CHASE Paymentech

Orbital Gateway Java SDK

A Developer's Guide

V 6.3.0

6.30.2009

Revision History

Date	Version	Description	Author
27-Jan-04	4.0.0	Initial Revision	Sam Ayers
07-Dec-05	5.0.0	General Enhancements	Scott Monahan
15-Aug-06	5.0.2	Rebuild	Scott Monahan
31-Jan-07	6.0.0	Update to PTI42 and added support for BML and PINLess Debit	Scott Ring
31-July-08	6.1.0	Added support for Managed Billing Profiles	Scott Ring
	6.1.0	Updated Apache HttpClient from 2.0.2 to 3.1	Scott Ring
	6.2.0	Updated to improve support for Failover	Scott Ring
30-June-09	6.3.0	Update to PTI47	Scott Ring

Table of Contents

Chase Paymentech SDK for Java Applications

1 Introduction

The Chase Paymentech Java Software Developer's Kit (SDK) is a powerful and easy-to-use Java API that enables developers to quickly create and integrate real-time electronic payment transactions into eCommerce applications in a secure and reliable fashion. Chase Paymentech supports a wide variety of Credit Card, Electronic Check and Purchase Card II transactions. The SDK has also been designed to support the addition of future eCommerce transactions without having to recompile or reinstall the SDK.

Chase Paymentech SDK Features and Benefits:

Security

- Uses Secure Sockets Layer
- Open Standards Based

Programmer Highlights

- Published API utilizing Java Interfaces
- Templates-based Architecture
- Transaction Processor acquires and releases resources
- Reduces Development Time
- Extendable
- Object Oriented
- Utilizes XML for flexible B2B connectivity
- Externalized Configuration

Enhanced Communication Options

- SSL Proxy

Performance and Reliability

- Advanced Optimization for High Transaction Throughput
- Automatic failover to alternate Orbital Gateway
- Transaction retry support to eliminate duplicate transactions

Functional Features

- Industries Supported:
  - eCommerce
  - Recurring
  - Mail Order / Telephone Order
- Fraud Features:
  - Verified-by-Visa and MasterCard Secure Code
  - Address Verification
  - CVV2, CVC2, and CID
- Methods of Payment Supported
  - Credit Card
  - Gift Card
  - Bill Me Later (Salem Only)

  - Electronic Check (Salem Only)
  - European Direct Debit (Salem Only)
  - PINLess Debit

- Purchasing Card Support
  - Purchasing Card II
  - Amex Purchasing Card (Salem Only)
  - Purchasing Card III

- Other Features:
  - International Currency Support (Salem only)
  - Managed Billing
  - Orbital Gateway Profile Support
  - Soft Descriptor Support

2 Overview

"XML Templates" define the message formats required by the Chase Paymentech Orbital Gateway, while eliminating the overhead incurred by sophisticated XML parsers. Chase Paymentech's Java, COM and C++ SDK's share XML Templates, making it practical to synchronize all applications with the latest updates.

Version 6.3.0 of the Java SDK is only backward compatible with the 4.x and 5.x Java SDKs. The lack of compatibility with versions prior to 4.0 is due the changes made to the XML templates. The version 6.3.0 XML templates are much easier to implement and understand.

The implementation code of any 4.x or 5.x version of the Java SDK will only need to make coding changes if the new features added in the 6.3.0 templates are used. To take advantage of the enhancements and structural changes to version 6 of the SDK, replace the contents of the SDK's install directory with the version 6.3.0 install contents.

2.1 Architecture

2.1.1 Components

The SDK relies heavily on the software design concepts of published interfaces, composition and packaging. Along with the new published interfaces, the depth of inheritance has been optimized to be much more "shallow", and components have been carefully designed and crafted into optimal package structures.

There are five components with which an application using the Java SDK interacts:

- Configurator – loads and initializes the SDK properties, XML Templates and loggers
- Request and Template objects – sets, validates field values into the templates and produces an XML string
- TransactionProcessor – executes a transaction, managing resources and error conditions
- Response – parses attributes and field values from the response on behalf of the application

Programming details are included in Section 7, "Chase Paymentech Java SDK Developer's Reference". In addition, javadoc is provided for the new java classes (See Section 7 "Chase Paymentech Java SDK Developer's Reference).

2.1.2 Published Interfaces

Published interfaces are designed so they will not change from release to release. If an application developer uses only the published interfaces, then the resulting code is less likely to "break" as a result of a new SDK release.

It is therefore strongly recommended that application developers use only the published interfaces, identified below:

com.paymentech.orbital.sdk.interfaces.RequestIF.java
com.paymentech.orbital.sdk.interfaces.TransactionProcessorIF.java
com.paymentech.orbital.sdk.interfaces.ResponseIF.java
com.paymentech.orbital.sdk.interfaces.TemplateIF.java
com.paymentech.orbital.sdk.interfaces.ConfiguratorIF.java

The Configurator, being a singleton class, has no associated Java interface file. The Configurator class file itself is also considered to be its published interface.

com.paymentech.orbital.sdk.configurator.Configurator.java

2.2 Certification

Before aggregators or merchants can utilize any of the functionality in this section, the implementation must go through the appropriate certification process with Chase Paymentech. Please work with your Chase Paymentech Representative to schedule testing and certification as necessary.

2.3 Authentication

The Orbital Gateway supports two methods of authenticating incoming requests: Source IP authentication and Connection Username/Password authentication. What this means from a client implementation is as follows:

- For IP-based authentication, when processing transactions against both the Test and Production Orbital Gateway, the client server's Source IPs must be registered on the Orbital Gateway.
- For Connection Username/Password authentication, the Username and Password is passed in the message payload. Similar to IP-based authentication, they must match what is set up on the Orbital Gateway in order to process transactions against the Test and Production Orbital Gateway.
- Any activity presented on an IP address or Connection Username/Password that is not registered in the Orbital Gateway will result in an HTTP 412 error with the accompanying XML payload containing a ProcStatus 20412 error.
- In addition, these IP addresses/Connection Usernames must be affiliated with the Merchant IDs for which the client should be submitting transactions, specifically:
    - This does allow Third-Party Hosting service organizations presenting on behalf of other merchants to submit transactions. However, each time a new customer is added, the merchant or third-party hosting organization must ensure that the new Merchant IDs or Chain IDs are affiliated with the hosting company's IPs or Connection Usernames.
    - If the merchant expects to have more than one merchant account with the Orbital Gateway, it should have its IP addresses/Connection Usernames affiliated at the Chain-level hierarchy within the Orbital Gateway.

      Each time a new Merchant ID is added, as long as it is placed within the same Chain, it will simply work. If it is not placed within the same Chain, the additional MIDs must be affiliated with the merchant IPs or Connection Usernames. For example, we generally affiliate all Salem accounts (BIN 000001) with their Company Number (formerly called *MA #*), so all MIDs or Divisions under that Company will automatically be affiliated.
- MID-Association Failures
    - If an IP is registered, but the client presents a MID that has NOT been associated with the originating IP, the Orbital Gateway will return a HTTP 200 error with a ProcStatus of 9717.
    - If a Connection Username is registered, but the client presents a MID that has NOT been associated with the Username, the Orbital Gateway will return a ProcStatus 20412.

2.3.1 Connection Username/Password Format

The Orbital Connection Username and Password are submitted within the message payload as two separate elements:

- <OrbitalConnectionUsername>
- <OrbitalConnectionPassword>

Similar to Source IP authentication setup, the Connection Username and Password must be set up on the Orbital Gateway. Please contact your Technical Analyst or Account Representative for more information about getting set up.

The Connection Username and Password must follow specific formatting rules. Both Username and Password:

- Must be between 8–32 characters.
- Must contain at least 1 number.
- Must contain only standard English letters or digits (a-z, A-Z, 0-9).
- Cannot contain embedded spaces.

Additionally, the Connection Password is case-sensitive, while the Connection Username is not.

Note: IP-based authentication and Connection Username/Password authentication are exclusive of each other. If a merchant is set up for both IP-based authentication and Connection Username/Password authentication, request messages will be authenticated based on whether the Connection Username and/or Connection Password elements exist within the payload.

If either element does exist, the Orbital Gateway will attempt to validate the Username/Password values. If the authentication fails (for example, due to an invalid Password), the Orbital Gateway will NOT revert to IP-based authentication

3 Installation Process

This section explains how to install the Chase Paymentech Java Software Development Kit (Java SDK). It also identifies any prerequisite software for the desired installation platform. Once the SDK is installed, application developers may begin using it for creating custom eCommerce applications.

3.1 Prerequisites

The Java SDK includes a “jar” file (PaymentechSDK.jar) that contains pure Java classes. The jar file will execute in any host environment, so long as a supported version of the Java Virtual Machine (JVM) is available (see Section 3.1.1).

The Java SDK is available in a “.zip” distribution format for Windows and a “.tar.gz” format for UNIX and Linux. The “.zip” distribution includes Windows scripts; the “.tar.gz” for UNIX (or Linux) includes KornShell scripts. Both distributions contain the pure java PaymentechSDK.jar and its source code. Environment-specific scripts are necessary only if you wish to run the sample programs, or if you wish to re-build the jar file from the source code.

3.1.1 Supported Java Virtual Machines

Before you begin creating applications that use the Chase Paymentech Java SDK, verify that a supported Java Virtual Machine (JVM) is available in your host environment:

- JVM 1.5.x (latest tested 1.5.0_08)
- JVM 1.4.x

Note: Chase Paymentech tested the Java SDK executing under 1.5.0_08 and 1.4.2_13.

3.1.2 Java Security Provider Settings

Your java.security file (located in the jre/lib/security directory of your Java installation) must include the following security providers:

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.sun.rsajca.Provider
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
```

The first two providers (security.provider.1 and security.provider.2) are present in your default java.security file. The third provider must be added in order to enable the SSL classes that are used by the Java SDK.

3.2 Installing the Java SDK on Windows

3.2.1 Installation

1. Download the “.zip” formatted distribution (e.g. PaymentechSDK_6.3.0.zip).
2. Unzip the file to your desired path location (you may use WinZip, WinRAR, or any application that supports the .zip format).
3. If you have previously installed the Java SDK, merge customizations from your old environment (such as changes to the linehandler.properties file).
4. Configure the environment (see Section 3.2.2)

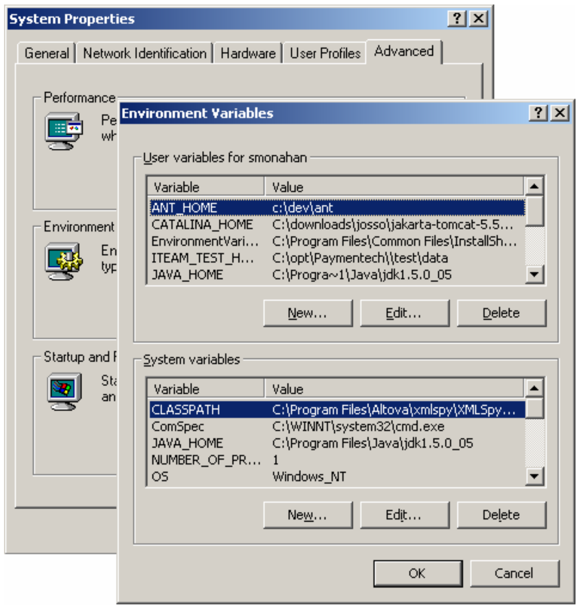
3.2.2 Environment Settings

The run scripts provided in the Java SDK use two environment variables, **JAVA_HOME** and **PAYMENTECH_HOME,** to locate runtime resources. **JAVA_HOME** must be set to your Java SDK installation. **PAYMENTECH_HOME** must be set to the SDK's installation directory. Sections 3.2.2.1 and 3.2.2.2 describe how to do this for the various Windows platforms.

If you are not using the scripts provided in the Java SDK, you are not required to set the **JAVA_HOME** and **PAYMENTECH_HOME** environment variables. Instead, you must include the java "-D" option in your script in order to set the PAYMENTECH_HOME Java system property, and you must additionally include the Java SDK library jar files (located in the lib directory) in your Java classpath. For an illustration, please refer to the "**run.bat**" script, located in the samples directory.

3.2.2.1 Configuring Environment Variables in Windows 2000 and XP

- Right Click on "My Computer" and select "Properties"
- Select the "Advanced Tab"
- Select "Environment …" button.
- Add the environment variable in the "System Variables" area. Select New... and add the following:
- Variable Name: "PAYMENTECH_HOME"
- Variable Value: c:\PaymentechSDK (substitute the directory where your SDK is installed).
- Repeat Step 4 to configure the "JAVA_HOME" environment variable.
- Click on the "Ok" button



User variables for smonahan

Variable	Value
ANT_HOME	c:\dev\ant
CATALINA_HOME	C:\downloads\josso\jakarta-tomcat-5.5...
EnvironmentVari...	C:\Program Files\Common Files\InstallSh...
ITEAM_TEST_H...	C:\opt\Paymentech\\test\data
JAVA_HOME	C:\Progra~1\Java\jdk1.5.0_05

System variables

Variable	Value
CLASSPATH	C:\Program Files\Altova\xmlspy\XMLSpy...
ComSpec	C:\WINNT\system32\cmd.exe
JAVA_HOME	C:\Program Files\Java\jdk1.5.0_05
NUMBER_OF_PR...	1
OS	Windows_NT

3.2.2.2 Configuring Environment Variables in Windows NT

- Right Click on "My Computer" and select "Properties"
- Select the "Environment" tab
- Under the "System Variables" section... select "New …"
- Add the environment variable
- Variable Name: "PAYMENTECH_HOME"

- Variable Value: c:\PaymentechSDK (substitute the directory where your SDK is installed).
- Click on the “Set” button
- Repeat Steps 4-6 to configure the “JAVA_HOME” environment variable.
- Click on the “Apply” button.

3.2.2.3 Checking for Long File Names

If your JAVA_HOME directory was defined using a “Long File Name”, then you will most likely need to modifiy you JAVA_HOME environment variable using the short name method. For example, if your JAVA_HOME was set to “C:\Program Files\Java\jre1.5.0_08”, you might need to change to “C:\Progra~1\Java\jdk1.5.0_08”.

Creating a Short File Name

Spaces are stripped, then the first 6 characters are used as the name stub, followed by a tilde (~ or "squiggle") and next required digit {1, 2, 3...}, then a dot (not stored internally) and the first 3 characters following the last dot in the LFN. The digit chosen will be the lowest that avoids a same-name clash with an 8.3 name already present in that directory; if all digits 1-9 are taken, the name stub is shortened and the number takes an extra digit to the left (i.e. ...NEXTIS~8.EXT, NEXTIS~9.EXT, NEXTI~10.EXT...)

Examples:

"A Long File Name.a.b.c.d" -> ALONGF~1.D
"My Safe Picture.gif.exe" -> MYSAFE~1.EXE
"My Safe Picture.gif.executable" -> MYSAFE~2.EXE

If the registry setting to disable numeric tales was added, these names would be created differently:

"A Long File Name.a.b.c.d" -> ALONGFIL.D
"My Safe Picture.gif.exe" -> MYSAFEPI.EXE
"My Safe Picture.gif.executable" -> MYSAFE~1.EXE

3.3 Installing the Java SDK on UNIX (or Linux)

3.3.1 Installation

1. Download the “.tar.gz” distribution (e.g. Paymentech_Java_Orbital_v6.3.0.tar.gz).
2. Copy the file to the desired directory location of your installation.
3. Uncompress the file using the gnu “gunzip” utility

```
gunzip Paymentech_Java_Orbital_v6.3.0.tar.gz
```

4. Untar the resulting .tar file using the UNIX tar command

```
tar xvf Paymentech_Java_Orbital_v6.3.0.tar
```

5. If you have previously installed the Java SDK, merge your customizations from your old environment (such as changes to the linehandler.properties file) into the new installation.
6. Configure the environment (see Section 3.3.2)

3.3.2 Environment Settings

The run scripts provided in the Java SDK use two environment variables, **JAVA_HOME** and **PAYMENTECH_HOME,** to locate runtime resources. **JAVA_HOME** must be set to your Java SDK installation. **PAYMENTECH_HOME** must be set to the SDK's installation directory. Section 3.3.2.1 describes how to do this.

If you are not using the scripts provided in the Java SDK, the **JAVA_HOME** and **PAYMENTECH_HOME** environment variables are not required. Instead, you must include the java "-D" option to set the **PAYMENTECH_HOME** Java system property. You must additionally include the Java SDK library jar files (located in the lib directory) in your Java classpath. For an illustration, please refer to the "**run.sh**" script in the samples directory.

3.3.2.1 Configuring Environment Variables in UNIX or Linux

The **PAYMENTECH_HOME** environment variable is required only if you are using the shell scripts provided in the Java SDK.

If you are using the KornShell (ksh), edit your ".profile" so that the **PAYMENTECH_HOME** and **JAVA_HOME** environment variables are set to the location of your installation:

```
export PAYMENTECH_HOME=/export/home/username/PaymentechSDK_6.3.0
export JAVA_HOME=/opt/jdk1.3.1_03
```

(Substitute the directory paths of your installations for the paths above)

If you are using shell environments other than KornShell, use the appropriate syntax of your shell to set the environment variable.

3.4 Upgrading or Re-Installing the Java SDK

3.4.1 Migrate your Customizations

If you are upgrading or re-installing your SDK and have modified portions of it (such as the linehandler.properties file), be sure that you make a backup copy of the customizations so you can merge them into the new installation.

3.4.2 Upgrading to Java SDK 6.3.0 from 6.0.0/5.x

3.4.2.1 If you **are** looking to take advantage of the new functionality in v6.3.0 you will need to install our SDK, make changes to your web app to recognize and use the new transaction type, and then build the web app using the PaymentechSDK.jar and deploy.

3.4.2.2 If you do **not** wish to take advantage of the new features offered in version 6.3.0, then you will not need to modify your java application to use the new API. Just overlay the contents of the new SDK install over your existing SDK install and modify the linehandler.properties file to your specification (if needed).

3.4.3 Upgrading to Java SDK 6.3.0 from 4.x

3.4.3.1 If you **are** looking to take advantage of the new functionality in v6.3.0 (Managed Billing) you will need to install our SDK, make changes to your web app to recognize and use the new transaction type, and then build the web app using the PaymentechSDK.jar and deploy.

3.4.3.2 If you do **not** wish to take advantage of the new features offered in version 6.3.0, then you will not need to modify your java application to use the new API. Just overlay the contents of the new SDK install over your existing SDK install and modify the linehandler.properties file to your specification (if needed). Since there may be XML fields in the 4.x version that are no longer in the 6.3.0 version, you may get FieldNotFoundExceptions thrown. To prevent this, there is a property in the linehandler.properties file, skipFieldNotFoundExceptions. The property is commented out by default. If you do not want the FieldNotFoundException thrown, then simply uncomment this property. FieldNotFoundExceptions will be written to the logs as a WARN, but will not prevent the transaction from being sent.

3.4.4 Upgrading to Java SDK 6.3.0 from 3.x (or lower)

3.4.4.1 If you are upgrading to Java SDK Version 6.3.0 from Version 3.x or lower, then you must modify your java application to use the new published API. You must also use the new linehandler.properties file format. (See Section 7, Chase Paymentech Java SDK Developer’s Reference).

4 Configuring the SDK

The Chase Paymentech Java SDK provides a wide variety of runtime options. The configuration of the SDK has been externalized to allow the runtime behavior to be tuned appropriately. Runtime configuration information is read from the ***linehandler.properties*** file located in the **%PAYMENTECH_HOME%/config** directory.

The ***linehandler.properties*** file has been preconfigured with default runtime settings that should be adequate for most environments. Some entries in this configuration file have been commented out and may not be necessary in some environments. The '**#**' character is used to indicate a commented line.

```
# This is a comment line
```

Notes:

- The SDK will not automatically detect changes made to the ***linehandler.properties*** file. The application or web server using the SDK will need to be restarted before changes will take affect.
- The values used within the linehandler.properties file are case sensitive.

4.1 General Properties

4.1.1 DTD Version

The Orbital Gateway validates incoming requests against a published XML Document Template Description (DTD). For an https request, this value is sent to the Orbital Gateway as part of the MIME header. The DTD version should not be changed except as part of a Gateway/SDK upgrade or unless instructed to do so by Chase Paymentech support.

```
################################################
# General Properties
################################################
DTDVersion=PTI44
```

4.2 Transaction Processor Properties

The Transaction Processor is a Java object that processes transactions on behalf of the application. It is responsible for managing resources, error handling and for governing the transactions.

```
################################################
# Transaction Processor Properties
################################################
TransactionProcessor.poolSize=10
TransactionProcessor.retries=2
```

4.2.1 TransactionProcessor.poolSize

Specifies the number of "engines" (concurrent protocol processing threads) that are available to the application. The default value is 10. The maximum value is 99.

4.2.2 *TransactionProcessor.retries*

Specifies the number of times the SDK will attempt to retry a transaction in the event of a technical error condition such as a momentary network outage. (Retries will occur only if the application sets a "trace number" in the request).

4.3 Response Code Configuration

```
#############################################################################################
# Response code configuration ('gateway' or 'host')
#############################################################################################
Response.response_type=gateway
```

4.3.1 *Response.response_type*

The SDK provides a parameter setting for configuring the Response objects behavior with respect to the Authorization Host Response Code value and AVS Response Code values.

- **Gateway** – If the SDK is configured in the '***gateway'*** mode, the response objects methods will report on the Orbital Gateway normalized response codes. This is the recommended approach especially if there is any possibility that the interface could process against both the Salem and Tampa host platforms.
- **Host –** If the SDK is configured in the '***host***' mode, the response objects methods will report on the ***host*** response codes.

Note: The SDK will default to the gateway.

4.4 Engine Properties

An "engine" is an object that executes a protocol, such as https, in order to send an xml request to the Orbital gateway, and to receive the response. The Engine Properties section of the linehandler.properties file specifies operational characteristics of the engine.

```
####################################################
# Engine Properties
####################################################
engine.class=com.paymentech.orbital.sdk.engine.https.HttpsEngine
engine.hostname=orbitalvar1.paymentech.net
engine.port=443
engine.hostname.failover=orbitalvar2.paymentech.net
engine.port.failover=443
engine.connection_timeout_seconds=90
engine.read_timeout_seconds=90
engine.authorizationURI=/authorize
engine.sdk_version=PaymentechSDK_6.3.0

