.triversity.transnet.core.tms.connection.TNConnectionManager
    " forcedShutdownWaitTime="5000">
        <ActivationInterval value="10" unit="seconds" />


        ...


        <Connection name="GPSUSA"
        class="com.triversity.transnet.core.apm.APMConnectionHandler">
            <APM name="GPSUSA" minimumCount="1" maximumCount="1"
            intervalMillis="4"
            class="com.triversity.transnet.xtension.credit.tenderretail.TRR
            eaderWriter"
            connectionClass="com.triversity.transnet.xtension.credit.tender
            retail.TRSocket" TransactionLifeSpan="10000"
            ReadObjectTimeout="30000" UseDebitAccountType="true"
            SettlementRetries="3" ReconcileRetries="4"
            ReconcileFileName="TRReconcileReport.txt"
            SettleFileName="TRSettleReport.txt">
                <ErrorReporter
                class="com.triversity.transnet.core.util.FileReporter"
                fileame="TNReport.txt" reportTime="18:00" />
                <AllowdIdleTime value="1" unit="hour" />
                <Connection id="1" name="127.0.0.1:3858" />
            </APM>
        </Connection>
```

Configuring GPS USA client terminal IDs

Each client terminal (for example, each SAP Transactionware POS terminal) must be assigned a unique terminal identification name for use by the GPS/NDC transaction processing services. Centralized EFT identifies a client by its chain, store, and register numbers, but allows a single client to have separate terminal identification names to denote them as credit and debit service terminals. You must set this information through the TRTerminals.xml file.

For example, in the following example (a small tag section sample from a TRTerminals.xml file) the terminal identification names OURSTOREC3 and OURSTORED3 have both been assigned to register number 3, in store number 1 of chain number 10, to denote this as both a credit and debit service terminal, respectively.

```
<Terminal>
    <Name>OURSTOREC3</Name>
    <Chain>10</Chain>
    <Store>1</Store>
    <Register>3</Register>
    <Debit>false</Debit>
</Terminal>
<Terminal>
    <Name>OURSTORED3</Name>
```

```
    <Chain>10</Chain>
    <Store>1</Store>
    <Register>3</Register>
    <Debit>true</Debit>
</Terminal>
```

GPS USA TRTerminals.xml settings

A terminal entry must be created for each of your GPS USA client terminals (POS terminals) together with each chain number, store, and register you use, and for every combination thereof. This may take some time during the initial set up of your system, but seldom needs to be altered thereafter. The following tags must be configured before you can employ the GPS USA APM:

Tag	Set to	Notes
<Name>	Name of the terminal For example, OurStore1	You may assign multiple terminal names to a single register to denote different services supported. For example, you might attach both the names OurStoreC1 and OurStoreD1 to a single register to identify it as a credit terminal and a debit terminal.
<Chain>	Chain Number For example, 1	
<Store>	Store Number For example, 1	
<Register>	Register Number For example, 1	
<Debit>	Indicates whether you want to perform debit transactions through this service True or False	

ISO 8583 Credit APM

The ISO 8583 Credit APM (*Application Processing Module*) enables your retail POS system to communicate with credit service providers using socket communications. The APM connects to a published port on the host at the service provider using TCP sockets (stream sockets).

The communication can be performed in either synchronous or asynchronous mode.

- In the synchronous mode, the APM establishes a connection, sends a request, receives a response from the host, and then closes the connection.
- In the asynchronous mode, a connection is permanently established. The APM can also re-establish a broken connection as long as it is active. Requests are sent to the host without waiting for responses. When responses are received from the host, they are processed appropriately.

Each message header has a MessageID field to assist in matching responses to original requests.

Message format

All request and response messages are comprised of a message header followed by the message body, as illustrated here:

Message header
Message body

Message header

Each message header displays the following format:

1 Byte Length size (Binary)
Message Length (in bytes) (ASCII)
MessageID (ASCII, 5 digits)

The first byte of the message header is a one-byte binary field that specifies the number of bytes in the message length field. It is followed by the length of the message in the ASCII field. That is followed by a five digit MessageID in ASCII, which must be right aligned with the zero filled-in on the left. The MessageID field is used for matching responses to corresponding requests and therefore must be echoed back in each response.

For example:

- If the message is an ISO message 123 bytes in length, the size of the message length would be 3 bytes and the following header would be sent through the socket, assuming a MessageID of 12:

  ```
  0x03 0x31 0x32 0x33 0x30 0x30 0x30 0x31 0x032
  ```

Message body

The body of the message sent between the Credit APM and the service provider host conforms to the ISO 8583 (1993) message specification. For more information, contact the ISO Web site at: http://www.iso.org/iso/en/CatalogueListPage.CatalogueList?ICS1=35&ICS2=240&ICS3=15

Private fields and values

Apart from the fields specified by the ISO 8583 message specification, two private fields, numbered 62 and 63, are also used.

Field 62 is used to identify the POS Terminal, and it must be echoed back in the response; it is an 11 byte numeric field. The first three bytes represent the Company ID; the next five bytes are the Store Numbers; and the last three bytes are used for Terminal (or Lane or Register) IDs. Note that the Company ID is required only when a single server is used to support multiple companies (for example, in an ASP environment) or banners.

Field 63 is used to pass certain messages between the system and the POS terminal; it can be ignored by the credit service provider.

For proper identification of a card as a private label card in the requests (where the card number is in the range used by major cards) private values are placed in the Processing Code field of the ISO message. Account fields at the positions 3-4 and 5-6 are set to 90 (ASCII).

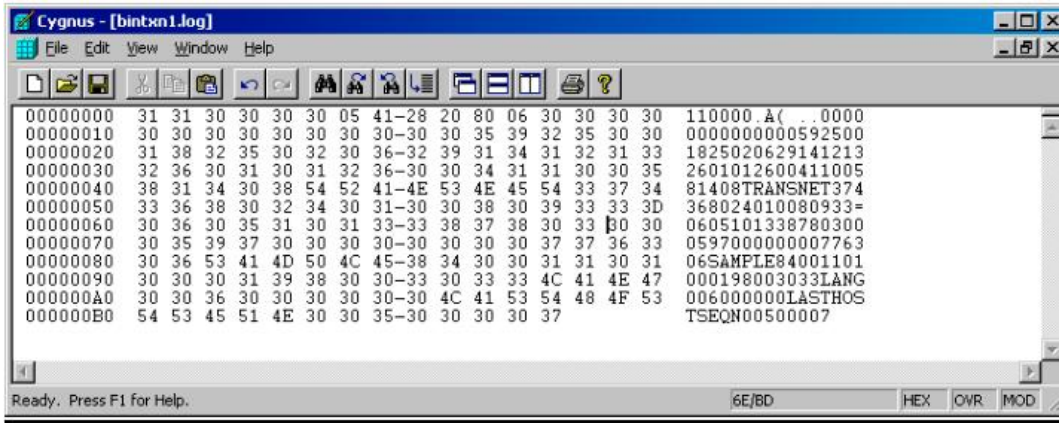
Transaction Type	ISO Message	Processing Code	Function Code
Authorization	1100	009090	100
Void Authorization	1100	229090	100
Return	1100	209090	100
Void Return	1100	029090	100
Payment	1100	509090	100
Void Payment	1100	589090	100
Balance Inquiry	1100	319090	100
PreAuthorization	1100	009090	101
PostAuthorization	1100	009090	102
Void PostAuthorization	1100	229090	102

Sample credit card transaction messages

The following examples show sample ISO8583 Credit APM messages, and include screen capture depictions of the formats, together with the equivalent field values.

Credit card request for authorization

The binary output from an editor is displayed in the following example. The left-most column represents the position of the byte; each row contains 16 bytes. The right-most column is the ASCII representation of the binary data).



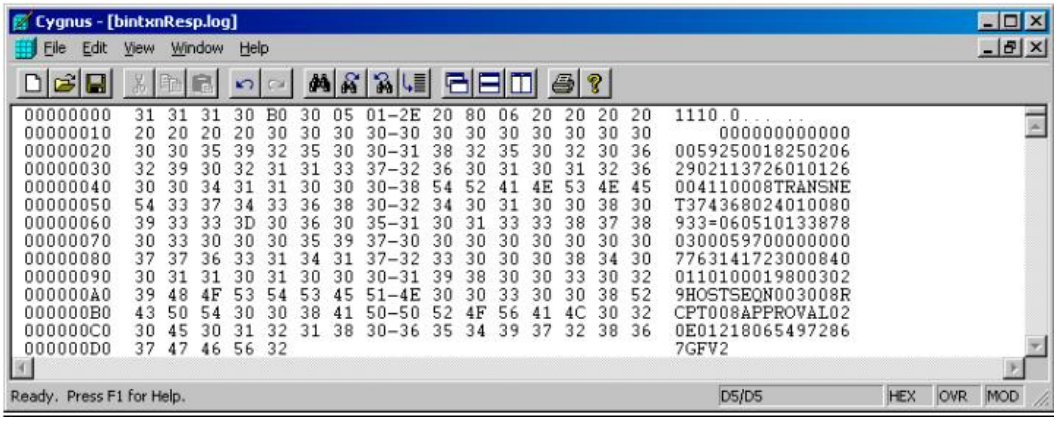
The individual field values for the above message are as follows:

```
(M T I D > 1100 <
BIT MAP >
Field 003 >      0 <
Field 004 >        10.82 <
```

```
Field 011 >   3718 <
Field 012 > 020710100040 <
Field 022 > 260101260041 <
Field 024 > 100 <
Field 026 > 5814 <
Field 032 - Length > 8 <
Field 032 - Value > TRANSNET <
Field 035 - Length > 32 <
Field 035 - Value > 4417122983129249=021210110000242 <
Field 037 >         5553 <
Field 043 - Length > 6 <
Field 043 - Value > SAMPLE <
Field 049 > 840 <
Field 062 - Length > 11 <
Field 062 - Value > 01000106003 <
Field 063 - Length > 33 <
Field 063 - Value > LANG006000000LASTHOSTSEQN00500025
```

Credit service provider response

The following graphic illustrates the response to the credit card request message.



The individual field values for the above message are as follows:

```
M T I D > 1110 <
Field 003 > 000000 <
Field 004 >        10.82 <
Field 011 > 003718 <
Field 012 > 020710100052 <
Field 022 > 260101260041 <
Field 024 > 100 <
Field 032 - Length > 8 <
Field 032 - Value > TRANSNET <
Field 035 - Length > 32 <
Field 035 - Value > 4417122983129249=021210110000242 <
Field 037 > 000000005553 <
Field 038 > 010104 <
Field 039 > 000 <
```

```
Field 049 > 840 <
Field 062 - Length > 11 <
Field 062 - Value > 01000106003 <
Field 063 - Length > 29 <
Field 063 - Value > HOSTSEQN003001RCPT008APPROVAL <
Field 123 - Length > 20 <
Field 123 - Value > E0121915045255 62GLZN <
```

Configuration

To configure the ISO8583 APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

ISO8583 specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the ISO8583 APM.

```
<Transnet SystemID="10_1_1">
   <Logging>logging.properties</Logging>
   <Service name="ConnectionManager" description="Transnet Connection
   Manager"
   class="com.triversity.transnet.core.tms.connection.TNConnectionManager
   " forcedShutdownWaitTime="5000">
      <ActivationInterval value="10" unit="seconds" />

      ...