engine.ssl.socketfactory=default

# To specify a non-default location for your truststore (cacerts) file, uncomment and edit these two lines
#engine.ssl.trustore.filename=C:/jdk1.3.1_03/jre/lib/security/cacerts
#engine.ssl.trustore.passphrase=changeit
```

4.4.1 *engine.class*

Specifies a class that implements the EngineIF interface. In essence, the engine class selection determines which protocol will be used to send the transaction to the Orbital gateway. Presently, the engine class should always be set to HttpsEngine, as indicated.

4.4.2 *engine.hostname*

Specifies the domain name or IP address of the Orbital Gateway. Chase Paymentech maintains two Orbital Gateway systems.

- Testing and certification system. The address for this system is:
  - URL: orbitalvar1.paymentech.net
  - Port: 443
- Production system. The address for this system is:
  - URL: orbital1.paymentech.net
  - Port: 443

4.4.3 *engine.port*

Specifies the port on the primary Orbital Gateway to which the engine should connect (see engine.hostname)

4.4.4 *engine.hostname.failover*

Specifies the domain name or IP address of the failover Orbital Gateway on which transactions will be executed when a failover condition occurs.

- Testing and certification system address is:
  - URL: orbitalvar2.paymentech.net
  - Port 443
- Production system address is:
  - URL: orbital2.paymentech.net
  - Port 443

Chase Paymentech exposes redundant hostname/port network endpoints to ensure high availability for the Orbital Gateway. The SDK may be configured to automatically fail-over to an alternate hostname/port if the automatic retries are exhausted against the primary hostname/port. The failover is completely transparent to the application, unless the transaction additionally fails in failover mode, in which case an error is reported to the application.

4.4.5 *engine.port.failover*

Specifies the port on the failover Orbital Gateway to which the engine should connect when a failover condition occurs.

4.4.6 *engine.connection_timeout_seconds*

The connection timeout specifies the number of seconds the engine will wait for a tcp/ip connection to the Orbital Gateway before timing out. This is the only timeout that has the possibility of a retry scenario. If the connection timeout is set lower then 90 seconds, then the retry ability is not available.

4.4.7 *engine.read_timeout_seconds*

The read timeout specifies the number of seconds the engine will wait for a tcp/ip read from the Orbital

Gateway before timing out. Since it is impossible to determine if data was successfully received by the Orbital Gateway, a read timeout will never have the possible of a retry scenario.

4.4.8 engine.authorizationURI

Specifies the Universal Resource Indicator (URI) that is used to specify the location of the DTD in the MIME header. This value should not be changed.

4.4.9 engine.sdk.version

Specifies the present version of the SDK. This value is automatically updated with each release build of the SDK, and should not be changed.

4.4.10 engine.ssl.socketfactory

This property defined the type of SSL SocketFactory that will be used to send secure transactions to the Orbital Gateway. There are 2 possible values for this property:

- default: Industry standard SSL communication

- strict: Strict (or Host) verification will make sure the SSL sessions server host name matches with the host name returned in the server certificates "Common Name" field of the "SubjectDN" entry.

4.4.11 engine.ssl.trustore.filename

Specifies the location of the trusted servers file, commonly named “cacerts”. This property should be left commented, unless you wish to change the location of your cacerts file from its default location defined in the java runtime environment.

4.4.12 engine.ssl.truststore.passphrase

Specifies the password used to access the cacerts file. This property should be left commented, unless you wish to change the location of your cacerts file from its default location defined in the java runtime environment.

4.5 Log4j Logging Properties

The Java SDK logs to two logfiles – eCommerce.log and engine.log. The SDK implements Apache Commons Logging, which allows the use of several logging implementations. By default, both loggers have been implemented with “log4j”and their configurations are specified in the log4j-config.xml file.

The Java SDK also supports live logging which allows the ablity to update the logging level without the need to restart the application. In order to make use of this functionality remove the log4j related reference from the old linehandler.properties if you intend to use it with the new Orbital SDK

The eCommerce.log contains only XML transactions and essential information that is required to interpret the XML transactions. The engine.log contains technical details and information in addition to the XML transactions.

With the exception of the logging level and log file path specification you should not modify the log4j configuration properties. If you do, it may be difficult for Chase Paymentech support staff to interpret your log files if you need support.

Note: For security purposes, the SDK will not log a customer's credit card number or CVV information. The first 12 digits of the card number are replaced with X's. The CVV information is also replaced with X's.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration>

  <appender name="eCommerce" class="org.apache.log4j.RollingFileAppender">
```

```
            <param name="File" value="${PAYMENTECH_LOGDIR}/eCommerce.log"/>
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%5p,%d,[%c] - %m%n"/>
        </layout>
    </appender>
    <appender name="engine" class="org.apache.log4j.RollingFileAppender">
            <param name="File" value="${PAYMENTECH_LOGDIR}/engine.log"/>
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%5p,%d,[%c] - %m%n"/>
        </layout>
    </appender>
    <category name="eCommerceLogger" additivity="false">
            <priority value="INFO" />
            <appender-ref ref="eCommerce"/>
    </category>
    <category name="engineLogger" additivity="false">
            <priority value="INFO" />
            <appender-ref ref="engine"/>
    </category>
    <category additivity="false" name="org.apache.commons.httpclient">
            <appender-ref ref="engine"/>
    </category>
    <category additivity="false" name="httpclient">
            <appender-ref ref="engine"/>
    </category>
</log4j:configuration>
```

Specifies the root Log4J logger instance and its debug output level. Many of the classes used in the Chase Paymentech SDK have the ability to use Log4J (such as HTTPClient), but it is unneeded to see their general logging output. Your custom classes will fall under the root Log4J definition (if you decide to use the SDK’s Log4J configuration), so you may need to create your own Log4J appender to better control your logging output destinations and levels.

4.5.1 log4j.logger.eCommerceLogger

Specifies the log level and the name of the logger instance for the eCommerce logger. The log level can be one of the following selections:

- DEBUG
- INFO
- WARN
- ERROR

- FATAL

FATAL is the highest logging level. If the FATAL level is specified, only FATAL log messages will be recorded in the logs.

DEBUG is the lowest logging level. If the FATAL level is specified, all log messages will be recorded in the log files.

For normal operation, the log level should be set to INFO. If you wish to see the XML requests and responses for all transactions, set the log level to DEBUG.

4.5.2 *log4j.appender.eCommerce*

Specifies the method by which the class that implements the log4j appender for the eCommerce log file. The "org.apache.log4j.DailyRollingFileAppender" rolls the log files every day, so that they do not grow too large. This value should not be changed.

4.5.3 *log4j.appender.eCommerce.layout*

Specifies the class that implements the layout for the logfile. This value should not be changed.

4.5.4 *log4j.appender.eCommerce.layout.ConversionPattern*

Specifies the format of the log messages. This value should not be changed.

4.5.5 *log4j.appender.eCommerce.File*

Specifies the location of your log file. You should edit this property to specify the location of your logfiles.

4.5.6 *log4j.logger.engineLogger*

Specifies the log level and the name of the logger instance for the engine logger. The log level can be one of the following selections:

- DEBUG
- INFO
- WARN
- ERROR
- FATAL

- FATAL is the highest logging level. If the FATAL level is specified, only FATAL log messages will be recorded in the logs.
- DEBUG is the lowest logging level. If the FATAL level is specified, all log messages will be recorded in the log files.
- For normal operation, the log level should be set to INFO. If you wish to see the XML requests and responses for all transactions, set the log level to DEBUG.

4.5.7 *log4j.appender.engine*

Specifies the method by which the class that implements the log4j appender for the eCommerce log file. The "org.apache.log4j.DailyRollingFileAppender" rolls the log files every day, so that they do not grow too large. This value should not be changed.

4.5.8 *log4j.appender.engine.layout*

Specifies the class that implements the layout for the log file. This value should not be changed.

4.5.9 log4j.appender.engine.layout.ConversionPattern

Specifies the format of the log messages. This value should not be changed.

4.5.10 log4j.appender.engine.File

Specifies the full path name of your log file. You should edit the path specify the desired location of your log files.

4.6 XML Templates Configuration

The Java SDK, versions 4.0 and higher, forms its XML requests using a set of templates. The location of these templates is specified in this section of the linehandler.properties file.

The SDK automatically substitutes the **PAYMENTECH_HOME** setting (specified by the "–DPAYMENTECH_HOME=" system property) for the **"%PAYMENTECH_HOME%**" pattern wherever it appears in the linehandler.properties file.

As an example, if the java JVM is launched using "-DPAYMENTECH_HOME=c:\PaymentechSDK", then the SDK will expect the full path specification for the "Void" template to be "c:\PaymentechSDK\xml\Reverse.xml" (see the setting for "XMLTemplates.Request.Reverse" below).

```
##################################################################################
# XML Templates Configuration
##################################################################################
XMLTemplates.Request.NewOrder=%PAYMENTECH_HOME%/xml/NewOrder.xml
XMLTemplates.Request.EOD=%PAYMENTECH_HOME%/xml/EOD.xml
XMLTemplates.Request.Gift Card=%PAYMENTECH_HOME%/xml/Gift Card.xml
XMLTemplates.Request.MFC=%PAYMENTECH_HOME%/xml/MFC.xml
XMLTemplates.Request.Profile=%PAYMENTECH_HOME%/xml/Profile.xml
XMLTemplates.Request.Reverse=%PAYMENTECH_HOME%/xml/Reverse.xml


# Complex Type Mappings
XMLTemplates.Request.ComplexRoot.PC3Core=%PAYMENTECH_HOME%/xml/templates/PC3Core
.inc
XMLTemplates.Request.ComplexRoot.PC3Core.RecursiveElement1=PC3LineItems
XMLTemplates.Request.ComplexRoot.PC3Core.RecursiveElement1.CountElement=PC3LineItemCou
nt
XMLTemplates.Request.ComplexRoot.PC3Core.RecursiveElement1.EnforceGreaterThanZero=yes
XMLTemplates.Request.ComplexRoot.PC3Core.RecursiveElement1.MaxCount=98
XMLTemplates.Request.ComplexRoot.PC3LineItems=%PAYMENTECH_HOME%/xml/templates/PC3
LineItems.inc
XMLTemplates.Request.ComplexRoot.PC3LineItems.ChildIndexElement=PC3DtlIndex
XMLTemplates.Request.ComplexRoot.SettleRejectBin=%PAYMENTECH_HOME%/xml/templates/Set
tleRejectBin.inc
XMLTemplates.Request.ComplexRoot.PriorAuthID=%PAYMENTECH_HOME%/xml/templates/PriorA
uthID.inc
```

4.6.1 skipFieldNotFoundExceptions

Version 6.0 will work with older versions of the SDK (allowing for plenty of upgrade time). The newer templates are a simplified structure and do not contain all of the same fields as the older templates. By default, the SDK will throw a FieldNotFoundException when it encounters a field name that is not defined in the specified XML template. If you are upgrading from a pre-PTI40 version, you have the option of changing this behavior to just write a WARN message to the logs (as opposed to stopping the transaction). Please note that this feature is only advisable for pre-PTI40 SDK users with existing code. New implementations should leave this feature commented out.

4.6.2 templateLoader

The XML templates can be either loaded from the file system or from the classpath. The default loading is from the file system where the PAYMENTECH_HOME environment variable is substituted to create the full path to the XML template. If there is a need to read the XML template the classpath, such as when packaging in a war file, just uncomment the “templateLoader” loader property and ensure that the XML templates are included in your applications classpath.

4.7 Using a Proxy

If your application server must use a network proxy to connect to the Orbital Gateway, then your scripts that start the java virtual machine must include the following system properties (note the –D options).

```
java -Dhttp.proxyHost=<proxy server name> -Dhttp.proxyPort=<proxy server port>
```

The Java SDK includes example scripts that use proxy connections (see runproxy.sh or runproxy.bat in the samples directory).

4.8 Using the Java Security Upgrade Utility

For your Java SDK to securely communicate with the Orbital Gateway, you must make sure that you have the correct Verisign certificate installed in your Java security keystore.

In the PAYMENTECH_HOME/certupdate directory you will find the correct Verisign certificate (root2028.pem) and scripts that will assist you in viewing and updating your keystores contents.

Please refer to the PAYMENTECH_HOME/certupdate/Readme.txt for a detailed explanation of the included scripts.

NOTE: Before running the update scripts, it is important that you locate the SDK instance that *you are using to run the Paymentech SDK.* It is common for Java users to have several versions of the Java SDK installed on their machine, which means several keystores.

4.9 XML Test Tool

Located in the PAYMENTECH_HOME/xmltesttool directory is a simple test tool that can be used to send an XML payload to the Orbital Gateway and dump the response.

The sendXML script takes 4 parameters:

Host (Required) – ex. orbitalvar1.paymentech.net

Port (Required) – ex. 443

XML File (Required) – ex. ..\transactions\AuthSample.xml

PTI Version (Optional) – Used when sending a pre-PTI40 based XML transaction

NOTE: This script should not be used to send production transactions.

5 Verifying the Orbital Java SDK Installation

Basic operation of the Java SDK is verified using a built-in program called the “OrbitalSDKTester”. The OrbitalSDKTester is a Java class, containing a “main” program, which is compiled by Chase Paymentech and included in the PaymentechSDK.jar. The OrbitalSDK Tester sends a single transaction to the Orbital Gateway and prints the XML response. If you are a first time user, you will need to change the hostname property in the linehandler.properties file to the VAR Orbital Gateway instance. Please contact Chase Paymentech Support for additional help on getting set-up for processing.

5.1 Configuring the Environment

Step 1: Install the SDK and configure the environment as described in Section 0, “The Orbital Gateway supports two methods of authenticating incoming requests: Source IP authentication and Connection Username/Password authentication. What this means from a client implementation is as follows:

- For IP-based authentication, when processing transactions against both the Test and Production Orbital Gateway, the client server’s Source IPs must be registered on the Orbital Gateway.
- For Connection Username/Password authentication, the Username and Password is passed in the message payload. Similar to IP-based authentication, they must match what is set up on the Orbital Gateway in order to process transactions against the Test and Production Orbital Gateway.
- Any activity presented on an IP address or Connection Username/Password that is not registered in the Orbital Gateway will result in an HTTP 412 error with the accompanying XML payload containing a ProcStatus 20412 error.
- In addition, these IP addresses/Connection Usernames must be affiliated with the Merchant IDs for which the client should be submitting transactions, specifically:
    - This does allow Third-Party Hosting service organizations presenting on behalf of other merchants to submit transactions. However, each time a new customer is added, the merchant or third-party hosting organization must ensure that the new Merchant IDs or Chain IDs are affiliated with the hosting company's IPs or Connection Usernames.
    - If the merchant expects to have more than one merchant account with the Orbital Gateway, it should have its IP addresses/Connection Usernames affiliated at the Chain-level hierarchy within the Orbital Gateway.

      Each time a new Merchant ID is added, as long as it is placed within the same Chain, it will simply work. If it is not placed within the same Chain, the additional MIDs must be affiliated with the merchant IPs or Connection Usernames. For example, we generally affiliate all Salem accounts (BIN 000001) with their Company Number (formerly called *MA #*), so all MIDs or Divisions under that Company will automatically be affiliated.
- MID-Association Failures
    - If an IP is registered, but the client presents a MID that has NOT been associated with the originating IP, the Orbital Gateway will return a HTTP 200 error with a ProcStatus of 9717.
    - If a Connection Username is registered, but the client presents a MID that has NOT been associated with the Username, the Orbital Gateway will return a ProcStatus 20412.

5.1.1 Connection Username/Password Format

The Orbital Connection Username and Password are submitted within the message payload as two separate elements:

- <OrbitalConnectionUsername>
- <OrbitalConnectionPassword>

Similar to Source IP authentication setup, the Connection Username and Password must be set up on the Orbital Gateway. Please contact your Technical Analyst or Account Representative for more information about getting set up.

The Connection Username and Password must follow specific formatting rules. Both Username and Password:

- Must be between 8–32 characters.
- Must contain at least 1 number.
- Must contain only standard English letters or digits (a-z, A-Z, 0-9).
- Cannot contain embedded spaces.

Additionally, the Connection Password is case-sensitive, while the Connection Username is not.

Note: IP-based authentication and Connection Username/Password authentication are exclusive of each other. If a merchant is set up for both IP-based authentication and Connection Username/Password authentication, request messages will be authenticated based on whether the Connection Username and/or Connection Password elements exist within the payload.

If either element does exist, the Orbital Gateway will attempt to validate the Username/Password values. If the authentication fails (for example, due to an invalid Password), the Orbital Gateway will NOT revert to IP-based authentication
Installation Process". Confirm the PAYMENTECH_HOME and JAVA_HOME environment variables are properly configured.

Step 2: Configure the %PAYMENTECH_HOME%/test/linehandler.properties file as described in Section 4, "Configuring the SDK". Note that, for convenience, the log file for the OrbitalSDKTester program are pre-configured to be in the %PAYMENTECH_HOME%/test directory.

5.2 Configuring the Test Data

The "testdata.properties" file (located in the %PAYMENTECH_HOME%/test directory) contains test data for the test transaction that the OrbitalSDKTester can send. Edit this file and put in the values that you wish to use for your configuration testing.

```
####################################################################
###### Test Data for Orbital Java SDK Transaction Tests #####
####################################################################

##### Auth Test #####
Auth.MerchantID=041756
Auth.BIN=000001
Auth.OrderID=122003SA
Auth.AccountNum=4055011111111111
Auth.Amount=100
Auth.Exp=1205
Auth.AVSname=Jon Smith
Auth.AVSaddress1=4200 W Cypress St
Auth.AVScity=Tampa
Auth.AVSstate=FL
Auth.AVSzip=33607
Auth.Comments=This is Java SDK v6.3.0
Auth.ShippingRef=FEDEX WB12345678 Pri 1
```

5.3 Running the Script

To run a basic authorization request in Windows, execute the script "run.bat". For UNIX or Linux, execute "run.sh". (Note: for the UNIX and Linux environment, before running scripts you must use the UNIX chmod command to make them executable).

5.4 Examining the Results

If your transaction is successful, you will see output similar to that below, which includes the XML for the request, the XML for the response, and the attributes that were extracted using the response object's helper methods.

If your log levels in your linehandler.properties file are set to "DEBUG", you will also be able to see the transaction request and response recorded in the eCommerce.log file and the engine.log file (Note: credit card account numbers and security values will be masked with X's). In engine.log you will also find the technical details of the transaction.

If you do not see results similar to that below, then there is a problem with your configuration. Repeat Sections 5.1 through 5.3 in order to confirm the correct configuration.

```
C:\PaymentechSDK_6.3.0\samples>build AuthSample.java

C:\PaymentechSDK_6.3.0\samples>C:\Progra~1\Java\jdk1.5.0_08\bin\jav
ac -classpath C:\PaymentechSDK_6.3.0/lib/PaymentechSDK.jar AuthSamp
le.java

C:\PaymentechSDK_6.3.0\samples>run AuthSample

C:\PaymentechSDK_6.3.0\samples>C:\Progra~1\Java\jdk1.5.0_08/jre/bin
/java -DPAYMENTECH_HOME=C:\ PaymentechSDK_6.3.0 -classpath .;
C:\PaymentechSDK_6.3.0/lib/PaymentechSDK.jar;C:\ PaymentechSDK
_6.3.0/lib/commons-logging.jar;C:\PaymentechSDK_6.3.0/lib/jsse.jar;
C:\PaymentechSDK_6.3.0/lib/jcert.jar;C:\PaymentechSDK_
6.3.0/lib/jnet.jar;C:\PaymentechSDK_6.3.0/lib/commons-httpclient-2.
0.2.jar;C:\PaymentechSDK_6.3.0/lib/regexp.jar;C:\Payme
ntechSDK_6.3.0/lib/log4j-1.2.8.jar;C:\PaymentechSDK_6.3.0/lib/xml4j
.jar;C:\PaymentechSDK_6.3.0/lib/sunrsasign.jar AuthSample

Auth Request:
<?xml version="1.0" encoding="UTF-8"?><Request> <NewOrder>             <Industr
yType>EC</IndustryType>        <MessageType>A</MessageType>          <BIN>000
002</BIN>              <MerchantID>700000000413</MerchantID>          <Termina
lID>001</TerminalID>           <CardBrand></CardBrand>        <AccountNum>4055
011111111111</AccountNum>             <Exp>1209</Exp>        <CurrencyCode>84
0</CurrencyCode>               <CurrencyExponent>2</CurrencyExponent>
<CardSecValInd>1</CardSecValInd>               <CardSecVal>111</CardSecVal>
       <DebitCardIssueNum></DebitCardIssueNum>        <DebitCardStartDate></De
bitCardStartDate>              <BCRtNum></BCRtNum>            <CheckDDA></Chec
kDDA>          <BankAccountType></BankAccountType>            <ECPAuthMethod><
/ECPAuthMethod>        <BankPmtDelv></BankPmtDelv>            <AVSzip>11111</A
VSzip>         <AVSaddress1>4200 W Cypress St</AVSaddress1>           <AVSaddr
ess2></AVSaddress2>            <AVScity>Tampa</AVScity>              <AVSstat
e>FL</AVSstate>        <AVSphoneNum></AVSphoneNum>            <AVSname>Jon Smi
th</AVSname>           <AVScountryCode></AVScountryCode>             <AVSDest
zip></AVSDestzip>              <AVSDestaddress1></AVSDestaddress1>
```

```
<AVSDestaddress2></AVSDestaddress2>            <AVSDestcity></AVSDestcity>
       <AVSDeststate></AVSDeststate>          <AVSDestphoneNum></AVSDestphoneN
um>            <AVSDestname></AVSDestname>            <AVSDestcountryCode></AV
SDestcountryCode>              <CustomerProfileFromOrderInd>EMPTY</CustomerProf
ileFromOrderInd>               <CustomerRefNum></CustomerRefNum>
<CustomerProfileOrderOverrideInd></CustomerProfileOrderOverrideInd>
<AuthenticationECIInd></AuthenticationECIInd>          <CAVV></CAVV>
<XID></XID>                          <OrderID>122003SA</OrderID>
<Amount>100</Amount>           <Comments>This is Java SDK v6.3.0</Comments>
       <ShippingRef>FEDEX WB12345678 Pri 1</ShippingRef>              <TaxInd>
</TaxInd>              <Tax></Tax>            <AMEXTranAdvAddn1></AMEXTranAdvA
ddn1>          <AMEXTranAdvAddn2></AMEXTranAdvAddn2>          <AMEXTranAdvAddn
3></AMEXTranAdvAddn3>          <AMEXTranAdvAddn4></AMEXTranAdvAddn4>
<AAV></AAV>            <SDMerchantName></SDMerchantName>              <SDProdu
ctDescription></SDProductDescription>          <SDMerchantCity></SDMerchantCity
>              <SDMerchantPhone></SDMerchantPhone>            <SDMerchantURL><
/SDMerchantURL>        <SDMerchantEmail></SDMerchantEmail>            <Recurri
ngInd></RecurringInd>          <EUDDCountryCode></EUDDCountryCode>
<EUDDBankSortCode></EUDDBankSortCode>          <EUDDRibCode></EUDDRibCode>
       <BMLCustomerIP></BMLCustomerIP>        <BMLCustomerEmail></BMLCustomerE
mail>          <BMLShippingCost></BMLShippingCost>            <BMLTNCVersion><
/BMLTNCVersion>        <BMLCustomerRegistrationDate></BMLCustomerRegistrationDa
te>            <BMLCustomerTypeFlag></BMLCustomerTypeFlag>            <BMLItem
Category></BMLItemCategory>            <BMLPreapprovalInvitationNum></BMLPreapp
rovalInvitationNum>            <BMLMerchantPromotionalCode></BMLMerchantPromoti
onalCode>              <BMLCustomerBirthDate></BMLCustomerBirthDate>
<BMLCustomerSSN></BMLCustomerSSN>              <BMLCustomerAnnualIncome></BMLCu
stomerAnnualIncome>            <BMLCustomerResidenceStatus></BMLCustomerResiden
ceStatus>              <BMLCustomerCheckingAccount></BMLCustomerCheckingAccount
>              <BMLCustomerSavingsAccount></BMLCustomerSavingsAccount>
<BMLProductDeliveryType></BMLProductDeliveryType>              <BillerReference
Number></BillerReferenceNumber>        <PCOrderNum></PCOrderNum>
<PCDestZip></PCDestZip>        <PCDestName></PCDestName>              <PCDestA
ddress1></PCDestAddress1>              <PCDestAddress2></PCDestAddress2>
       <PCDestCity></PCDestCity>              <PCDestState></PCDestState>
              </NewOrder></Request>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?><Response><NewOrderResp><IndustryType></In
dustryType><MessageType>A</MessageType><MerchantID>700000000413</MerchantID><Ter
minalID>001</TerminalID><CardBrand>VI</CardBrand><AccountNum>4055011111111111</A
ccountNum><OrderID>122003SA</OrderID><TxRefNum>45BDF8F144032EE58042B7081F9D0BE18
6FB5443</TxRefNum><TxRefIdx>0</TxRefIdx><ProcStatus>0</ProcStatus><ApprovalStatu
s>1</ApprovalStatus><RespCode>00</RespCode><AVSRespCode>F </AVSRespCode><CVV2Res
pCode>M</CVV2RespCode><AuthCode>090339</AuthCode><RecurringAdviceCd></RecurringA
dviceCd><CAVVRespCode></CAVVRespCode><StatusMsg>Approved</StatusMsg><RespMsg></R
espMsg><HostRespCode>00</HostRespCode><HostAVSRespCode>A</HostAVSRespCode><HostC
VV2RespCode>M</HostCVV2RespCode><CustomerRefNum></CustomerRefNum><CustomerName><
/CustomerName><ProfileProcStatus></ProfileProcStatus><CustomerProfileMessage></C
ustomerProfileMessage><BillerReferenceNumber></BillerReferenceNumber><RespTime>0
83858</RespTime></NewOrderResp></Response>
```