      <Connection name="ISO8583PassThrough"
      class="com.triversity.transnet.core.tms.ip.TNBytesSocketClient"
      minimumSequenceNumber="0" host="127.0.0.1" port="20003"
      maximumSequenceNumber="999999">
         <MessageConverter
         class="com.triversity.transnet.core.eft.converter.XMLMessageByt
         esConverter" />
      </Connection>
```

MPS

The MPS APM provides a TCP/IP interface for credit and debit cards with a store based settlement through Centralized EFT.

Configuration

To configure the MPS APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

MPS specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the MPS APM.

```
- <Connection name="MPS"
class="com.triversity.transnet.core.apm.APMConnectionHandler">
- <APM name="MPS" maximumCount="1" minimumCount="1"
class="com.triversity.transnet.core.credit.CreditService"
connectionClass="com.triversity.transnet.xtension.credit.mps.MPSSocketCon
nection"
CreditServiceHandlerClassName="com.triversity.transnet.xtension.credit.mp
s.MPSHandler" TransactionLifeSpan="10000">
  <AllowdIdleTime value="1" unit="hour" MerchantsFile="MPSMerchant" />
  <Connection id="2" name="host ip address:host port" />
  </APM>
  </Connection>
```

For more information about the generic APM configuration files, see “Generic configuration file settings” on page 191.

MPSprecertifiedbyfifththird

The MPSprecertifiedbyfifththird APM supports sending a reconciliation request to the service provider.. The request contains the transaction breakdown along with totals. The service provider may then proceed with the settlement. Usually the response the reconciliation reuest is either "In Balance" or "Out of Balance". For an “Out of Balance” response, the service provider also provides any mismatch totals.

Configuration

To configure the MPSprecertifiedbyfifththird APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

MPSpresecritfiedbyfifththird specific transnet.xml settings

```
<Connection name="MPS"
   class="com.triversity.transnet.core.eft.EFTService"

creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.mp
sprecertifiedbyfifththird.MPSHandler"
   eftProviderName="MPS"
   transactionLifeSpan="38000" >
   <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.mpsprecertifiedb
yfifththird.MPSSocketConnection" useEBCDICHeader="true">
   <Destination id="1" name="192.152.100.99:37401"/>
   </APMConnection>
</Connection>
```

useEBCDICHeader flags the APM that a four byte EBCDIC length indicator is required for all messages going on to 5th3rd. This includes the sign on message.

MPSpresecritfiedbyfifththird specific merchant.xml settings

```
<Merchants>
  <Chain>
   <ChainId>20</ChainId>
   <Name>Zealler</Name>
   <Contact>John Smith, #123 SpringField Tel 416-234-5678</Contact>
        <BusinessCutoverTime>000000</BusinessCutoverTime>
   <Store>
      <StoreId>1</StoreId>
      <ExtStoreId>2002</ExtStoreId>
      <ServiceType>CPS2000</ServiceType>
      <Name>000000925990</Name>
      <City>Toronto</City>
      <State>ON.</State>
      <ZIPCode>M2H 2N5</ZIPCode>
      <Account>1042000314159005017</Account> Bank number appended by
Merchant Number
      <SecurityCode>2001</SecurityCode>
      <ClearingCode>6666</ClearingCode>
      <ExtAccount>666641238789</ExtAccount>
      <BINNumber>9B </BINNumber> Required for populating a future field
      in 5th3rd message format 9B and 4 spaces
      <ICANumber>AuthOnly</ICANumber> Required , indicates draft capture
      <CategoryCode>5412</CategoryCode>
      <CheckProcessor>TELECHECK</CheckProcessor>
      <Contact>Triversity Inc. #3550 Victoria Park Tel 416-791-7100
      </Contact>
      <BusinessCutoverTime>235959</BusinessCutoverTime>
      <Register>
         <RegisterId>4</RegisterId>
         <CreditName>00157200204 </CreditName> Terminal ID assignedby
         5th3rd
         <DebitName>00157200204 </DebitName>
```

```
        </Register>
        <Register>
            <RegisterId>3</RegisterId>
            <CreditName>00157200203 </CreditName>
            <DebitName>00157200203 </DebitName>
        </Register>
        <Register>
            <RegisterId>2</RegisterId>
            <CreditName>00157200202    </CreditName>
            <DebitName>00157200202    </DebitName>
        </Register>
        <Register>
            <RegisterId>1</RegisterId>
            <CreditName>00157200201    </CreditName>
            <DebitName>00157200201    </DebitName>
        </Register>
    </Store>
  </Chain>
</Merchants>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

NextelEclipse APM

NexelEclipse APM provides a mechanism to load payment Nextel and EAI settlement files in a format that allows data comparison between Store and Credit provider.

Configuration

The NextelEclipse APM is based upon the Generic Credit APM. As such, to configure the NextelEclipse APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (generic settings)
- transnet.xml — Tomcat version
- Merchant.xml

NextelEclipse specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the NextelEclipse APM.

```
<Transnet SystemID="1_1_1">
   <Logging>logging.properties </Logging>
   <Service name="ConnectionManager" description="Transnet Connection
Manager"
class="com.triversity.transnet.core.tms.connection.TNConnectionManager"
forcedShutdownWaitTime="5000">
      <ActivationInterval value="10" unit="seconds"/>


      <Connection name="ClientSocketConnection"
class="com.triversity.transnet.core.tms.ip.TNServerSocket"
handlerClass="com.triversity.transnet.core.tms.ip.TNXMLSocketServer"
host="localhost" port="10000" acceptTimeout="1000" Version="2"/>
```

```
      <Connection name="TPSServer"
class="com.triversity.transnet.core.tms.tps.TPSServerHandler"
TPSPipeName="Transnet" TPSDebugLogging="false"
TransactionBufferSize="80000"/>



      <Connection name="NextelBTA"
class="com.triversity.transnet.core.eft.EFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.Ne
xtelEclipse.NextelEclipseHandler" merchantsFile="NextelBTAMerchant.xml"
eftProviderName="NextelBTA" transactionLifeSpan="40000">
         <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.NextelEclipse.Ne
xtelEclipseMQConnection"
                        delegateWriteToConnection="true"
          SendToQueue="true"
          SendTo="Nextel.InputQueue"
          ReceiveFrom="Nextel.OutputQueue"
             QueueManagerName="QM_teh2k2">
            <Destination id="1" name="NextelBTA"/>
         </APMConnection>
      </Connection>


      <Connection name="NextelEFT"
class="com.triversity.transnet.core.eft.SynchronousEFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.Ne
xtelEclipse.NextelEclipseHandler" merchantsFile="NextelEFTMerchant.xml"
eftProviderName="NextelEFT" transactionLifeSpan="40000">
         <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.NextelEclipse.Ne
xtelEclipseMQConnection"
                  delegateWriteToConnection="true"
          SendToQueue="true"
          SendTo="PPRD.S.EADS.PAYMNTSVCS.PAYMENTREQST"
          SendToQueueIsModel="false"
          ReceiveFrom="PPRD.R.NRS.PAYMNTSVCS.PAYMENTRESP"
          ReceiveFromQueueIsModel="true"
             QueueManagerName="QM_teh2k2">
            <Destination id="1" name="NextelEFT"/>
         </APMConnection>
      </Connection>



   </Service>
   <Service name="MessageManager" description="Transnet Message Manager"
class="com.triversity.transnet.core.tms.message.TNMessageManager"
forcedShutdownWaitTime="5000">
      <ActivationInterval value="10" unit="seconds"/>



      <!-- All other messages route to BTA /-->
      <MessageDescriptor name="BTAMessage" root="FinancialMessage">
         <Check xmlType="text" name="MessageType.AccountType"
dataType="text">
```

```
            <DataText value="PrivateLabelCard"/>
         </Check>
      </MessageDescriptor>
      <MessageDescriptor name="DefaultFinance" root="FinancialMessage"/>
      <MessageDescriptor name="ISOBTAMessage" root="BytesMessage">
         <Check xmlType="text" name="CardType" dataType="text">
            <DataText value="PrivateLabelCard"/>
         </Check>
      </MessageDescriptor>
      <MessageDescriptor name="ISOEFTMessage" root="BytesMessage"/>


      <!-- ROUTING RULES /-->

      <RoutingRule message="BTAMessage">
         <Route name="BTAXMLRoute" timeoutSeconds="15"/>
      </RoutingRule>

       <RoutingRule message="DefaultFinance">
         <Route name="EFTXMLRoute" timeoutSeconds="15"/>
      </RoutingRule>

      <RoutingRule message="ISOBTAMessage">
         <Route name="BTARoute" timeoutSeconds="15"/>
      </RoutingRule>

      <RoutingRule message="ISOEFTMessage">
         <Route name="EFTRoute" timeoutSeconds="15"/>
      </RoutingRule>

      <!-- ROUTES /-->


      <Route name="BTARoute" connectionName="NextelBTA" conversion=""
class="com.triversity.transnet.core.tms.route.TNBytesMessageRoute"/>

      <Route name="BTAXMLRoute" connectionName="NextelBTA"/>

      <Route name="EFTRoute" connectionName="NextelEFT" conversion=""
class="com.triversity.transnet.core.tms.route.TNBytesMessageRoute"/>

      <Route name="EFTXMLRoute" connectionName="NextelEFT"/>


   </Service>
   <Database type="rdbms" Driver="com.inet.tds.TdsDriver"
Url="jdbc:inetdae7://10.8.20.145:1434/transnet" User="xxxxxx"
Password="xxxxxx">
```

```
      <ConnectionPool Debug="false" Name="TransnetJDBCPool"
CleaningIntervalSeconds="60" MinimumConnections="1"
MaximumConnections="10" ConnectionLifeSeconds="1800"
MaximumUseCount="25"/>
   </Database>
   <ThreadPool Name="TransnsetThreadPool" Debug="false"
CleaningIntervalSeconds="60" MinimumThreads="5"/>
</Transnet>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

Nova APM

The Nova APM provides the connectivity required to perform credit authorization and settlement services from the Nova service provider through Centralized EFT.

The services the Nova APM provide include:

Online transactions

- Credit Card Authorization: Supported cards include VISA, MasterCard, AMEX, Diners Club, Discover, JCB, and Carte Blanche. Possible credit card transactions include sales and authorizations, returns, and voids.

Settlement

- Provides credit card authorization data settlement with Nova. The system generates a batch settlement file in accordance with the daily online transactions, and transmits it to the Nova settlement center for completion.

Configuration

The Nova APM is based upon the Generic Credit APM. As such, to configure the Nova APM services for use, you must configure the following generic application processing module files:

- transnet.xml — Centralized EFT server version (generic settings)
- transnet.xml — Tomcat version
- Merchant.xml

NOVA specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the NOVA APM.

```
<Connection name="NovaAPMConnection"
class="com.triversity.transnet.core.eft.EFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.no
va.NovaHandler" eftProviderName="Nova" transactionLifeSpan="10000">
         <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.nova.NovaSocketC
onnection">
            <Destination id="1" name="127.0.0.1:20001"/>
         </APMConnection>
</Connection>Opt
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

RBC

The RBC (Royal Bank of Canada) APM provides a TCP/IP interface for credit, debit and gift cards, but no settlement

Configuration

To configure the RBC APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

RBC specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the RBC APM.

```
- <Connection name="RbcAPM"
class="com.triversity.transnet.core.eft.EFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.rb
c.RbcHandler" eftProviderName="RBC" transactionLifeSpan="10000">
- <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.rbc.RbcSocketCon
nection">
  <Destination id="1" name="127.0.0.1:20002" />
  </APMConnection>
  </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

Stored Value Application APM

The Stored Value Application (*SVA*) APM provides the connectivity required to perform centralized stored value card (giftcard) activation, tracking, and authorization using the SVA system via Centralized EFT.

Configuration

To configure the SVA APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

SVA specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the SVA APM.