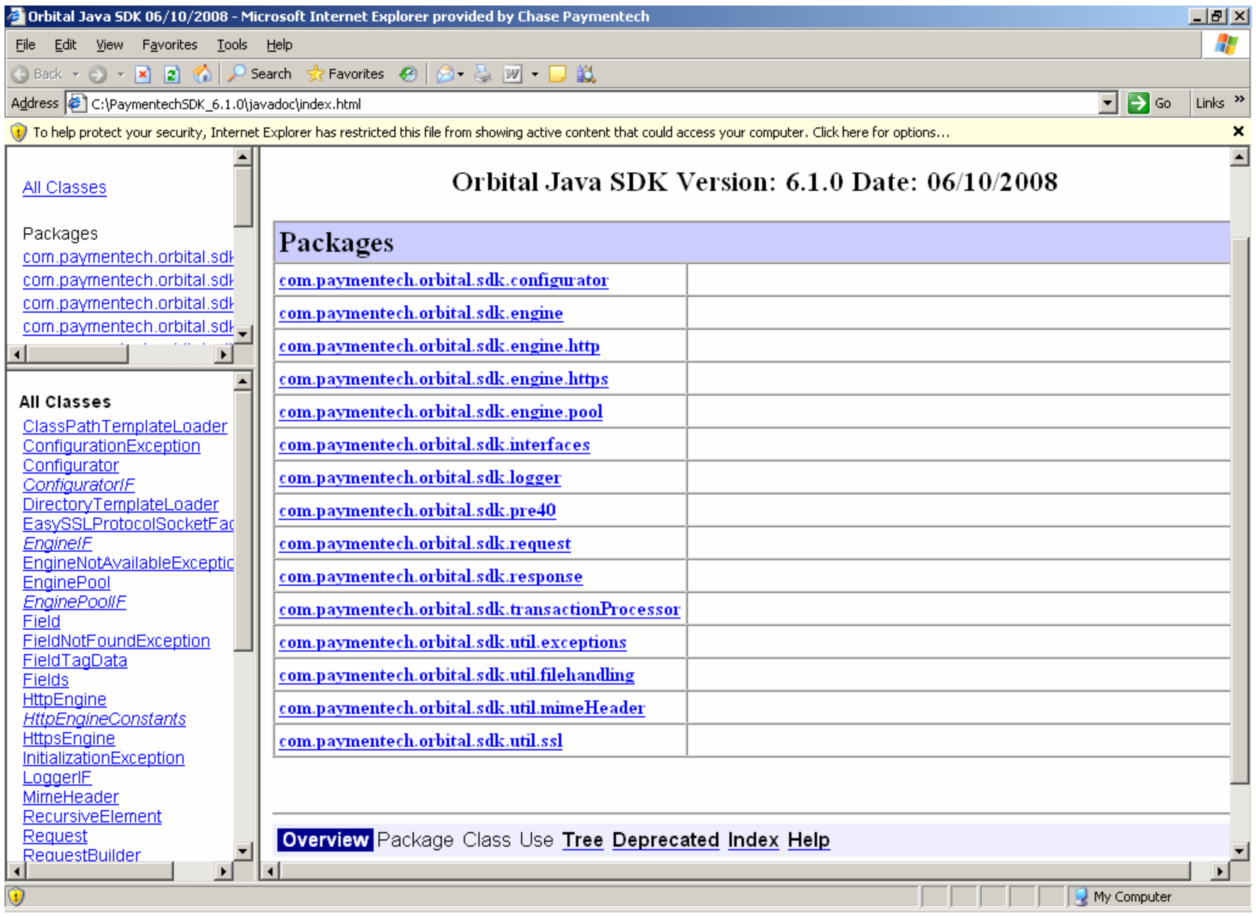
Response Attributes:

```
isGood=true
isError=false
```

```
isQuickResponse=false
isApproved=true
isDeclined=false
AuthCode=090339
TxRefNum=45BDF8F144032EE58042B7081F9D0BE186FB5443
ResponseCode=00
Status=0
Message=Approved
AVSCode=F
```

6 Online Documentation

The Chase Paymentech Java SDK ships with complete javadoc of the Java SDK classes, located in the **%PAYMENTECH_HOME%\javadoc** directory. To access the javadoc, use your browser to open the file %**PAYMENTECH_HOME%\javadoc\index.html.**


Orbital Java SDK 06/10/2008 - Microsoft Internet Explorer provided by Chase Paymentech
File Edit View Favorites Tools Help
Back Search Favorites
Address
Go
Links
My Computer

C:\PaymentechSDK_6.1.0\javadoc\index.html

To help protect your security, Internet Explorer has restricted this file from showing active content that could access your computer. Click here for options...

All Classes

Packages
com.paymentech.orbital.sdk
com.paymentech.orbital.sdk
com.paymentech.orbital.sdk
com.paymentech.orbital.sdk

All Classes
ClassPathTemplateLoader
ConfigurationException
Configurator
ConfiguratorIF
DirectoryTemplateLoader
EasySSLProtocolSocketFac
EngineIF
EngineNotAvailableExceptic
EnginePool
EnginePoolIF
Field
FieldNotFoundException
FieldTagData
Fields
HttpEngine
HttpEngineConstants
HttpsEngine
InitializationException
LoggerIF
MimeHeader
RecursiveElement
Request
RequestBuilder

Orbital Java SDK Version: 6.1.0 Date: 06/10/2008

Packages	
com.paymentech.orbital.sdk.configurator	
com.paymentech.orbital.sdk.engine	
com.paymentech.orbital.sdk.engine.http	
com.paymentech.orbital.sdk.engine.https	
com.paymentech.orbital.sdk.engine.pool	
com.paymentech.orbital.sdk.interfaces	
com.paymentech.orbital.sdk.logger	
com.paymentech.orbital.sdk.pre40	
com.paymentech.orbital.sdk.request	
com.paymentech.orbital.sdk.response	
com.paymentech.orbital.sdk.transactionProcessor	
com.paymentech.orbital.sdk.util.exceptions	
com.paymentech.orbital.sdk.util.filehandling	
com.paymentech.orbital.sdk.util.mimeHeader	
com.paymentech.orbital.sdk.util.ssl	

Overview Package Class Use **Tree** **Deprecated** **Index** **Help**

7 Chase Paymentech Java SDK Developer's Reference

7.1 Overview

The Chase Paymentech Software Developers Kit (SDK) for Java Applications is designed for use by programmers to integrate real-time payment processing into commerce applications.

Users of the SDK are expected to have working knowledge of the Java programming language and object-oriented programming. The SDK is designed to shield the application programmer from having to understand low-level details including sockets, and SSL.

The Java API consists of a singleton class and three Java interfaces:

- Configurator (singleton)
- TransactionProcessorIF
- RequestIF
- ResponseIF
- TemplateIF

Understanding each of these four subsystems is key to successful development.

7.2 Configurator

The Configurator does the following:

1. Loads properties from the Linehandler.properties file
2. Validates and loads the XML template files into fast memory
3. Initializes the loggers (eCommerce and engine loggers)
4. Dumps its contents to the engine log, so developers and support staff can confirm that the correct linehandler.properties file and XML Templates are being loaded.

The Configurator is a singleton class; this means it has a private constructor and can only be accessed via its getInstance method. The getInstance() method guarantees that only one instance of the Configurator exists at the same time, and, therefore, all the subsystems are guaranteed to be using the same configuration properties, XML templates and loggers.

Once the Configurator is created, it becomes available to all SDK subsystems, including the Request, Response and Transaction Processor subsystems as well as the "engines".

7.3 Transaction Processor

The Transaction Processor does the following:

1. Governs access to the Orbital Gateway by establishing a pool of transaction engines. The pool size is specified in the linehandler.properties file (see Section 4.2.1, "TransactionProcessor.poolSize")
2. Requests and releases resources (the application is no longer required to do so)
3. Executes transactions; sends the request and receives the response
4. Performs automatic retries of transactions, if they are assigned a trace number by the application
5. Manages error conditions and recovery, including failover
6. Returns the response to the application, or throws an exception indicating an error condition
7. Logs messages of appropriate log level to the eCommerce log and the engine log.

The application should access a Transaction Processor object using the TransactionProcessorIF.java interface

(TransactionProcessorIF.java is a "published" interface. Method signatures not contained in the published interface are subject to change).

7.4 Request

The Request object is used to create a String representation of the XML payload that will be sent to the Gateway. It does the following:

1. Selects the appropriate XML Template from the Configurator
2. Determines which fields are required vs. optional or not required
3. Validates the field settings
4. If there are field errors, throws exceptions to the application
5. Creates a String representation of the XML for the Transaction Processor
6. Logs messages of appropriate log level to the engine log.
7. Provides access to Complex xml objects (see 7.5 Complex Types)

After an application creates a Configurator, it next creates a Request object and populates its fields. If fields are populated successfully, then the Request object creates a String representation of the XML, which the Transaction Processor retrieves by calling the Request object's getXML() method.

The application should access a Request object using the RequestIF.java interface (RequestIF.java is a "published" interface. Method signatures not contained in the published interface are subject to change).

7.5 Complex Types (TemplateIF Interface)

Complex types allow the Chase Paymentech SDK to dynamically create sections of the XML transactions based off the presence of data. Complex types are basically just children nodes in a XML transaction and are created by calling the "getComplexRoot" method of the Request object. The method returns an object of type TemplateIF, which is used to populate the XML elements (see 7.7 Sample Programs for usage example and 7.8.2.4 Creating Complex Types for a detailed explanation).

As of the 5.0.0 release of the SDK, there are only 2 situations that require the use of complex types, transactions that require the "PriorAuthID" field to be populated and Pcard3 tranasction.

7.6 Response

The Response object accepts the raw XML response from the Orbital Gateway, and provides helper methods that return XML response attributes and field values. It does the following:

1. Receives the XML payload from the Orbital Gateway (an "engine" from the "engine pool" sets the raw XML payload into the Response object).
2. Determines various attributes of the message by parsing the raw XML payload.
3. Makes the attributes available to the application through the use of helper methods ("getters")
4. Makes the raw XML available to the application
5. Logs messages of appropriate log levels to the engine log.

The application should access a Response object using the ResponseIF.java interface (ResponseIF.java is a "published" interface. Method signatures not contained in the published interface are subject to change).

Any response element that is returned in the XML response can be extracted using the response object. The syntax for retrieving an element value is:

```
String value = <response object>.getValue("<name of xml response element>");
```

Example:

```
ResponseIF response = tp.process(request);

// get the TxRefNum returned for a transaction
String txRefNum = response.getValue("TxRefNum");
```

NOTE: The response object's "getValue" method is case-sensitve. The value must be the exact name of the XML element in the response.

7.7 Sample Programs

The Java SDK includes a sample program for every type of transaction, located in the "samples" directory. These programs are standalone java applications that contain their own main() methods. In addition to the java code, the samples directory also contains the required scripts for building and executing the samples.

The sample programs are independent from the rest of the development kit. A linehandler.properties file is included, and the eCommerce and engine logs are output to the samples directory. The only coupling between the sample programs and the rest of the SDK is in their use of the required library jar files (PaymentechSDK.jar, etc.)

For simplicity of understanding the API, test data for the sample programs is included directly within the programs as Strings. To change the test data in the samples, change the values of the Strings and re-compile.

Please note the sample programs are stand-alone programs; as such, each time you run a sample program you incur the entire burden of initialization. In other words, each time you execute a sample program, a new java jvm is initialized, the configurator singleton is loaded, the engine pool singleton is initialized, and an SSL random encryption seed is generated. Therefore, it is not advised that your application send transactions by simply executing sample programs in separate jvm instances. Instead, the code in the sample programs should be used as an example of how to integrate the Java SDK API calls directly into your application code.

The next section includes a complete walkthrough of a sample program.

7.8 Programming Examples

This section is a complete walkthrough of sample programs, including the steps for compiling, building, and an explanation of the API usage. The first example is a "normal" credit card example that will provide the means to get you started and the second example is a more complex example.

7.8.1 Sample Credit Card Program Example

7.8.1.1 Building and Running AuthSample.java Program

- To run the AuthSample.java program, do the following steps:

1. Install the SDK as described in Section 3, "Installation Process"
2. Configure the SDK as described in Section 4, "Configuring the SDK" (Please note that the linehandler.properties file used by the sample programs is located in the samples directory; you should edit that copy to include your specific properties).
3. Edit the AuthSample.java program, substituting your test data in place of the following in the request.setFieldValues() methods invocations shown below.

```
//Basic Auth Fields
request.setFieldValue("IndustryType", "EC");
request.setFieldValue("MessageType", "A");
request.setFieldValue("MerchantID", "700000000413");
request.setFieldValue("BIN", "000002");
request.setFieldValue("OrderID", "122003SA");
request.setFieldValue("AccountNum", "4055011111111111");
request.setFieldValue("Amount", "100");
request.setFieldValue("Exp", "1209");
// AVS Information
request.setFieldValue("AVSname", "Jon Smith");
request.setFieldValue("AVSaddress1", "4200 W Cypress St");
request.setFieldValue("AVScity", "Tampa");
request.setFieldValue("AVSstate", "FL");
request.setFieldValue("AVSzip", "11111");
// Additional Information
request.setFieldValue("Comments", "This is Java SDK v6.3.0");
request.setFieldValue("ShippingRef", "FEDEX WB12345678 Pri 1");
//Uncomment the line below and modify to add a card security value (CVV2, CVC2 or CID)
request.setFieldValue("CardSecVal", "111");
request.setFieldValue("CardSecValInd", "1");
```

4. Use the "build.bat" or "build.sh" script (provided in the samples directory) to compile the AuthSample.java program.

```
build.bat AuthSample.java
```

5. Use the "run.bat" or "run.sh" script (also provided in the samples directory) to run the sample program. Note: If you are using a network proxy, modify and use the "runproxy.bat" or "runproxy.sh" scripts.

```
run.bat AuthSample
```

6. If the transaction is successful, the results should look similar to the results below (Note: %PAYMENTECH_HOME% in this example is set to C:\PaymentechSDK_6.3.0; it could be different in your environment)

```
C:\PaymentechSDK_6.3.0\samples>build AuthSample.java

C:\PaymentechSDK_6.3.0\samples>C:\Progra~1\Java\jdk1.5.0_08\bin\jav
ac -classpath C:\PaymentechSDK_6.3.0/lib/PaymentechSDK.jar AuthSamp
le.java

C:\PaymentechSDK_6.3.0\samples>run AuthSample

C:\PaymentechSDK_6.3.0\samples>C:\Progra~1\Java\jdk1.5.0_08/jre/bin
/java -DPAYMENTECH_HOME=C:\ PaymentechSDK_6.3.0 -classpath .;
C:\PaymentechSDK_6.3.0/lib/PaymentechSDK.jar;C:\ PaymentechSDK
_6.3.0/lib/commons-logging.jar;C:\PaymentechSDK_6.3.0/lib/jsse.jar;
C:\PaymentechSDK_6.3.0/lib/jcert.jar;C:\PaymentechSDK_
6.3.0/lib/jnet.jar;C:\PaymentechSDK_6.3.0/lib/commons-httpclient-2.
0.2.jar;C:\PaymentechSDK_6.3.0/lib/regexp.jar;C:\Payme
ntechSDK_6.3.0/lib/log4j-1.2.8.jar;C:\PaymentechSDK_6.3.0/lib/xml4j
.jar;C:\PaymentechSDK_6.3.0/lib/sunrsasign.jar AuthSample
```

Auth Request:

```
<?xml version="1.0" encoding="UTF-8"?><Request> <NewOrder>          <Industr
yType>EC</IndustryType>        <MessageType>A</MessageType>          <BIN>000
002</BIN>            <MerchantID>700000000413</MerchantID>         <Termina
lID>001</TerminalID>          <CardBrand></CardBrand>        <AccountNum>4055
011111111111</AccountNum>            <Exp>1209</Exp>        <CurrencyCode>84
0</CurrencyCode>            <CurrencyExponent>2</CurrencyExponent>
<CardSecValInd>1</CardSecValInd>            <CardSecVal>111</CardSecVal>
       <DebitCardIssueNum></DebitCardIssueNum>        <DebitCardStartDate></De
bitCardStartDate>            <BCRtNum></BCRtNum>          <CheckDDA></Chec
kDDA>         <BankAccountType></BankAccountType>          <ECPAuthMethod><
/ECPAuthMethod>        <BankPmtDelv></BankPmtDelv>          <AVSzip>11111</A
VSzip>         <AVSaddress1>4200 W Cypress St</AVSaddress1>          <AVSaddr
ess2></AVSaddress2>           <AVScity>Tampa</AVScity>            <AVSstat
e>FL</AVSstate>        <AVSphoneNum></AVSphoneNum>          <AVSname>Jon Smi
th</AVSname>          <AVScountryCode></AVScountryCode>            <AVSDest
zip></AVSDestzip>            <AVSDestaddress1></AVSDestaddress1>
<AVSDestaddress2></AVSDestaddress2>          <AVSDestcity></AVSDestcity>
       <AVSDeststate></AVSDeststate>         <AVSDestphoneNum></AVSDestphoneN
um>           <AVSDestname></AVSDestname>          <AVSDestcountryCode></AV
SDestcountryCode>            <CustomerProfileFromOrderInd>EMPTY</CustomerProf
ileFromOrderInd>            <CustomerRefNum></CustomerRefNum>
<CustomerProfileOrderOverrideInd></CustomerProfileOrderOverrideInd>
<AuthenticationECIInd></AuthenticationECIInd>         <CAVV></CAVV>
<XID></XID>                          <OrderID>122003SA</OrderID>
<Amount>100</Amount>          <Comments>This is Java SDK v6.3.0</Comments>
       <ShippingRef>FEDEX WB12345678 Pri 1</ShippingRef>            <TaxInd>
</TaxInd>            <Tax></Tax>           <AMEXTranAdvAddn1></AMEXTranAdvA
ddn1>          <AMEXTranAdvAddn2></AMEXTranAdvAddn2>          <AMEXTranAdvAddn
3></AMEXTranAdvAddn3>          <AMEXTranAdvAddn4></AMEXTranAdvAddn4>
<AAV></AAV>           <SDMerchantName></SDMerchantName>            <SDProdu
ctDescription></SDProductDescription>          <SDMerchantCity></SDMerchantCity
>            <SDMerchantPhone></SDMerchantPhone>          <SDMerchantURL><
/SDMerchantURL>        <SDMerchantEmail></SDMerchantEmail>          <Recurri
ngInd></RecurringInd>          <EUDDCountryCode></EUDDCountryCode>
<EUDDBankSortCode></EUDDBankSortCode>          <EUDDRibCode></EUDDRibCode>
       <BMLCustomerIP></BMLCustomerIP>        <BMLCustomerEmail></BMLCustomerE
mail>          <BMLShippingCost></BMLShippingCost>          <BMLTNCVersion><
/BMLTNCVersion>        <BMLCustomerRegistrationDate></BMLCustomerRegistrationDa
te>           <BMLCustomerTypeFlag></BMLCustomerTypeFlag>          <BMLItem
Category></BMLItemCategory>          <BMLPreapprovalInvitationNum></BMLPreapp
rovalInvitationNum>           <BMLMerchantPromotionalCode></BMLMerchantPromoti
onalCode>            <BMLCustomerBirthDate></BMLCustomerBirthDate>
<BMLCustomerSSN></BMLCustomerSSN>            <BMLCustomerAnnualIncome></BMLCu
stomerAnnualIncome>           <BMLCustomerResidenceStatus></BMLCustomerResiden
ceStatus>            <BMLCustomerCheckingAccount></BMLCustomerCheckingAccount
>            <BMLCustomerSavingsAccount></BMLCustomerSavingsAccount>
<BMLProductDeliveryType></BMLProductDeliveryType>            <BillerReference
Number></BillerReferenceNumber>        <PCOrderNum></PCOrderNum>
<PCDestZip></PCDestZip>        <PCDestName></PCDestName>            <PCDestA
ddress1></PCDestAddress1>            <PCDestAddress2></PCDestAddress2>
       <PCDestCity></PCDestCity>            <PCDestState></PCDestState>
              </NewOrder></Request>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?><Response><NewOrderResp><IndustryType></In
dustryType><MessageType>A</MessageType><MerchantID>700000000413</MerchantID><
Ter
minalID>001</TerminalID><CardBrand>VI</CardBrand><AccountNum>4055011111111111</
A
ccountNum><OrderID>122003SA</OrderID><TxRefNum>45BDF8F144032EE58042B7081F9
D0BE18
6FB5443</TxRefNum><TxRefIdx>0</TxRefIdx><ProcStatus>0</ProcStatus><ApprovalStatu
s>1</ApprovalStatus><RespCode>00</RespCode><AVSRespCode>F
</AVSRespCode><CVV2Res
pCode>M</CVV2RespCode><AuthCode>090339</AuthCode><RecurringAdviceCd></Recurri
ngA
dviceCd><CAVVRespCode></CAVVRespCode><StatusMsg>Approved</StatusMsg><RespM
sg></R
espMsg><HostRespCode>00</HostRespCode><HostAVSRespCode>A</HostAVSRespCode
><HostC
VV2RespCode>M</HostCVV2RespCode><CustomerRefNum></CustomerRefNum><Custome
rName><
/CustomerName><ProfileProcStatus></ProfileProcStatus><CustomerProfileMessage></C
ustomerProfileMessage><BillerReferenceNumber></BillerReferenceNumber><RespTime>0
83858</RespTime></NewOrderResp></Response>

Response Attributes:
isGood=true
isError=false
isQuickResponse=false
isApproved=true
isDeclined=false
AuthCode=090339
TxRefNum=45BDF8F144032EE58042B7081F9D0BE186FB5443
ResponseCode=00
Status=0
Message=Approved
AVSCode=F
CVV2ResponseCode=M
```

7.8.2 *AuthSample.java Code Walkthrough*

The AuthSample.java program is presented below, in its entirety.

```java
import java.io.File;

import com.paymentech.orbital.sdk.configurator.Configurator;
import com.paymentech.orbital.sdk.configurator.ConfiguratorIF;
import com.paymentech.orbital.sdk.interfaces.RequestIF;
import com.paymentech.orbital.sdk.interfaces.ResponseIF;
import com.paymentech.orbital.sdk.interfaces.TransactionProcessorIF;
import com.paymentech.orbital.sdk.request.FieldNotFoundException;
import com.paymentech.orbital.sdk.request.Request;
import com.paymentech.orbital.sdk.transactionProcessor.TransactionException;
import com.paymentech.orbital.sdk.transactionProcessor.TransactionProcessor;
import com.paymentech.orbital.sdk.util.exceptions.InitializationException;

public class AuthSample {
   //Global Constants
   public final static int NORMAL_EXIT = 1;
   public final static int ERROR_EXIT = -1;
   //This program looks for the configFile in ${PAYMENTECH_HOME}/samples/linehandler.properties
   public final static String configFile = "samples" + File.separator + "linehandler.properties";
   //Global variables
   protected static ConfiguratorIF configurator = null;

   public static void main(String[] args) {
      //Create a configurator
      //The configurator is a "singleton" object that contains system configurations.
      //It loads its properties from the configuration file, that is for historical
      //reasons commonly named "linehandler.properties"
      //The configurator also initializes and contains references to the Paymentech
      //loggers (eCommerceLogger and engineLogger).
      try {
         //PAYMENTECH_HOME must be passed in as a system property via the java "-D" option
         String paymentechHome = System.getProperty("PAYMENTECH_HOME");
         //Create the config file fully qualified path name
         String configFileFullPath = paymentechHome + File.separator + configFile;
         configurator = Configurator.getInstance(configFileFullPath);
      } catch (InitializationException ie) {
         System.err.println("Configurator initialization failed.");
         System.exit(ERROR_EXIT);
      }
      //Create a request object
      //The request object uses the XML templates along with data we provide
      //to gemerate a String representation of the xml request
      RequestIF request = null;
      try {
         //Tell the request object which template to use (see RequestIF.java)
         request = new Request(RequestIF.NEW_ORDER_TRANSACTION);

         //If there were no errors preparing the template, we can now specify the data
         //Basic Auth Fields
         request.setFieldValue("IndustryType", "EC");
         request.setFieldValue("MessageType", "A");
         request.setFieldValue("MerchantID", "700000000413");
         request.setFieldValue("BIN", "000002");
         request.setFieldValue("OrderID", "122003SA");
         request.setFieldValue("AccountNum", "4055011111111111");
         request.setFieldValue("Amount", "100");
         request.setFieldValue("Exp", "1209");
         // AVS Information
         request.setFieldValue("AVSname", "Jon Smith");
         request.setFieldValue("AVSaddress1", "4200 W Cypress St");
         request.setFieldValue("AVScity", "Tampa");
         request.setFieldValue("AVSstate", "FL");
         request.setFieldValue("AVSzip", "11111");
         // Additional Information
```

```
        request.setFieldValue("Comments", "This is Java SDK v6.3.0");
        request.setFieldValue("ShippingRef", "FEDEX WB12345678 Pri 1");
        //Uncomment the line below and modify to add a card security value (CVV2, CVC2 or CID)
        request.setFieldValue("CardSecVal", "111");
        request.setFieldValue("CardSecValInd", "1");

        //Display the request
        System.out.println("\nAuth Request:\n" + request.getXML());
    } catch (InitializationException ie) {
        System.err.println("Unable to initialize request object");
        System.err.println(ie.getMessage());
        ie.printStackTrace();
        System.exit(ERROR_EXIT);
    } catch (FieldNotFoundException fnfe) {
        System.err.println("Unable to find XML field in template");
        System.err.println(fnfe.getMessage());
        fnfe.printStackTrace();
        System.exit(ERROR_EXIT);
    } catch (Exception e) {
                        e.printStackTrace();
                        System.exit(ERROR_EXIT);
            }

    //Create a Transaction Processor
    //The Transaction Processor acquires and releases resources and executes transactions.
    //It configures a pool of protocol engines then uses the pool to execute transactions.
    TransactionProcessorIF tp = null;
    try {
        tp = new TransactionProcessor();
    } catch (InitializationException iex) {
        System.err.println("TransactionProcessor failed to initialize");
        System.err.println(iex.getMessage());
        iex.printStackTrace();
        System.exit(ERROR_EXIT);
    }
    //Process the transaction
    //Pass in the request object (created above), and receive a response object.
    //If the resources required by the Transaction Processor have been exhausted,
    //this code will block until the resources become available.
    //The "TransactionProcessor.poolSize" configuration property specifies how many resources
    //will be available.  The TransactionProcessor acts as a governor, only allowing
    //up to "poolSize" transactions outstanding at any point in time.
    //As transactions are completed their resources are placed back in the pool.
    ResponseIF response = null;
    try {
        response = tp.process(request);
    } catch (TransactionException tex) {
        System.err.println("Transaction failed, including retries and failover");
        System.err.println(tex.getMessage());
        tex.printStackTrace();
        System.exit(ERROR_EXIT);
    }
    //Display the response
```

```
        //This line displays the entire xml response on the java system console.
        System.out.println("\nResponse:\n" + response.toXmlString() + "\n");
        //The lines below report all the various attributes of the response object.
        //It is not necessary to use all of these attributes - use only the ones you need.
        //Also, some of the attributes are meaningful only for specific types of transactions,
        //but for consistency and simplicity in the sample code we dump them all for every transaction
type.
        System.out.println("Response Attributes:");
        System.out.println("isGood=" + response.isGood());
        System.out.println("isError=" + response.isError());
        System.out.println("isQuickResponse=" + response.isQuickResponse());
        System.out.println("isApproved=" + response.isApproved());
        System.out.println("isDeclined=" + response.isDeclined());
        System.out.println("AuthCode=" + response.getAuthCode());
        System.out.println("TxRefNum=" + response.getTxRefNum());
        System.out.println("ResponseCode=" + response.getResponseCode());
        System.out.println("Status=" + response.getStatus());
        System.out.println("Message=" + response.getMessage());
        System.out.println("AVSCode=" + response.getAVSResponseCode());
        System.out.println("CVV2ResponseCode=" + response.getCVV2RespCode());
    }
}
```