```
<Transnet SystemID="10_1_1">
   <Logging>logging.properties</Logging>
   <Service name="ConnectionManager" description="Transnet Connection
   Manager"
   class="com.triversity.transnet.core.tms.connection.TNConnectionManager
   " forcedShutdownWaitTime="5000">
      <ActivationInterval value="10" unit="seconds" />
```

```
...

<Connection name="TSVAConnection"
class="com.triversity.transnet.core.tms.jms.TNExternalJMSConnectio
n" sendTo="topic/StoredValueRequest" receiveFrom="topic/
StoredValueResponse" durableName="">
    <JMSContext name="JBOSSMQ"
    contextBroker="org.jnp.interfaces.NamingContextFactory"
    providerURL="jnp://127.0.0.1:1099"
    urlPackage="org.jboss.naming" topicBroker="ConnectionFactory"
    queueBroker="ConnectionFactory" username="" password="">
        <jndiProperty name="java.naming.rmi.security.manager"
        value="no" />
    </JMSContext>
</Connection>
```

SVS (NewSVS)

The SVS APM provides TCP/IP interface for gift cards only. It supports activations, redemptions, voids and reversals.

Configuration

To configure the SVS APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

SVS specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the SVS APM.

```
- <Connection name="SVS"
class="com.triversity.transnet.core.eft.SynchronousEFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.ne
wsvs.NewSvsHandler" eftProviderName="SVS">
- <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.newsvs.NewSvsSoc
ketConnection" echoTestIntervalSeconds="300000"
delegateWriteToConnection="true">
  <Destination id="1" name="Connection ip:port" />
  </APMConnection>
  </Connection>
```

For more information about the generic APM configuration files, see “Generic configuration file settings” on page 191.

TD

The Toronto Dominion (TD) APM provides credit and debit across TCP/IP with no settlement.

Configuration

To configure the TD APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (specific settings, outlined below)
- Merchant.xml

TD specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the TD APM.

```
- <Connection name="TDAPM"
class="com.triversity.transnet.core.eft.MultiConnectionEFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.TD
.TDHandler" eftProviderName="TD">
- <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.td.TDSocketConne
ction">
  <Destination id="1" name="142.205.96.67:5011" />
  <Destination id="2" name="142.205.96.67:5012" />
  </APMConnection>
  </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

Telecheck and Paymentech (Nextel) APM

Telecheck and Paymentech (Nextel) APM provides a mechanism to load paymenTech and Telecheck settlement files in a format that allows data comparison between Store and Credit provider.

Configuration

The First Data Value Link APM is based upon the Generic Credit APM. As such, to configure the First Data Value Link APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (generic settings)
- transnet.xml — Tomcat version
- Merchant.xml

Telecheck and Paymentech (Nextel)specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the Telecheck and Paymentech (Nextel) APM.

```
<Connection name="TeleCheckIPPLoader"

class="com.triversity.transnet.core.tms.file.DirectoryConsumerMonitor"
       interval="1000"
       target="data\\telecheckIPP"
       working_directory="c:\\temp\\telecheckIPP">



       <adapter
class="com.triversity.transnet.core.reconciliation.ReconciliationFileCons
umer" >
       <Reconciliation  chain="10"
                service="TeleCheck"
                tableName="PaymentDetail_Bank"
                                archiveFolder="C:\program
files\Transnet\logs\telecheckIPP\archive"
                merchantFileName="Merchant.xml">
```

```
                                <Parser name="TeleCheckFileParser"
class="com.triversity.transnet.xtension.credit.firstdata.telecheck.settle
ment.TeleCheckFileParser"/>
             </Reconciliation>
</Connection>

<Connection name="TeleCheckCorporateLoader"

class="com.triversity.transnet.core.tms.file.DirectoryConsumerMonitor"
       interval="1000"
       target="data\\telecheckCorporate"
       working_directory="c:\\temp\\telecheckCorporate">



       <adapter
class="com.triversity.transnet.core.reconciliation.ReconciliationFileCons
umer" >
       <Reconciliation  chain="11"
                service="TeleCheck"
                tableName="PaymentDetail_Bank"
                                archiveFolder="C:\program
files\Transnet\logs\telecheckCorporate\archive"
                merchantFileName="Merchant.xml">
<Parser name="TeleCheckFileParser"
class="com.triversity.transnet.xtension.credit.firstdata.telecheck.settle
ment.TeleCheckFileParser"/>
</Reconciliation>



</adapter>
</Connection>
<Connection name="PaymentechIPPLoader"

class="com.triversity.transnet.core.tms.file.DirectoryConsumerMonitor"
       interval="1000"
       target="data\\paymentech"
       working_directory="c:\\temp\\paymentechIPP">



       <adapter
class="com.triversity.transnet.core.reconciliation.ReconciliationFileCons
umer" >
       <Reconciliation  chain="10"
                service="Paymentech"
                tableName="PaymentDetail_Bank"
                                archiveFolder="C:\program
files\Transnet\logs\paymentechIPP\archive"
                merchantFileName="Merchant.xml">
<Parser name="PaymentechFileParser"
class="com.triversity.transnet.xtension.credit.paymentech.settlement.Paym
entechFileParser"/>
</Reconciliation>


</adapter>
```

```
</Connection>


<Connection name="PaymentechCorporateLoader"

class="com.triversity.transnet.core.tms.file.DirectoryConsumerMonitor"
       interval="1000"
       target="data\\paymentechCorporate"
       working_directory="c:\\temp\\paymentechCorporate">



       <adapter
class="com.triversity.transnet.core.reconciliation.ReconciliationFileCons
umer" >
       <Reconciliation  chain="11"
                service="Paymentech"
                tableName="PaymentDetail_Bank"
                                archiveFolder="C:\program
files\Transnet\logs\paymentecChorporate\archive"
              merchantFileName="Merchant.xml">
<Parser name="PaymentechFileParser"
class="com.triversity.transnet.xtension.credit.paymentech.settlement.Paym
entechFileParser"/>
</Reconciliation>
</adapter>
</Connection>

Store Configuration
==================



<Connection name="IPPStoreTLOGLoader"

class="com.triversity.transnet.core.tms.file.DirectoryConsumerMonitor"
         interval="1000" target="data\\IPPStores"
         working_directory="c:/temp//IPPStores">
       <adapter
class="com.triversity.transnet.core.reconciliation.ReconciliationFileCons
umer">
            <Reconciliation chain="10"
                 service="Store"
                 tableName="PaymentDetail_Store"
                 archiveFolder="C:\program
files\Transnet\logs\IPPStores\archive"
                 merchantFileName="Merchants.xml">
               <Parser name="TLogFileParser"
class="com.triversity.transnet.core.tlog.TLogFileParser"/>
            </Reconciliation>
         </adapter>
      </Connection>
```

```
<Connection name="CorporateStoreTLOGLoader"

class="com.triversity.transnet.core.tms.file.DirectoryConsumerMonitor"
            interval="1000"
            target="data\\CorporateStores"
            working_directory="c:/temp/CorporateStores">
         <adapter
class="com.triversity.transnet.core.reconciliation.ReconciliationFileCons
umer">
            <Reconciliation chain="11"
                  service="Store"
                  tableName="PaymentDetail_Store"
                  archiveFolder="C:\program
files\Transnet\logs\CorporateStores\archive"
                  merchantFileName="Merchants.xml">
               <Parser name="TLogFileParser"
class="com.triversity.transnet.core.tlog.TLogFileParser"/>
            </Reconciliation>
         </adapter>
</Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

VirginMobile

The VirginMobile APM provides the connectivity required to perform gift card authorization/activation from the VirginMobile service provider through Centralized EFT.

Configuration

The VirginMobile APM is based upon the Generic Credit APM. As such, to configure the VirginMobile APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (generic settings)
- Merchant.xml

VirginMobilespecific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the VirginMobile APM.

```
- <Transnet SystemID="10_1_1">
  <Logging>logging.properties</Logging>
- <Service name="ConnectionManager" description="Transnet Connection
Manager"
class="com.triversity.transnet.core.tms.connection.TNConnectionManager"
forcedShutdownWaitTime="5000">
  <ActivationInterval value="10" unit="seconds" />
  <Connection name="XMLSocketClient"
class="com.triversity.transnet.core.tms.ip.TNServerSocket"
handlerClass="com.triversity.transnet.core.tms.ip.TNXMLSocketServer"
host="localhost" port="10000" acceptTimeout="1000" version="1"
xmlEncoding="UTF-8" />
- <Connection name="ConcordEFSAPM"
class="com.triversity.transnet.core.apm.APMConnectionHandler">
- <APM name="ConcordEFS" maximumCount="1" minimumCount="1"
class="com.triversity.transnet.core.credit.CreditService"
connectionClass="com.triversity.transnet.xtension.credit.concordEFS.Conco
rdEFSSocketConnection"
CreditServiceHandlerClassName="com.triversity.transnet.xtension.credit.co
ncordEFS.ConcordEFSHandler" TransactionLifeSpan="15000"
ConnectionReadTimeOut="2000">
  <AllowdIdleTime value="1" unit="hour"
MerchantsFile="ConcordEFSMerchant" />
  <Connection id="2" name="204.194.125.43:7735" />
  </APM>
  </Connection>
- <Connection name="VirginMobileAPM"
class="com.triversity.transnet.core.eft.EFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.vi
rginMobile.VirginMobileHandler" eftProviderName="VirginMobile"
transactionLifeSpan="30000">
- <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.virginMobile.Vir
ginMobileSocketConnection">
  <Destination id="1" name="205.239.223.73:15217" />
  </APMConnection>
  </Connection>
  </Service>
- <!--
  Please note that order is important in configuring MessageDescriptors,
              a message is compared against a MessageDescriptor in order


  -->
- <Service name="MessageManager" description="Transnet Message Manager"
class="com.triversity.transnet.core.tms.message.TNMessageManager"
forcedShutdownWaitTime="5000">
  <ActivationInterval value="10" unit="seconds" />
- <MessageDescriptor name="VirginMobileMessage" root="FinancialMessage">
- <Check xmlType="text" name="Request.CardDetail.UPC" dataType="text">
  <DataText value="836182005645" />
  <DataText value="836182005638" />
  <DataText value="836182005621" />
  </Check>
```

```
  </MessageDescriptor>
  <MessageDescriptor name="DefaultFinance" root="FinancialMessage" />
  <Route name="VirginMobileRoute" connectionName="VirginMobileAPM" />
- <RoutingRule message="VirginMobileMessage">
  <Route name="VirginMobileRoute" timeoutSeconds="15" />
  </RoutingRule>
  <Route name="ConcordRoute" connectionName="ConcordEFSAPM" />
- <RoutingRule message="DefaultFinance">
  <Route name="ConcordRoute" timeoutSeconds="15" />
  </RoutingRule>
  </Service>
  <ThreadPool Name="TransnsetThreadPool" Debug="false"
CleaningIntervalSeconds="60" MinimumThreads="5" MaximumThreads="300" />
  </Transnet>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

Vital APM

The Vital APM provides the connectivity required to perform credit services from the Vital service provider through Centralized EFT.It also support store-based settlement.