7.8.2.1 Import the Required Classes

In order to use the Java SDK API, you must import the classes indicated below. These classes are contained in the PaymentechSDK.jar file, which is included in the java classpath.

For Java SDK 5.x and higher, all API related classes are located in the “com.Paymentech.orbital.sdk” package.

```
import java.io.File;

import com.paymentech.orbital.sdk.configurator.Configurator;
import com.paymentech.orbital.sdk.configurator.ConfiguratorIF;
import com.paymentech.orbital.sdk.interfaces.RequestIF;
import com.paymentech.orbital.sdk.interfaces.ResponseIF;
import com.paymentech.orbital.sdk.interfaces.TransactionProcessorIF;
import com.paymentech.orbital.sdk.request.FieldNotFoundException;
import com.paymentech.orbital.sdk.request.Request;
import com.paymentech.orbital.sdk.transactionProcessor.TransactionException;
import com.paymentech.orbital.sdk.transactionProcessor.TransactionProcessor;
import com.paymentech.orbital.sdk.util.exceptions.InitializationException;
```

7.8.2.2 Create the Configurator

The Configurator is a singleton class that contains global configurations, the XML Templates and the eCommerce and engine loggers.

The Configurator loads the properties from the linehandler.properties file, then loads the XML Templates and then initializes the loggers. It then dumps its contents into the engine log, so the configuration can be confirmed by visually examining the log.

A benefit of the Configurator architecture is that all required file I/O is performed up-front during initialization. Other than logging, the Java SDK performs no file I/O after the Configurator is initialized. Another benefit is that all global configurations and loggers are included in a single object that is accessible by all subsystems of the Java SDK.

Since the Configurator is a singleton class, its constructor is private. You create a Configurator by calling its getInstance() method, passing in the full path specification of your configuration file. For historical and support purposes, the configuration file should always be named “linehandler.properties”.

```
    try {
   //PAYMENTECH_HOME must be passed in as a system property via the java "-D" option
   String paymentechHome = System.getProperty("PAYMENTECH_HOME");
   //Create the config file fully qualified path name
   String configFileFullPath = paymentechHome + File.separator + configFile;
   configurator = Configurator.getInstance(configFileFullPath);
} catch (InitializationException ie) {
   System.err.println("Configurator initialization failed.");
   System.exit(ERROR_EXIT);
}
```

7.8.2.3 Create the Request

A Request object is always declared as a RequestIF type. RequestIF is the supported “published interface”, and contains all supported method signatures. (Only the method signatures contained in the RequestIF interface are guaranteed not to change in future releases.)

After declaring the request object you instantiate it by invoking its constructor, passing in the desired transaction type. Transaction types are included in the RequestIF interface definition (do not use deprecated constants – these only provide backward compatibility with 4.x versions). The request object uses the transaction type to locate the XML Template that will be used for the field substitutions and validation.

After instantiating the request object, you set the data fields by invoking the request.setFieldValue() method, passing in the XML field element name, and the value, as indicated in the example. If the field cannot be found in either the required or optional fields, then a FieldNotFoundException will be thrown.

To enable the automatic retry of a transaction, all you must do is set a Trace Number into the request using the setTraceNumber() method. The Trace Number must be a numeric value (a String representation of an integer).

Two types of exceptions may occur when creating a request object. Your application must catch and appropriately handle these exception types.

- An InitializationException() will be thrown if there is a problem initializing the request object (for instance, if there is a problem with the Configurator). It is important to note that if the appropriate template can not be found an IntializationException will be thrown with the message content indicating which template could not be found.

- A FieldNotFoundException() is thrown if an invalid XML field element name is passed into the setFieldValue() method.

```
RequestIF request = null;
```

```
try {
    //Tell the request object which template to use (see RequestIF.java)
    // If the template can not be an IntializationException will be thrown
    request = new Request(RequestIF.NEW_ORDER_TEMPLATE);

    // use the request object's setFieldValue to fill the XML data
    // this method will throw a FieldNotFound exception if the named field
    // can't be found in the transaction's template file
    request.setFieldValue ("MerchantID", "041756");

} catch (InitializationException ie) {
    System.err.println("Unable to initialize request object");
    System.err.println(ie.getMessage());
    ie.printStackTrace();
    System.exit(ERROR_EXIT);
} catch (FieldNotFoundException fnfe) {
    System.err.println("Unable to find XML field in template");
    System.err.println(fnfe.getMessage());
    fnfe.printStackTrace();
    System.exit(ERROR_EXIT);
} catch (Exception ex) {
    ex.printStackTrace();
    System.exit(ERROR_EXIT);
 }
```

7.8.2.4 Create a Transaction Processor

The Transaction Processor shields the application developer from all the details of resource management, governing, communication protocol, and error handling, including retries and failover.

Each time you execute a transaction you create a new Transaction Processor. The Transaction Processor is thread-safe; it is intended that multiple application threads will be concurrently executing Transaction Processors.

A Transaction Processor object is always declared as a TransactionProcessorIF type. TransactionProcessorIF is the supported "published interface", and contains all supported method signatures. (Only the method signatures contained in the TransactionProcessorIF interface are guaranteed not to change in future releases.)

The application must catch and handle one exception – the InitializationException(), which occurs if there is any error while initializing the Transaction Processor. This error would most likely be the result of a configuration error.

```
//Create a Transaction Processor
//The Transaction Processor acquires and releases resources and executes transactions.
//It configures a pool of protocol engines, then uses the pool to execute transactions.
TransactionProcessorIF tp = null;
try {
    tp = new TransactionProcessor();
} catch (InitializationException iex) {
    System.err.println("TransactionProcessor failed to initialize");
    System.err.println(iex.getMessage());
    iex.printStackTrace();
    System.exit(ERROR_EXIT);
}
```

7.8.2.5 Process the Transaction

After creating the Configurator, the Request and the Transaction Processor, sending the request is simple. Just invoke the process() method, passing in the request object; the Transaction Processor takes care of everything else, and returns a response.

The application should always declare the response object returned by the Transaction Processor to be of type ResponseIF (this is the “published” interface for the response object).

In the event that the Transaction Processor is unable to complete the Transaction (usually due to catastrophic network failure), it throws a TransactionException(). This exception is thrown only after all retry attempts both in normal and failover mode have failed. The application should always handle a TransactionException, since it indicates the Transaction did not complete.

```
//Process the transaction
//Pass in the request object (created above), and receive a response object.
//If the resources required by the Transaction Processor have been exhausted,
//this code will block until the resources become available.
//The "TransactionProcessor.poolSize" configuration property specifies how many resources
//will be available.  The TransactionProcessor acts as a governor, only allowing
//up to "poolSize" transactions outstanding at any point in time.
//As transactions are completed, their resources are placed back in the pool.
ResponseIF response = null;
try {
    response = tp.process(request);
} catch (TransactionException tex) {
    System.err.println("Transaction failed, including retries and failover");
    System.err.println(tex.getMessage());
    tex.printStackTrace();
    System.exit(ERROR_EXIT);
}
```

7.8.2.6 Interpret the Response

The ResponseIF provides full access to the raw XML content of the message, and additionally provides “helper methods” that indicate various message attributes, as illustrated below.

Remember it is against regulations to save real credit card security values (such as CVV2). If you must save responses, ResponseIF provides a method signature, toMaskedXmlString(), that will return the XML with those fields masked. The eCommerce logger and engine logger both use toMaskedXmlString().

```
//Display the response
//This line displays the entire xml response on the java system console.
System.out.println("\nResponse:\n" + response.toXmlString() + "\n");
//The lines below report all the various attributes of the response object.
//It is not necessary to use all of these attributes - use only the ones you need.
//Also, some of the attributes are meaningful only for specific types of transactions,
//but for consistency and simplicity in the sample code we dump them all for every transaction
type.
System.out.println("Response Attributes:");
System.out.println("isGood=" + response.isGood());
System.out.println("isError=" + response.isError());
System.out.println("isQuickResponse=" + response.isQuickResponse());
System.out.println("isApproved=" + response.isApproved());
```

```
        System.out.println("isDeclined=" + response.isDeclined());
        System.out.println("AuthCode=" + response.getAuthCode());
        System.out.println("TxRefNum=" + response.getTxRefNum());
        System.out.println("ResponseCode=" + response.getResponseCode());
        System.out.println("Status=" + response.getStatus());
        System.out.println("Message=" + response.getMessage());
        System.out.println("AVSCode=" + response.getAVSResponseCode());
        System.out.println("CVV2ResponseCode=" + response.getCVV2RespCode());
    }
```

7.8.3 Building and Running the more complex PC3Sample.java Program

1. To run the PC3Sample.java program, do the following steps:

Install the SDK as described in Section 0, "The Orbital Gateway supports two methods of authenticating incoming requests: Source IP authentication and Connection Username/Password authentication. What this means from a client implementation is as follows:

- For IP-based authentication, when processing transactions against both the Test and Production Orbital Gateway, the client server's Source IPs must be registered on the Orbital Gateway.
- For Connection Username/Password authentication, the Username and Password is passed in the message payload. Similar to IP-based authentication, they must match what is set up on the Orbital Gateway in order to process transactions against the Test and Production Orbital Gateway.
- Any activity presented on an IP address or Connection Username/Password that is not registered in the Orbital Gateway will result in an HTTP 412 error with the accompanying XML payload containing a ProcStatus 20412 error.
- In addition, these IP addresses/Connection Usernames must be affiliated with the Merchant IDs for which the client should be submitting transactions, specifically:
    - This does allow Third-Party Hosting service organizations presenting on behalf of other merchants to submit transactions. However, each time a new customer is added, the merchant or third-party hosting organization must ensure that the new Merchant IDs or Chain IDs are affiliated with the hosting company's IPs or Connection Usernames.
    - If the merchant expects to have more than one merchant account with the Orbital Gateway, it should have its IP addresses/Connection Usernames affiliated at the Chain-level hierarchy within the Orbital Gateway.

      Each time a new Merchant ID is added, as long as it is placed within the same Chain, it will simply work. If it is not placed within the same Chain, the additional MIDs must be affiliated with the merchant IPs or Connection Usernames. For example, we generally affiliate all Salem accounts (BIN 000001) with their Company Number (formerly called *MA #*), so all MIDs or Divisions under that Company will automatically be affiliated.
- MID-Association Failures
    - If an IP is registered, but the client presents a MID that has NOT been associated with the originating IP, the Orbital Gateway will return a HTTP 200 error with a ProcStatus of 9717.
    - If a Connection Username is registered, but the client presents a MID that has NOT been associated with the Username, the Orbital Gateway will return a ProcStatus 20412.