Configuration

The Vital APM is based upon the Generic Credit APM. As such, to configure the Vital APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (generic settings)
- Merchant.xml

Vital specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the Vital APM.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Transnet SystemID="10_1_1">
  <Logging>logging.properties</Logging>
- <Service name="ConnectionManager" description="Transnet Connection
Manager"
class="com.triversity.transnet.core.tms.connection.TNConnectionManager"
forcedShutdownWaitTime="5000">
  <ActivationInterval value="10" unit="seconds" />
  <Connection name="XMLSocketClient"
class="com.triversity.transnet.core.tms.ip.TNServerSocket"
handlerClass="com.triversity.transnet.core.tms.ip.TNXMLSocketServer"
host="localhost" port="10000" acceptTimeout="1000" version="1"
xmlEncoding="UTF-8" />
- <Connection name="GCFQuickCreditAPMConnection"
class="com.triversity.transnet.core.tms.connection.TNMessageConsumerConne
ction">
- <adapter
class="com.triversity.transnet.xtension.credit.GCF.GCFMessageProcessor"
connectionClass="com.triversity.transnet.xtension.credit.GCF.GCFSocketCon
nection">
  <Connection id="1" name="127.0.0.1:20001" />
```

```
  </adapter>
  </Connection>
- <Connection name="VitalAuthorizationSocketConnection"
class="com.triversity.transnet.core.eft.SynchronousEFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.vi
tal.VitalHandler" eftProviderName="Vital" transactionLifeSpan="10000"
reuseConnection="false">
- <APMConnection
connectionClass="com.triversity.transnet.xtension.credit.vital.VitalAutho
rizationSocketConnection">
  <Destination id="2" name="206.175.128.101:10198" />
  </APMConnection>
  </Connection>
- <Connection name="VitalSocketSettlement"
class="com.triversity.transnet.core.tms.connection.TNMessageConsumerConne
ction">
- <adapter
class="com.triversity.transnet.adapter.settlement.GMBatchSettlementMessag
eProcessor">
  <handler
class="com.triversity.transnet.adapter.settlement.FileBasedBatchSettlemen
tHandler" outputDirectory="data\settlement" />
- <processor
class="com.triversity.transnet.xtension.credit.vital.VitalSettlementSocke
tProcessor" cardType="All"
connectionClass="com.triversity.transnet.xtension.credit.vital.VitalSettl
ementSocketConnection">
  <Connection id="4" name="206.175.128.101:10198" />
- <configParms>
  <merchantFileName>VitalMerchant.xml</merchantFileName>
  <outputDir>data\settlement</outputDir>
  <agentBankNumber>000000</agentBankNumber>
  <agentChainNumber>111111</agentChainNumber>
  <timeZoneDifferential>605</timeZoneDifferential>
- <!--
 Optional settings, if not set, then these are the defaults
                   <recordFormat>K</recordFormat>
                   <applicationType>3</applicationType>
                   <currencyCode>840</currencyCode>
                   <languageIndicator>00</languageIndicator>
                   <terminalIDNumber>00000001</terminalIDNumber>
                   <merchantLocationNumber>00001</
merchantLocationNumber>
                   <blockingIndicator>2</blockingIndicator>



  -->
  </configParms>
  </processor>
  </adapter>
  </Connection>
  </Service>
- <Service name="MessageManager" description="Transnet Message Manager"
class="com.triversity.transnet.core.tms.message.TNMessageManager"
forcedShutdownWaitTime="5000">
```

```
  <ActivationInterval value="10" unit="seconds" />
- <MessageDescriptor name="VitalSettlementRequest"
root="FinancialMessage">
- <Check xmlType="text" name="Originator.Chain" dataType="number">
  <Range minimum="1" maximum="999" />
  </Check>
- <Check xmlType="text" name="Originator.Store" dataType="number">
  <Range minimum="1" maximum="9999" />
  </Check>
- <Check xmlType="text" name="Originator.Register" dataType="number">
  <Range minimum="1" maximum="999" />
  </Check>
- <Check xmlType="text" name="MessageType.Function" dataType="text">
  <DataText value="BatchDetail" />
  <DataText value="SendBatch" />
  </Check>
  </MessageDescriptor>
- <MessageDescriptor name="QuickCreditRequest" root="FinancialMessage">
- <Check xmlType="text" name="Originator.Chain" dataType="number">
  <Range minimum="1" maximum="999" />
  </Check>
- <Check xmlType="text" name="Originator.Store" dataType="number">
  <Range minimum="1" maximum="9999" />
  </Check>
- <Check xmlType="text" name="Originator.Register" dataType="number">
  <Range minimum="1" maximum="999" />
  </Check>
- <Check xmlType="text" name="MessageType.Function" dataType="text">
  <DataText value="Apply" />
  </Check>
  </MessageDescriptor>
- <MessageDescriptor name="DefaultFinance" root="FinancialMessage">
- <Check xmlType="text" name="Originator.Chain" dataType="number">
  <Range minimum="1" maximum="999" />
  </Check>
- <Check xmlType="text" name="Originator.Store" dataType="number">
  <Range minimum="1" maximum="9999" />
  </Check>
- <Check xmlType="text" name="Originator.Register" dataType="number">
  <Range minimum="1" maximum="999" />
  </Check>
- <!--
          <Check xmlType="text" name="AccountNumber" dataType="number"
substring="0,4">
             <Range minimum="0" maximum="9999"/>
          </Check>
          <Check xmlType="text" name="CardType" dataType="text">
             <DataText value="VISA"/>
             <DataText value="MASTERCARD"/>
          </Check>
```

```
            <Check xmlType="text" name="Function" dataType="text">
               <DataText value="ALL"/>
            </Check>


  -->
  </MessageDescriptor>
  <Route name="VitalRoute"
connectionName="VitalAuthorizationSocketConnection" />
  <Route name="VitalSettlementRoute"
connectionName="VitalSocketSettlement" />
  <Route name="GCFQuickCreditRoute"
connectionName="GCFQuickCreditAPMConnection" />
- <RoutingRule message="DefaultFinance">
  <Route name="VitalRoute" timeoutSeconds="95" />
  </RoutingRule>
- <RoutingRule message="VitalSettlementRequest">
  <Route name="VitalSettlementRoute" timeoutSeconds="300" />
  </RoutingRule>
- <RoutingRule message="QuickCreditRequest">
  <Route name="GCFQuickCreditRoute" timeoutSeconds="30" />
  </RoutingRule>
  </Service>
- <Database type="jdbms" Driver="com.inet.tds.TdsDriver"
Url="jdbc:inetdae7://:1433/" User="" Password="">
  <ConnectionPool Debug="false" Name="TransnetJDBCPool"
CleaningIntervalSeconds="60" MinimumConnections="1"
MaximumConnections="10" ConnectionLifeSeconds="1800" MaximumUseCount="25"
/>
  </Database>
  <ThreadPool Name="TransnsetThreadPool" Debug="false"
CleaningIntervalSeconds="60" MinimumThreads="5" />
  </Transnet>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

WildCard APM

The WildCard APM provides the connectivity required to perform gift card authorization/activation and voiding services from the WildCard service provider through Centralized EFT.

Configuration

The WildCard APM is based upon the Generic Credit APM. As such, to configure the WildCard APM services for use, you must configure the following files:

- transnet.xml — Centralized EFT server version (generic settings)
- transnet.xml — Tomcat version
- Merchant.xml

WildCard specific transnet.xml settings

This section shows a sample of the transnet.xml tag settings specifically required for the WildCard APM.

```
<Connection name="WildCard"
class="com.triversity.transnet.core.eft.SynchronousEFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.credit.wi
ldcard.WildCardHandler" eftProviderName="Wildcard"
transactionLifeSpan="10000">
         <APMConnection
connectionClass="com.triversity.transnet.core.apm.URLAPMConnection"
synchronous="true">
          <!-- Following WildCard configuration connects to WildCard test
host  -->
            <Destination id="1" name="https://partners.claimcard.com/A2A/
"/>
         </APMConnection>
      </Connection>
```

For more information about the generic APM configuration files, see "Generic configuration file settings" on page 191.

Customer Order Management APM (XIHttpAdapter)

The Customer Order Management (COM) APM's XIHttp Adapter provides for communications between Transactionware GM and SAP ERP through Transnet and XI. The requests and respnses are in XML form.

The three request types are differentiated by the TypeCode parameter as shown int the XML for each type.

TRansnet can support other requests by using different TypeCodes, so future expansion probably not need any code changes to Transnet.

The following request messages that will come from Transnactionware GM:

- Sales Order
- Billing Document
- Credit Memo

Configuration

Sales Order XML

```
<?xml version="1.0"?>
<ns1:POSLog xmlns:ns1="http://sap.com/xi/StoreConnectivity">
       <Transaction>
      <RetailStoreID>10</RetailStoreID>
      <WorkstationID>1</WorkstationID>
      <SequenceNumber>1</SequenceNumber>
      <RetailTransaction Version="2.0" TypeCode=”01”>
         <SpecialOrderNumber>0000006493</SpecialOrderNumber>
      </RetailTransaction>
       </Transaction>
</ns1:POSLog>
```

Billing Document XML

```
<?xml version="1.0"?>
<ns1:POSLog xmlns:ns1="http://sap.com/xi/StoreConnectivity">
       <Transaction>
       <RetailStoreID>10</RetailStoreID>
       <WorkstationID>1</WorkstationID>
       <SequenceNumber>1</SequenceNumber>
       <RetailTransaction Version="2.0" TypeCode=”11”>
          <SpecialOrderNumber>0000006493</SpecialOrderNumber>
       </RetailTransaction>
       </Transaction>
</ns1:POSLog>
```

Credit Memo XML

```
<?xml version="1.0"?>
<ns1:POSLog xmlns:ns1="http://sap.com/xi/StoreConnectivity">
       <Transaction>
       <RetailStoreID>10</RetailStoreID>
       <WorkstationID>1</WorkstationID>
       <SequenceNumber>1</SequenceNumber>
       <RetailTransaction Version="2.0" TypeCode=”12”>
          <SpecialOrderNumber>0000006493</SpecialOrderNumber>
       </RetailTransaction>
       </Transaction>
</ns1:POSLog>
```

For more information about the generic APM configuration files, see “Generic configuration file settings” on page 191.

About client backward compatibility

Note: Previous versions of Centralized EFT/File Transfer were called Transnet.

Transnet 1.5 Server accepts and responds to requests from either Transnet 1.3 or Transnet 1.4 site clients, or directly from the XPS Server through the NTM Client Server. It differentiates between TN1.3 and TN1.4 messages through the userHeaderLen field of the message header. In TN1.3 messages, this field is always 22 characters long. In TN1.4 messages, this field is 41 characters long or greater. It differentiates between ISO8583 messages and XML messages by the first byte of the message. If the first byte is a carat symbol, or '<', it is read as an XML message. If the first byte is not '<', or a carat symbol, it is assumed to be an ISO8583 message.

Transnet 1.3 Client Server

The Transnet 1.3 Client Server receives requests from XPS through calls into ntmapi.dll. These are exclusively ISO8583 requests and are sent directly to the Transnet Server. The responses are received by the Client Server and sent back to XPS through the dll.

The Transnet 1.5 Server accepts and responds to the Client Server just as the Transnet 1.3 Server did. Any APMs that were ported to Transnet 1.4 will work using the Transnet 1.3 Client Server and the Transnet 1.5 Server.

Transnet 1.3 TPSClient

The Transnet 1.3 TPSClient receives requests in XML format and passes them to the Transnet Server for processing. If configured to do so, as in EFT requests, the TPSClient converts them to ISO8583 format prior to passing them on. When the response is received from the server it is converted to XML if required, and passed back to the application.

The Transnet 1.5 Server accepts and responds to the TPSClient just as the Transnet 1.3 Server does. Any APMs that are ported to Transnet 1.4 will work using the Transnet 1.3 TPSClient and the Transnet 1.5 Server.

Transnet 1.4 Site Client

The Transnet 1.4 Site Client receives requests in XML format, creates a corresponding message object, serializes the object and passes it to the Transnet Server. It receives serialized object responses from the Transnet Server, de-serializes the response object and passes the response back to the application. The Transnet 1.5 Server works identically to the Transnet 1.4 Server for the Transnet 1.4 Site client.

Transnet 1.5 Server

The following is a description of how the Transnet 1.5 Server determines how the message was produced.

- The method onTransactionMessages receives any incoming messages from the NTM server. It recognizes whether the incoming message is TN1.3 or TN1.4 format. When a message comes in, the header is read in and determined to be either a TN1.3 or a TN1.4 message. If the message is a TN1.4 message, then readObject is called to de-serialize the object and processing continues as in TN1.4 Server. If the message is a TN1.3 message, then a determination is made whether the incoming data is in XML or in ISO8583 format. (The methods isTNXMLMessage and isTNBytesMessage are added to make this determination.) If the incoming message is a TN1.3

XML message, then a TNXMLMessage object is created and initialized. If the incoming message is a TN1.3 ISO8583 message, then a TNBytesMessage object is created and initialized. All code downstream from here processes as if it were a TN1.4 de-serialized TNBytesMessage. The code which follows this only sees the TNXMLMessage or TNBytesMessage. The only difference between a TN1.3 message and a TN1.4 message is the header.

- The method sendMessage is called to respond from the Server to the client. Prior to actually sending the response, the header is checked to see if it is a TN1.3 or TN1.4 message. If the message is a TN1.4 message, the Object is serialized through writeObject. If the message is a TN1.3 message, then a check is made to see if it was a TNXMLMessage or a TNBytesMessage. If it was a TNXMLMessage, then a byte array is created from the XML response and sent. If it was a TNBytesMessage, then the byte array is extracted from the object and sent.
- TPSHeader contains all of the header information for either TN1.3 or TN1.4 messages. The methods readExternal and writeExternal read the message header and determine whether it is a TN1.3 or TN1.4 message by looking at the header field userHeaderLen. If userHeaderLen is 22, the message is a TN1.3 message, if userHeaderLen is not 22 the message is a TN1.4 or later message. The user part of the header is in TNTPSUserHeader.java
- TPSUserHeader contains the user header information for TN1.4 messages. If it is a TN1.3 message, then it also has the TN1.3 user header info. The methods readExternal and writeExternal to read the message user header. In addition, for TN1.3 messages TN1.4 info is generated from the TN1.3 info. This allows the message header to look like a TN1.4 header in the downstream code. Two new methods were added to this class. The method setTransnet13 (Boolean) is added to say this is a TN1.3 message or TN1.4 or later message. The method isTransnet13() is added to tell whether this is a TN1.3 message or TN1.4 or later message.

8 Configuring Generic Settings for APMs

This section provides instructions for configuring the basic Centralized EFT settings to support your Application Processing Modules (APMs).

Overall configuration process

This section describes the overall procedures required to configure and run the supported Application Processing Modules used to facilitate the credit, debit, settlement, and checking services. Each general step listed here also references more detailed setup instructions provided elsewhere in this guide.

To configure and run Application Processing Modules:

1. Before you start:
   - Install your SQL database and Centralized EFT/File Transfer components, as required.
   - Start your SQL server, and then your browser program.
2. Start Centralized EFT, and log in. Then configure the generic settings required to run the APMs (see "Configuring Generic Settings for APMs" on page 245.)
3. Configure the XML and related configuration files for the APMs you will be using, as required, and save your changes (see "Configuring the APM XML Files" on page 191.)
4. Stop your server, and then your SQL server.
5. Ensure that all your services are set to "Automatic" and "Start".
6. Restart your SQL server, and then your server.

Configuring Centralized EFT for the APMs

Before you can fully configure and use your APMs to transfer message and authorization data, you must configure a set of generic properties within the Centralized EFT user interface. To use the APMs, you must first:

- Configure a Message Format
- Configure a Route Definition
- Configure a Module
- Configure a Server

This section provides the complete directions and configuration information required for you to set these generic properties. For more information about the functions and settings available within the Centralized EFT user interface, see the *Centralzied EFT User Guide*.

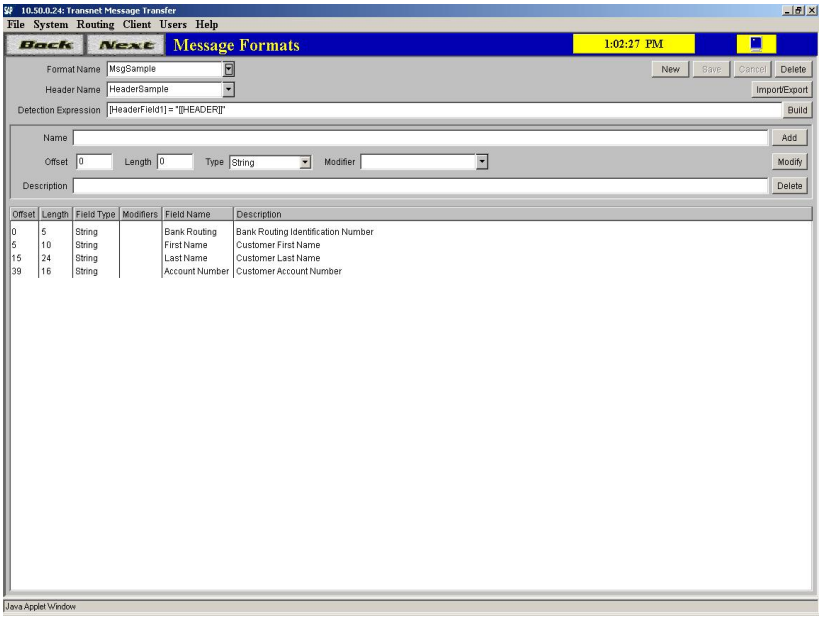
Before you complete any of these processes, you will need to:

- Install your SQL database and Centralzied EFT/File Transfer components, as required.
- Start your SQL server, and then your browser program.
- Start Centralized EFT, and log in.

Note: For all of the following procedures, type or enter settings exactly as described, using the same case shown, and with no additional spaces.

To configure the APM Message Format:

1. On the main menu, click Server, and then click Message Formats. The Message Formats screen appears.



2. Click New.
3. In the Format Name field, type transnetmessage, and then click Save.
4. Next, create the first field for this message format. To do this:
   - In the Name field, type clientID.
   - In the Offset field, type 0.
   - In the Length field, type 2.
   - In the Type drop-down list, select the Binary type.
   - Click Add. The field information appears in the table section in the lower half of the screen.
5. Next, create a second field for this message format. To do this:
   - In the Name field, type messagetype.
   - In the Offset field, type 2.
   - In the Length field, type 4.
   - In the Type drop-down list, select the String type.
   - Click Add. The field information appears in the table section in the lower half of the screen.
6. Next, create a third field for this message format. To do this:

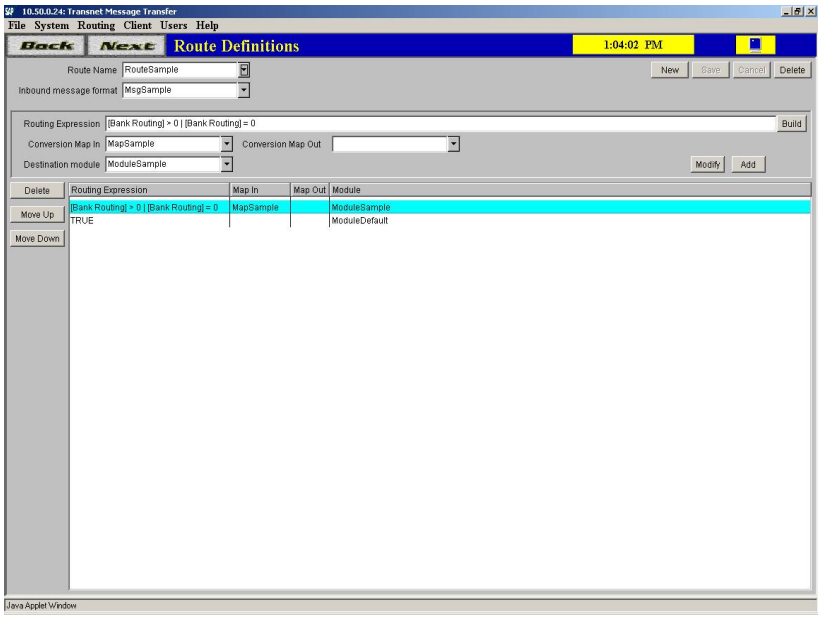
- In the Name field, type version.
- In the Offset field, type 6.
- In the Length field, type 4.
- In the Type drop-down list, select the String type.
- Click Add. The field information appears in the table section in the lower half of the screen.

7. Click Save.
8. Next, add the Detection Expression for this message format. To do this:
   - Click Build. the Expression Builder box appears.
   - In the Field Name drop-down list, select Message Type.
   - In the Conversion drop-down list, select Case Insensitive.
   - In the Comparison drop-down list, select =.
   - In the Value field, type TNET.
   - Click to select the String bubble.
   - Click OK. The information you added appears in the Detection Expression field.
9. Click Save.

Note: If the Save button does not activate when required, click or type elsewhere on the screen (for example, type an X in the Name field) to activate it (the type on the button will turn black again), and then click it.

To configure the APM Route Definition:

1. On the main menu, click Server, and then click Route Definitions. The Route Definitions screen appears.

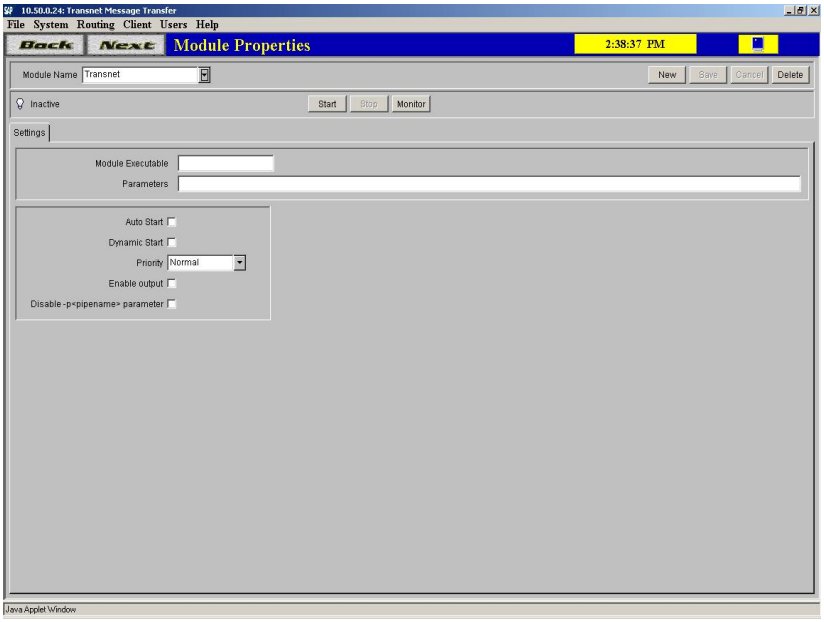


2. Click New.
3. In the Route Name field, type transnetroute, and then click Save.

4. In the Inbound Message Format drop-down list, select the transnetmessage format you just created.
5. Add the Routing Expression for this route definition. To do this:
   - In the Routing Expression field, type TRUE.
   - In the Destination Module field, type transnet.
   - Click Add. The routing expression information appears in the table section in the lower half of the screen.
6. Click Save.

To configure the APM Module:

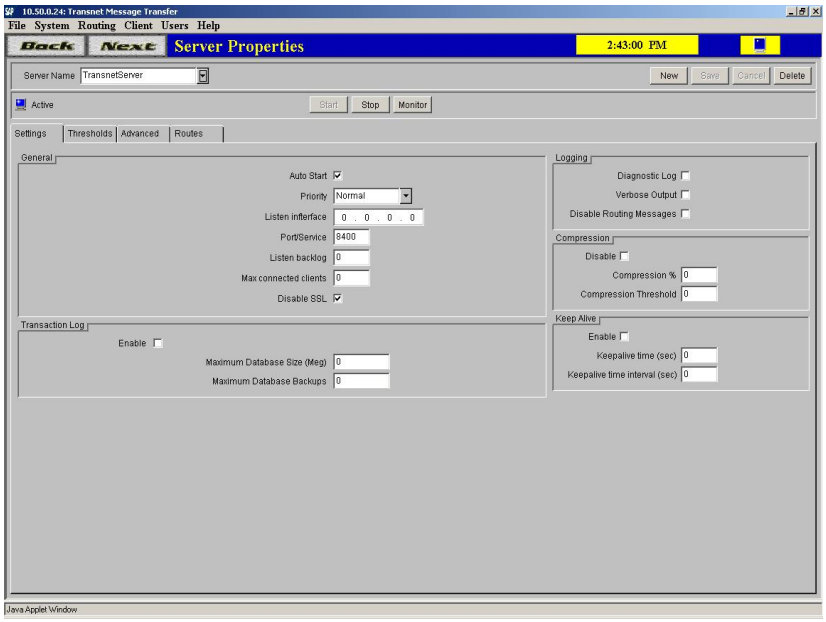
1. On the main menu, click Server, and then click Modules. The Modules screen appears.



2. Click New.
3. In the Module Name field, type transnet.
4. Ensure that the following checkboxes are left unchecked:
   - Auto Start
   - Enable Output
   - Disable -p<pipename> parameter
5. Click Save. The new module appears in the table section in the lower half of the screen.

To configure the APM Server:

1. On the main menu, click Server, then click Servers, and then click Server Properties. The Server Properties screen appears.



2. In the Server Name drop-down list, select ServerSample, and then click Delete.
3. In the Server Name field, type transnetserver, and then click Save.
4. Click the Settings tab, and configure general settings for this server as follows:
   - In the Port/Service field, type 8400.
   - Ensure that the following checkboxes are selected:
     - Auto Start
     - Enable Trace
     - Disable SSL

Note: The Centralized EFT Server transmits data, by default, on port 8400. This port should therefore be kept clear for data transmissions, or another port must be configured for use with the Centralized EFT Server.

---

5. Click the Routes tab, and configure routing settings for this server as follows:
   - In the drop-down list, select the transnetroute you created earlier.
   - Click Add Route. The route information appears in the table section in the Routes tab.
6. Click Save.
7. The server should now read Active, and the client should now be able to connect to it. To test this, do the following:
   - Open a DOS prompt.
   - At the C:> prompt, type netstat -na, and then press **<Enter>**.
   - In the resulting report data, look for the following line:

     ```
     TCP 123.0.0.1:8400 123.0.0.2:1055 ESTABLISHED
     ```

where the first IP address and port are for your server machine, and the second IP address and port are for your client machine.

- Close the DOS prompt.

APM Configuration File Reference

This section provides reference information for the XML files used to configure APM settings. Use this section as a troubleshooting reference, or to review default setting information for these files. Files described include:

Transnet.xml — Centralized EFT server

This file contains basic settings for your Centralized EFT server. This file is located by default at *root*:\Program Files\transnet\config on your Centralized EFT server machine.

```
<Transnet SystemID="10_1_1">
<!-- SAMPLE SERVER SIDE TRANSNET.XML -->
    <Logging>logging.properties</Logging>
    <!-- The ConnectionManager Service tag set defines all of the
    connections used for your TMT message movement.
    Note that the forcedShutdownWaitTime attribute is measured in
    milliseconds. -->
    <Service name="ConnectionManager" description="Transnet Connection
    Manager"
    class="com.triversity.transnet.core.tms.connection.TNConnectionManager
    " forcedShutdownWaitTime="5000">
        <ActivationInterval value="10" unit="seconds" />
        <!-- In the TPSServer Connection tag set, you configure basic
        settings for your TMT server connections.
        The TPS PipeName attribute is taken from the installation settings;
        it must match the name of the generic APM Module set in the TMT user
        interface; the default name is "transnet".
        Set the TPSDebugLogging attribute to either "true" or "false",
        depending on whether you want to run detailed logging (usually used
        for setup and troubleshooting purposes only).
        The TransactionBufferSize attribute must be set to a value (in
        bytes) greater than your maximum anticipated TLOG size. A default
        value of 80000 is sufficient if you are only running credit
        services; a default value of 4000000 is preferred if you are also
        running TLOG trickling. The same setting must also be configured in
        the TMT client transnet.xml file, where it should be set to a
        default value of 100 bytes less than that set at the server. -->
        <Connection name="TPSServer"
        class="com.triversity.transnet.core.tms.tps.TPSServerHandler"
        TPSPipeName="transnet" TPSDebugLogging="true"
        TransactionBufferSize="4000000" />
```

```
<Connection name="TlogAdapter"
class="com.triversity.transnet.core.apm.APMConnectionHandler">
    <APM name="Tlog"
    class="com.triversity.transnet.core.adapter.AdapterContainer"
    maximumCount="1" minimumCount="1" intervalMillis="4">
        <AllowdIdleTime value="1" unit="hour" />
        <Adapter
        class="com.triversity.transnet.core.adapter.TLogAdapter"
        outputDirectory="logs" />
    </APM>
</Connection>
<!-- The StoreTlogLoader Connection settings provide information
for the Transnet Directory Monitoring Service. This service
monitors the designated directories for new bank data and store
data TLog files, and loads them into the Transnet Manager reporting
system for settlement and reconciliation. -->
<Connection name="StoreTlogLoader"
class="com.triversity.transnet.core.tms.file.DirectoryConsumerMoni
tor" interval="1000" target="data" working_directory="c:/temp">
    <adapter
    class="com.triversity.transnet.core.reconciliation.Reconciliati
    onFileConsumer">
        <Reconciliation chain="10" service="Store"
        tableName="PaymentDetail_Store" archiveFolder="C:\program
        files\Transnet\logs\archive"
        merchantFileName="Merchants.xml">
            <Parser name="TlogFileParser"
            class="com.triversity.transnet.core.tlog.TlogFileParser"
            />
        </Reconciliation>
    </adapter>
</Connection>
<!-- For each financial APM or credit service provider you are
using, you must have a Connection tag set in this file, as with the
NovaAPMConnection section below. The Nova, WildCard, and FNMS
financial APMs all use the same generic APM setting structure. The
FDNorth, Concord, and Tender Retail (for example GPS Canada and GPS
USA) based APMs have unique tag structures; examples of these
follow later in the file.
In each section, you must specify a name for the connection, as
well as the general name for the EFT provider.
You must also specify a Destination ID number (one for a single
connection; successive numbers for each additional connection), and
provide the IP address and port number (default 2000X) of the
service provider. -->
<Connection name="NovaAPMConnection"
class="com.triversity.transnet.core.eft.EFTService"
creditServiceHandlerClassName="com.triversity.transnet.xtension.cr
edit.nova.NovaHandler" eftProviderName="Nova"
transactionLifeSpan="10000">
    <APMConnection
    connectionClass="com.triversity.transnet.xtension.credit.nova.N
    ovaSocketConnection">
        <Destination id="1" name="127.0.0.1:20001" />
    </APMConnection>
</Connection>
<!-- In the FDNorth APM Connection settings, the
transactionLifeSpan setting (measured in milliseconds) indicates
the amount of time after which a request is returned to Transnet if
no response is generated at the service provider. This setting
should be slightly less (-1000 milliseconds) than the corresponding
setting configured at the client, which in turn should be less than
```

```
that configured in the Route Definition, then the Routing Rule,
then the Client.conf and ClientAPI.conf files, and finally the
Transactionware GM Xpress server POS.ini file.

The minimumCount and maximumCount attributes denote the number of
connections; this should be determined by the size of your system
and the volume of data expected. -->
<Connection name="FDNorth"
class="com.triversity.transnet.core.apm.APMConnectionHandler">
    <APM name="FDNorth" minimumCount="1" maximumCount="1"
    intervalMillis="4"
    class="com.triversity.transnet.xtension.credit.firstdata.north.
    FDNorthCreditThread"
    connectionClass="com.triversity.transnet.core.apm.SocketAPMConn
    ection" transactionLifeSpan="10000">
        <AllowdIdleTime value="1" unit="hour" />
        <Connection id="1" name="127.0.0.1:20002" />
    </APM>
</Connection>
<!-- The GPSCA Connection settings show an example of a Tender
Retail based APM connection information set. -->
<Connection name="GPSCA"
class="com.triversity.transnet.core.apm.APMConnectionHandler">
<!-- The minimumCount and maximumCount attributes denote the number
of connections; this should be determined by the size of your
system and the volume of data expected. -->
    <APM name="GPSCA" minimumCount="1" maximumCount="1"
    intervalMillis="4"
    class="com.triversity.transnet.xtension.credit.tenderretail.TRR
    eaderWriter"
    connectionClass="com.triversity.transnet.xtension.credit.tender
    retail.TRSocket" TransactionLifeSpan="10000"
    ReadObjectTimeout="30000" UseDebitAccountType="true"
    SettlementRetries="3" ReconcileRetries="4"
    ReconcileFileName="TRReconcileReport.txt"
    SettleFileName="TRSettleReport.txt">
        <ErrorReporter
        class="com.triversity.transnet.core.util.FileReporter"
        fileame="TNReport.txt" reportTime="18:00" />
        <AllowdIdleTime value="1" unit="hour" />
        <Connection id="1" name="127.0.0.1:3858" />
    </APM>
</Connection>
<!-- The ConcordAPMConnection settings show an example of a Concord
Services based APM connection information set. -->
<Connection name="ConcordAPMConnection"
class="com.triversity.transnet.core.finance.xml.XMLFinancialMessag
eHandler" />
<Connection name="JBOSS"
class="com.triversity.transnet.core.tms.jms.TNJMSClientHandler"
sendTo="FromTransnet" receiveFrom="ToTransnet" durableName="">
    <JMSContext name="JBOSSMQ"
    contextBroker="org.jnp.interfaces.NamingContextFactory"
    providerURL="jnp://127.0.0.1:1099"
    urlPackage="org.jboss.naming" topicBroker="ConnectionFactory"
    queueBroker="ConnectionFactory" username="" password="">
        <jndiProperty name="java.naming.rmi.security.manager"
        value="no" />
    </JMSContext>
</Connection>
<!-- The TSVAConnection settings show an example of a Triversity
Stored Value Application APM connection information set. The TSVA
```

```
APM provides the connectivity required to perform centralized
stored value card (giftcard) activation, tracking, and
authorization using the TSVA system via TMT. -->
<Connection name="TSVAConnection"
class="com.triversity.transnet.core.tms.jms.TNExternalJMSConnectio
n" sendTo="topic/StoredValueRequest" receiveFrom="topic/
StoredValueResponse" durableName="">
    <JMSContext name="JBOSSMQ"
    contextBroker="org.jnp.interfaces.NamingContextFactory"
    providerURL="jnp://127.0.0.1:1099"
    urlPackage="org.jboss.naming" topicBroker="ConnectionFactory"
    queueBroker="ConnectionFactory" username="" password="">
        <jndiProperty name="java.naming.rmi.security.manager"
        value="no" />
    </JMSContext>
</Connection>
<Connection name="OpenJMS"
class="com.triversity.transnet.core.tms.jms.TNJMSClientHandler"
sendTo="FromTransnet" receiveFrom="ToTransnet" durableName="">
    <JMSContext name="OpenJMS"
    contextBroker="org.exolab.jms.jndi.rmi.RmiJndiInitialContextFac
    tory" providerURL="rmi://localhost:1099/JndiServer"
    urlPackage="" topicBroker="JmsTopicConnectionFactory"
    queueBroker="JmsQueueConnectionFactory" username="" password=""
    />
</Connection>
<!-- The TLog Connection settings provide information for TLog
trickling (queueing) within the TMT system. -->
<Connection name="TETLog"
class="com.triversity.transnet.core.tms.connection.TNMessageConsum
erConnection">
    <adapter
    class="com.triversity.transnet.xtension.ixretail.TETlogAdapter"
    >
        <Configuration path="C:\Tlog" root="Transactions" />
    </adapter>
</Connection>
<Connection name="ABCTextMonitor"
class="com.triversity.transnet.core.tms.file.DirectoryConsumerMoni
tor" interval="20000" target="abc.txt">
    <adapter
    class="com.triversity.transnet.core.tms.file.samples.SampleFile
    ConsumerOnly" />
</Connection>
<!-- The Allegiance1to1 Connection settings show an example of an
Allegiance 1-to-1 APM connection information set. The Allegiance
1-to-1 APM provides the connectivity required to perform
centralized loyalty program tracking and authorization using the
Triversity Allegiance CRM system via TMT. -->
<Connection name="Allegiance1To1"
class="com.triversity.transnet.core.tms.connection.TNMessageConsum
erConnection">
    <adapter
    class="com.triversity.transnet.xtension.allegiance.AllegianceAd
    apter" encoding="UTF-16" serverName="AllegianceServerDetails">
        <Pool minimumSize="1" maximumSize="1000"
        poolCleaningIntervalMillis="1000" allowedIdleTime="60000" />
    </adapter>
</Connection>
<!-- The ISO8583PassThrough Connection settings show an example of
an ISO8583 APM connection information set. The ISO8583 APM enables
```

```
your retail POS system to communicate with credit service providers
using socket communications. The APM connects to a published port
on the host at the service provider using TCP sockets (stream
sockets). -->
<Connection name="ISO8583PassThrough"
class="com.triversity.transnet.core.tms.ip.TNBytesSocketClient"
minimumSequenceNumber="0" host="127.0.0.1" port="20003"
maximumSequenceNumber="999999">
    <MessageConverter
    class="com.triversity.transnet.core.eft.converter.XMLMessageByt
    esConverter" />
</Connection>
<!-- The MonitorConnection settings show an example of a Credit
Monitor connection information set. -->
<Connection name="MonitorConnection" class =
"com.triversity.transnet.core.monitor.APMMonitor"
IntervalSeconds="10" ResponseTimeoutSeconds="10" >
    <Message>
        <FinancialMessage>
            <Originator>
                <Chain>10</Chain>
                <Store>1</Store>
                <Register>1</Register>
            </Originator>
            <MessageType>
                <Function>SignOn</Function>
                <Modifier>Original</Modifier>
                <AccountType>CreditCard</AccountType>
                <StoreAndForward> false </StoreAndForward>
                <AdditionalModifier>Original</AdditionalModifier>
            </MessageType>
            <Request>
                <TransactionNumber>38</TransactionNumber>
                <SequenceNumber>38</SequenceNumber>
                <TransactionDateTime>20020215231010.000</
                TransactionDateTime>
            </Request>
        </FinancialMessage>
    </Message>
    <PerformanceMonitor
    class="com.triversity.transnet.core.alerts.PerformanceAlerter"


    ResponseTimeOutAlerter.RepeatThreshold="2"
    ResponseTimeOutAlerter.AlertIntervalMinutes="15"
    ResponseThresholdAlerter.RepeatThreshold="0"
    ResponseThresholdAlerter.AlertIntervalMinutes="15"
    ResponseThresholdAlerter.RepeatCount="1"
    ResponseThresholdAlerter.ResponseThresholdSeconds="15"


    UnreachableAlerter.RepeatThreshold="2"
    UnreachableAlerter.AlertIntervalMinutes="15"


    UnroutableAlerter.RepeatThreshold="0"
    UnroutableAlerter.AlertIntervalMinutes="15" >
        <Reporter mailServer = "142.67.6.150" subject = "Transnet
        Monitor Test Alert" from = "" >
```

```
            <To address="" />
            <CopyTo address="" />
        </Reporter>
    </PerformanceMonitor>
</Connection>
</Service>
<!-- The MessageManager Service tag set defines all of the message
types, routes, and routing rules used for your TMT message movement.
Each message type you will use requires a MessageDescriptor tag section
naming the message type, plus a corresponding route (where the message
goes) and routing rule (what is to be done with it). Note that multiple
routes may be attached to the same routing rule.-->
<Service name="MessageManager" description="Transnet Message Manager"
class="com.triversity.transnet.core.tms.message.TNMessageManager"
forcedShutdownWaitTime="5000">
    <ActivationInterval value="10" unit="seconds" />
    <MessageDescriptor name="DefaultFinance" root="FinancialMessage">
        <Check xmlType="text" name="Originator.Chain"
        dataType="number">
            <Range minimum="1" maximum="999" />
        </Check>
        <Check xmlType="text" name="Request.CardDetail.AccountNumber"
        dataType="number" substring="0,4">
            <Range minimum="0" maximum="9999" />
        </Check>
        <Check xmlType="text" name="Request.CardDetail.CardType"
        dataType="text">
            <DataText value="VISA" />
            <DataText value="MASTERCARD" />
        </Check>
        <Check xmlType="text" name="MessageType.Function"
        dataType="text">
            <DataText value="ALL" />
        </Check>
    </MessageDescriptor>
    <MessageDescriptor name="ISOMessage" root="BytesMessage" />
    <MessageDescriptor name="TLog" root="Transactions" />
    <Route name="TPSFinanceRoute" connectionName="10">
        <Queue name="TPSFinanceQueue" limit="Unlimited"
        class="com.triversity.transnet.core.tms.queue.TNMessageQueueHan
        dler">
            <Addition trigger="always" />
            <Flushing trigger="online" intervalSeconds="300"
            countThreshold="5" burstCount="5">
                <InactiveDuration day="sun" startTime="0900"
                endTime="2000" />
                <InactiveDuration day="mon" startTime="1500"
                endTime="2000" />
                <InactiveDuration day="mon" startTime="0900"
                endTime="1130" />
            </Flushing>
        </Queue>
    </Route>
    <Route name="TlogRoute" connectionName="Tlog" />
    <Route name="FDNorth" connectionName="FDNorth" />
```

```
            <Route name="TETlogRoute" connectionName="TETLog" />
            <Route name="Nova" connectionName="Nova" conversion=""
            class="com.triversity.transnet.core.tms.route.TNBytesMessageRoute"
            />
            <RoutingRule message="ISOMessage">
               <Route name="Nova" timeoutSeconds="15" />
            </RoutingRule>
            <RoutingRule message="DefaultFinance">
               <Route name="FDNorth" timeoutSeconds="15" />
            </RoutingRule>
            <RoutingRule message="TLog">
               <Route name="TETlogRoute" timeoutSeconds="15" />
            </RoutingRule>
         </Service>
         <!-- Use the “Database” tag section to define your rdbms database
         settings (particular to your SQL database).
         The URL attribute must contain the URL of your Transnet SQL host
         machine, the port being used (the default is 1433), and the name of
         your Transnet SQL database. Also, you must ensure that your User and
         Password settings match those set in your SQL database.-->
         <Database type="rdbms" Driver="com.inet.tds.TdsDriver"
         Url="jdbc:inetdae7://127.0.0.1:1433/transnetDBname" User="sa"
         Password="1">
            <ConnectionPool Debug="false" Name="TransnetJDBCPool"
            CleaningIntervalSeconds="60" MinimumConnections="1"
            MaximumConnections="10" ConnectionLifeSeconds="1800"
            MaximumUseCount="25" />
         </Database>
         <!-- The ThreadPool tag set defines the number of threads used for
         message transport. The MinimumThreads and MaximumThreads settings
         should be based on the size of your system, and the volume of messaging
         expected; contact your system administrator for assistance.-->
         <ThreadPool Name="TransnsetThreadPool" Debug="false"
         CleaningIntervalSeconds="60" MinimumThreads="5" MaximumThreads="300" /
         >
      </Transnet>
```

Transnet.xml — Tomcat

This file contains configuration information for your Web-based reporting system. It is located by default at *root*:\Program Files\transnet\tomcat\webapps\transnet on any machine on which you have installed the Tomcat Web server applications.

```
<?xml version="1.0" ?>
<!-- THIS TRANSNET.XML IS FOR TOMCAT/TRANSNET REPORTING CONFIGURATION ONLY
-->
<TransnetServerConfiguration>
    <!-- The MeterTable name must match the database table used for loading
    data to the Response Time Analysis reporting system. The default is
    “transnet_meter”.-->
    <MeterTable name="transnet_meter" />
    <!-- The MerchantFile tags specify the APMs or service providers you
    are using, and indicate the naming conventions for each corresponding
    XXXMerchant.xml configuration file. The names of your XXXMerchant.xml
    files must correspond to those listed here exactly.-->
    <MerchantFile name="NovaMerchant.xml" service="Nova"/MerchantFile>
    <MerchantFile name="WildCardMerchant.xml" service="WildCard"/
    MerchantFile>
    <!-- The MismatchComparison tag sections define the settings for your
    mismatch reports in Transnet Manager. There must be a section for each
    chain/service combination in your system, with at least one field
    identified for comparison. The field list indicates which files must be
    compared between the Store TLOG, Service Provider, and Transnet
    transaction files. These settings are determined by comparing the file
    formats required by each system, and should be configured only in
    consultation with your qualified Triversity personnel.
    The SAFFileLocation setting refers to the Store and Forward file. This
    is used in the case where the store and bank send transaction
    information to Transnet at different rates, so the information is
    stored until all files are present and can then be reconciled. The
    default location for the creation of this file is “.”, which represents
    the current working directory (typically, C:\Program
    Files\transnet\tomcat\webapps\transnet).-->
    <MismatchComparison SAFFileLocation=".">
        <Chain name="ABC" id="10">
            <Service name="Nova">
                <Mismatch SAFFileLocation=".">
                    <Field id="1" name="chain" />
                    <Field id="2" name="store" />
                    <Field id="3" name="register" />
                    <Field id="4" name="tender_type" />
                    <Field id="5" name="card_type" />
                    <Field id="6" name="card_num" />
                    <Field id="7" name="amount" />
                    <Field id="8" name="approval_code" />
                    <SortByRegister>true</SortByRegister>
                    <HiddenReportDate>none</HiddenReportDate>
                    <BusinessDateCutOverTime>020000</
                    BusinessDateCutOverTime>
                </Mismatch>
            </Service>
        </Chain>
        <Chain name="XYZ" id="20">
```

```
            <Service name="WildCard">
               <Mismatch SAFFileLocation=".">
                  <Field id="1" name="chain" />
                  <Field id="2" name="store" />
                  <Field id="3" name="register" />
                  <Field id="4" name="txn_num" />
                  <Field id="5" name="card_num" />
                  <Field id="6" name="amount" />
                  <Field id="7" name="approval_code" />
                  <SortByRegister>true</SortByRegister>
                  <HiddenReportDate>none</HiddenReportDate>
                  <BusinessDateCutOverTime>020000</
                  BusinessDateCutOverTime>
               </Mismatch>
            </Service>
         </Chain>
      </MismatchComparison>
      <!-- The DatabaseConnectionPool section stores information relating to
      your SQL database. It must include the IP address and Port (default
      1433) used for your SQL database, as well as its administrative user
      name and password.-->
      <DatabaseConnectionPool Name="TNServerPool" Debug="false"
      Driver="com.inet.tds.TdsDriver" Url="jdbc:inetdae7://127.0.0.1:1433/
      transnet" User="sa" Password="1" CleaningIntervalSeconds="60"
      MinimumConnections="5" ConnectionLifeSeconds="1800"
      MaximumUseCount="25" />
   </TransnetServerConfiguration>
```

Merchant.xml

This APM configuration file is used to define merchant and service provider information for each chain/store/terminal in your retail hierarchy. It is located by default at *root*:\Program Files\transnet\config on your Centralized EFT server or APM install machine.

```
<?xml version="1.0" ?>
<Merchants>
   <Chain>
      <ChainId>10</ChainId>
      <Name>SellCo</Name>
      <Contact>John Smith, #123 SpringField Tel 416-234-5678</Contact>
      <BusinessCutoverTime>000000</BusinessCutoverTime>
      <Store>
         <StoreId>1</StoreId>
         <ExtStoreId>0001</ExtStoreId>
         <ServiceType>CPS2000</ServiceType>
         <Name>000000925990</Name>
         <City>Toronto</City>
         <State>ON.</State>
         <ZIPCode>M2H 2N5</ZIPCode>
         <Account>270186180880</Account>
         <SecurityCode>2001</SecurityCode>
         <ClearingCode>6666</ClearingCode>
         <ExtAccount>666641238789</ExtAccount>
         <BINNumber>4012</BINNumber>
         <ICANumber>2343</ICANumber>
         <CategoryCode>5412</CategoryCode>
         <CheckProcessor>TELECHECK</CheckProcessor>
         <Contact>Triversity Inc. #3550 Victoria Park Tel 416-791-7100</
         Contact>
         <BusinessCutoverTime>235959</BusinessCutoverTime>
         <Register>
            <RegisterId>2</RegisterId>
            <CreditName>640432</CreditName>
            <DebitName>640432</DebitName>
         </Register>
         <Register>
            <RegisterId>88</RegisterId>
            <CreditName>640432</CreditName>
            <DebitName>640432</DebitName>
         </Register>
      </Store>
   </Chain>
</Merchants>
```

TRTerminals.xml

This APM configuration file will be installed only if you have chosen a Tender Retail based APM (such as GPS Canada or GPS USA); it is located by default at *root*:\Program Files\transnet\config on your Centralized EFT server or APM install machine. It is used to define merchant and service provider information for each chain/store/terminal in your retail hierarchy.

```
<?xml version="1.0" ?>
<!DOCTYPE Terminals (View Source for full doctype...)>
<Terminals>
   <Terminal>
      <Name>TOWNSHD1</Name>
      <Chain>10</Chain>
      <Store>1</Store>
      <Register>2</Register>
      <Debit>true</Debit>
      <MerchantId>MerchantID</MerchantId>
   </Terminal>
   <Terminal>
      <Name>TOWNSHC1</Name>
      <Chain>10</Chain>
      <Store>1</Store>
      <Register>2</Register>
      <Debit>false</Debit>
      <MerchantId>MerchantID</MerchantId>
   </Terminal>
</Terminals>
```

About the Remote Server Interface

B

The File Transfer (*FT*) Remote Server Interface (or NFMI command) provides a command line interface to the FT server from any FT client computer. This can be used when it is desirable to have the FT client computer initiate FT activity rather than scheduling it from the FT server. The following section details the use of the NFMI command.

Using the Remote Server Interface

The NFMI program ships with the FT client package. On Windows systems, the NFMI program is located at `root:/Program Files/transnet/NFMClient/nfmi.exe` by default (on Linux systems at `/opt/tps/nfmc/bin`, and should be available on the path from a symbolic link in `/usr/bin/`). The NFMI program has the following syntax:

```
nfmi [-i] parm=value parm=value . . .
```

The NFMI program sends a command string and a separate list of assigned parameter pairs in the form `parm=value` as seen above, up to the FT server for execution. The command string consists of the command itself as well as any command line arguments. Each command is actually an existing executable program (most likely a script or batch file) that resides in the commands directory of your main FT server installation location (on Linux systems at `/var/tps/nfm/commands`). The remaining command line arguments are passed to the command, and the parameter pairs are available to the program in the form of OS environment variables. The variable -i optionally instructs NFMI to run in interactive mode.

Once started, the NFMI program performs the following steps:

- The NFMI program scans for a default configuration file on the local FT client called `nfmi.cfg` (located in the FT client's main install location, or at `/var/tps/nfmc`.) This text file may contain any number of parameter pairs in the form `parm=value` (one per line). These are read into the NFMI parameter list. Certain parameter names are reserved, indicating additional meaning or action. These are:
  - `NFMSERV=hostname`: Indicates the IP address or hostname of the FT server computer to contact. This is a required field.
  - `NFMNODE=nodename`: Specifies the FT node name by which the FT server knows this node. This is a required field.
  - `CONFIG=filename` :This instructs NFMI to also read in an additional configuration file of the specified name. This parameter is optional. Also, note that this is the only parameter name that may appear more than once.
  - `COMMAND=command string:` This is one of the methods of entering a command to run. If NFMI is not run interactively, then this or the following parameter is required.
  - `CMDFILE=filename`:This is another way of entering a command. This filename may contain one or several commands to run in sequence. Note that if clients will be consistently running

the same set of commands, we recommend that those commands be in the command script on the server, so they appear as a single command to the client.

- Any parameter pairs typed in as arguments to NFMI will be added to the parameter list.
- If NFMI is not run interactively, it will send up each command string along with the parameter pairs to the FT server, execute the command, print the output locally, and then exit.
- If NFMI is run interactively, the prompt `nfmi>` will appear and you will be required to type commands. Certain commands are interpreted locally, these are:
  - `help`:Prints a list of these local commands. It will then run the help command on the server if it exists.
  - `quit`: Closes and exits the NFMI program.
  - `set parm=value` :Allows you to type in a parameter value.

  Anything else typed as input is assumed to be a command string, and is sent to the server for execution.

Configuring the Remote Server Interface

Several overall steps must be performed to enable the FT remote server interface to function. These include the following:

- The actual commands must be created and placed in a `commands` directory. When a command is run from a FT client, it will be done for a particular user so that FT user restrictions can be enforced. It is possible to make certain remote commands limited to certain users. To do this, create a subdirectory in the command directory that matches the users name. Place the appropriate commands in this directory. These commands will now be available only for that user. Commands in the base directory will be available for all users.
- The FT server must have the Remote Server Interface service turned on. This is accomplished through the FT user interface in the System Settings section. Under the Server folder, the 'Enable client initiated commands' checkbox must be checked.
- The following two steps must be performed for every FT client node that is to be enabled for this service:
  - A configuration file must be created on each client with the minimum required parameters `NFMSERV` and `NFMNODE` set.
  - On the FT server the node record must have the following environment variables added:

    ```
    NFMI_USER=username
    NFMI_PASSWORD=password
    ```

    This will determine what commands the user will run with regards to the use of any embedded FT commands. These may be placed in the nodes model, if several nodes are to share the same user name.

To configure the Remote Server Interface:

1. In your NFM directory, create a configuration file called `NFMI.cfg`, with the following settings:
   - `NFMSERV=123.1.1.0:` The IP address of your Transnset server.
   - `NFMNODE=nodename:` The node name by which this station is known by your server.
   - `COMMAND=commandname.cmd` : The name of the command that you want to run on the server.
2. On your server, create a directory called `commands` in your `root:\Program Files\transnet\tftserver` directory.

Note: All commands located in the generic `commands` directory will be available to all FT users. If you want specific commands to be limited to certain users, also create a subdirectory with the desired user account name and then place the command file for those commands within that subdirectory.

3. When creating your node, add the following two lines to the Environment tab:
   - `NFMI_USER=username` : The name of the user account on the FT module.
   - `NFMI_PASSWORD=password` :The password for the user account on the FT module.

   Note that both the user account name and password are case-sensitive.
4. Create a command file in the desired directory (either the `commands` directory, or a user-specific subdirectory). To do this, use a DOS editor as the resulting text file must have a linefeed and carriage return at the end of each line, including the last line of the file. The command file may include any of the available command lines.
5. Test your settings. To do this, copy your commands to your `tftserver` directory, and rename them with a `.bat` extension. Then run them at the server to ensure that you have set all of the command syntaxes correctly.

Remote Server Interface sample configuration

The following example shows how to configure and use the NFMI command.

- At the request of each 4690 store, send the file `c:/tmp/inventory` to the `TFT server/tmp` directory. Append the store name (node name) to the end of the file.
- Set up the appropriate fileset and plan to accommodate the transfer, leaving the source node blank as it will be supplied by each separate command.

```
Fileset INVFILE:
    Source prefix=c:/tmp/ Rename prefix=/tmp/
    Source file=INVENTORY Rename file=INVENTORY.$(NFMSNODE)
Plan UPLOADINV:
    Function transfer    Source Node=(blank)    Target Node=CENTRAL
```

- Node extensions cannot be used to append the node name because they only apply to a group operation, whereas each of these transfers will occur as a separate instance of the plan. Instead the target (or rename) file uses the reserved environment variable supplied by FT to indicate the current source node name.
- The command that will actually run will be called `UPLOADINV.cmd`. The script by this name will be placed in the `commands` directory, and will contain the following line:

```
nfm callplan,UPLOADINV,%NFMNODE%
```

- This will cause the plan to be run by using the command UPLOADINV with NFMI. The `NFMNODE` environment variable will pass the desired source node name to the NFM command. This in turn will be substituted for the file extension `$(NFMSNODE)` in the fileset rename field when the plan is run resulting in the desired copy from the 4690 computer to the FT server.

```
c:/tmp/INVENTORY    | /tmp/INVENTORY.store001
```

- Each store could perform a similar copy at their request. Although the example here shows a single operation performed in the command script (in this case the NFM command), the scripts can contain several commands that will all be executed from a single command sent up from NFMI. These can be any commands available on the system, not just the NFM command.

Index

D

E

F

G

I

L

M

N

O

R

S

T

U

V

W

X