7.8.4 Connection Username/Password Format

The Orbital Connection Username and Password are submitted within the message payload as two separate elements:

- <OrbitalConnectionUsername>

- <OrbitalConnectionPassword>

Similar to Source IP authentication setup, the Connection Username and Password must be set up on the Orbital Gateway. Please contact your Technical Analyst or Account Representative for more information about getting set up.

The Connection Username and Password must follow specific formatting rules. Both Username and Password:

- Must be between 8–32 characters.
- Must contain at least 1 number.
- Must contain only standard English letters or digits (a-z, A-Z, 0-9).
- Cannot contain embedded spaces.

Additionally, the Connection Password is case-sensitive, while the Connection Username is not.

Note: IP-based authentication and Connection Username/Password authentication are exclusive of each other. If a merchant is set up for both IP-based authentication and Connection Username/Password authentication, request messages will be authenticated based on whether the Connection Username and/or Connection Password elements exist within the payload.

If either element does exist, the Orbital Gateway will attempt to validate the Username/Password values. If the authentication fails (for example, due to an invalid Password), the Orbital Gateway will NOT revert to IP-based authentication

a. Installation Process”
b. Configure the SDK as described in Section 4, “Configuring the SDK” (Please note that the linehandler.properties file used by the sample programs is located in the samples directory; you should edit that copy to include your specific properties).
c. Edit the PC3Sample.java program, substituting your test data in place of the following in the request.setFieldValues() methods invocations shown below.

```
//Basic Auth Fields
request.setFieldValue("IndustryType", "EC");
request.setFieldValue("MessageType", "A");
request. setFieldValue ("MerchantID", "700000000417");
request. setFieldValue ("BIN", "000002");
request. setFieldValue ("OrderID", "122003SA");
request. setFieldValue ("AccountNum", "5191409037560100");
request. setFieldValue ("Amount", "100");
request. setFieldValue ("Exp", "1205");

// AVS Information
request. setFieldValue ("AVSname", "Sam Ayers");
request. setFieldValue ("AVSaddress1", "4200 W Cypress St");
request. setFieldValue ("AVScity", "Tampa");
request. setFieldValue ("AVSstate", "FL");
request. setFieldValue ("AVSzip", "33607");

// Additional Information
request. setFieldValue ("Comments", "This is Java SDK v6.0");
request. setFieldValue ("ShippingRef", "FEDEX WB12345678 Pri 1");

// add data to the top level PC3 element
pc3Root.setFieldValue("PC3FreightAmt", "10");
pc3Root.setFieldValue("PC3DutyAmt", "10");
```

```
pc3Root.setFieldValue("PC3DestCountryCd", "USA");
pc3Root.setFieldValue("PC3ShipFromZip", "34667");
pc3Root.setFieldValue("PC3DiscAmt", "5");
pc3Root.setFieldValue("PC3VATtaxAmt", "5");
pc3Root.setFieldValue("PC3VATtaxRate", "2");
pc3Root.setFieldValue("PC3AltTaxInd", "1");
pc3Root.setFieldValue("PC3AltTaxAmt", "5");


 // fill the line item with data
 lineItem.setFieldValue("PC3DtlDesc", "1234567890123456789");
 lineItem.setFieldValue("PC3DtlProdCd", "123456789");
 lineItem.setFieldValue("PC3DtlQty", "1");
 lineItem.setFieldValue("PC3DtlUOM", "LBR");
 lineItem.setFieldValue("PC3DtlTaxAmt", "0");
 lineItem.setFieldValue("PC3DtlTaxRate", "0");
 lineItem.setFieldValue("PC3Dtllinetot", "50");
 lineItem.setFieldValue("PC3DtlDisc", "0");
 lineItem.setFieldValue("PC3DtlCommCd", "3");
 lineItem.setFieldValue("PC3DtlUnitCost", "5");
 lineItem.setFieldValue("PC3DtlGrossNet", "Y");
 lineItem.setFieldValue("PC3DtlTaxType", "Y");
 lineItem.setFieldValue("PC3DtlDiscInd", "Y");
 lineItem.setFieldValue("PC3DtlDebitInd", "D");
```

d. Use the “build.bat” or “build.sh” script (provided in the samples directory) to compile the PC3Sample.java program.

```
build.bat PC3Sample.java
```

e. Use the “run.bat” or “run.sh” script (also provided in the samples directory) to run the sample program. Note: If you are using a network proxy, modify and use the “runproxy.bat” or “runproxy.sh” scripts.

```
•   run.bat PC3Sample
```

f. If the transaction is successful, the results should look similar to the results below (Note: %PAYMENTECH_HOME% in this example is set to c:\dev\gateway\active\OrbitalSDK; it will be different in your environment)

```
C:\dev\gateway\active\OrbitalSDK\samples>build PC3Sample.java

C:\dev\gateway\active\OrbitalSDK\samples>c:\dev\jdk1.3.1_03\bin\javac -classpath
 c:\dev\gateway\active\OrbitalSDK/lib/PaymentechSDK.jar PC3Sample.java
```

```
C:\dev\gateway\active\OrbitalSDK\samples>run PC3Sample

C:\dev\gateway\active\OrbitalSDK\samples>c:\dev\jdk1.3.1_03/jre/bin/java -DPAYME
NTECH_HOME=c:\dev\gateway\active\OrbitalSDK -classpath .;..;c:\dev\gateway\activ
e\OrbitalSDK/lib/PaymentechSDK.jar;c:\dev\gateway\active\OrbitalSDK/lib/jsse.jar
;c:\dev\gateway\active\OrbitalSDK/lib/jcert.jar;c:\dev\gateway\active\OrbitalSDK
/lib/jnet.jar;c:\dev\gateway\active\OrbitalSDK/lib/xml4j.jar;c:\dev\gateway\acti
ve\OrbitalSDK/lib/httpClient.jar;c:\dev\gateway\active\OrbitalSDK/lib/regexp.jar
;c:\dev\gateway\active\OrbitalSDK/lib/log4j-1.2.8.jar PC3Sample

PC3 Request:

Response:

Response Attributes:
isGood=true
isError=false
isQuickResponse=false
isApproved=true
isDeclined=false
AuthCode=194784
TxRefNum=40180EDB8CC20623DB19435C53190483F25B4F47
ResponseCode=00
Status=0
Message=Approved
AVSCode=F
CVV2ResponseCode=

C:\dev\gateway\active\OrbitalSDK\samples>
```

7.8.5 PC3Sample.java Code Walkthrough

The PC3Sample.java program is presented below, in its entirety.

```
import com.paymentech.orbital.sdk.configurator.Configurator;
import com.paymentech.orbital.sdk.configurator.ConfiguratorIF;
import com.paymentech.orbital.sdk.interfaces.RequestIF;
import com.paymentech.orbital.sdk.interfaces.ResponseIF;
import com.paymentech.orbital.sdk.interfaces.TemplateIF;
import com.paymentech.orbital.sdk.interfaces.TransactionProcessorIF;
import com.paymentech.orbital.sdk.request.FieldNotFoundException;
import com.paymentech.orbital.sdk.request.Request;
import com.paymentech.orbital.sdk.transactionProcessor.TransactionException;
import com.paymentech.orbital.sdk.transactionProcessor.TransactionProcessor;
import com.paymentech.orbital.sdk.util.exceptions.InitializationException;;

public class PC3Sample
{
  //Global Constants
   public final static int NORMAL_EXIT = 1;
   public final static int ERROR_EXIT = -1;
   //This program looks for the configFile in ${PAYMENTECH_HOME}/samples/linehandler.properties
   public final static String configFile = "samples/linehandler.properties";
```

```
//Global variables
protected static ConfiguratorIF configurator = null;

public static void main(String[] args) {
   //Create a configurator
   //The configurator is a "singleton" object that contains system configurations.
   //It loads its properties from the configuration file, that is for historical
   //reasons commonly named "linehandler.properties"
   //The configurator also initializes and contains references to the Chase Paymentech
   //loggers (eCommerceLogger and engineLogger).
   try {
      //PAYMENTECH_HOME must be passed in as a system property via the java "-D" option
      String paymentechHome = System.getProperty("PAYMENTECH_HOME");
      //Create the config file fully qualified path name
      String configFileFullPath = paymentechHome + "/" + configFile;
      configurator = Configurator.getInstance(configFileFullPath);
   } catch (InitializationException ie) {
      System.err.println("Configurator initialization failed.");
      System.exit(ERROR_EXIT);
   }
```

```
// Create Request object
//The request object uses the XML templates along with data we provide
//to gemerate a String representation of the xml request
RequestIF request = null;

try {
   //Tell the request object which template to use (see RequestIF.java)
   request = new Request (RequestIF.NEW_ORDER_TRANSACTION);

   //Basic Information
   request.setFieldValue("IndustryType", "EC");
   request.setFieldValue("MessageType", "AC");
   request.setFieldValue("MerchantID", "700000000413");
   request.setFieldValue("BIN", "000002");
   request.setFieldValue("OrderID", "122003SA");
   request.setFieldValue("AccountNum", "5191409037560100");
   request.setFieldValue("Amount", "100");
   request.setFieldValue("Exp", "1205");

   //AVS Information
   request.setFieldValue("AVSname", "Jon Smith");
   request.setFieldValue("AVSaddress1", "4200 W Cypress St");
   request.setFieldValue("AVScity", "Tampa");
   request.setFieldValue("AVSstate", "FL");
   request.setFieldValue("AVSzip", "33607");

   //Common Optional
   request.setFieldValue("Comments", "This is Java SDK v4.0");
   request.setFieldValue("ShippingRef", "FEDEX WB12345678 Pri 1");

   // add the PC3 data
   TemplateIF pc3Root = request.getComplexRoot(RequestIF.PC3_CORE);
```

```
        // add data to the top level PC3 element
        pc3Root.setFieldValue("PC3FreightAmt", "100");
        pc3Root.setFieldValue("PC3DutyAmt", "100");
        pc3Root.setFieldValue("PC3DestCountryCd", "USA");
        pc3Root.setFieldValue("PC3ShipFromZip", "34667");
        pc3Root.setFieldValue("PC3DiscAmt", "100");
        pc3Root.setFieldValue("PC3VATtaxAmt", "100");
        pc3Root.setFieldValue("PC3VATtaxRate", "2");
        pc3Root.setFieldValue("PC3AltTaxInd", "1");
        pc3Root.setFieldValue("PC3AltTaxAmt", "100");

        // add 5 line items (recursive elements)
        TemplateIF lineItem = null;

        for (int i = 0; i < 5; i++) {
            lineItem = pc3Root.getRecursiveElement(RequestIF.PC3_LINE_ITEMS);

            // fill the line item with data
            lineItem.setFieldValue("PC3DtlDesc", "1234567890123456789");
            lineItem.setFieldValue("PC3DtlProdCd", "123456789");
            lineItem.setFieldValue("PC3DtlQty", "1");
            lineItem.setFieldValue("PC3DtlUOM", "LBR");
            lineItem.setFieldValue("PC3DtlTaxAmt", "0");
            lineItem.setFieldValue("PC3DtlTaxRate", "0");
            lineItem.setFieldValue("PC3Dtllinetot", "100");
            lineItem.setFieldValue("PC3DtlDisc", "100");
            lineItem.setFieldValue("PC3DtlCommCd", "3");
            lineItem.setFieldValue("PC3DtlUnitCost", "100");
            lineItem.setFieldValue("PC3DtlGrossNet", "Y");
            lineItem.setFieldValue("PC3DtlTaxType", "Y");
            lineItem.setFieldValue("PC3DtlDiscInd", "Y");
            lineItem.setFieldValue("PC3DtlDebitInd", "D");
        }

        //Display the request
        System.out.println("\nPC3 Request:\n" + request.getXML());

    } catch (InitializationException ie) {
        System.err.println("Unable to initialize request object");
        System.err.println(ie.getMessage());
        ie.printStackTrace();
        System.exit(ERROR_EXIT);
    } catch (FieldNotFoundException fnfe) {
        System.err.println("Unable to find XML field in template");
        System.err.println(fnfe.getMessage());
        fnfe.printStackTrace();
        System.exit(ERROR_EXIT);
    } catch (Exception e) {
        e.printStackTrace();
        System.exit(ERROR_EXIT);
    }
    //Create a Transaction Processor
    //The Transaction Processor acquires and releases resources and executes transactions.
```

```
        //It configures a pool of protocol engines, then uses the pool to execute transactions.
        TransactionProcessorIF tp = null;
        try {
           tp = new TransactionProcessor();
        } catch (InitializationException iex) {
           System.err.println("TransactionProcessor failed to initialize");
           System.err.println(iex.getMessage());
           iex.printStackTrace();
           System.exit(ERROR_EXIT);
        }
        //Process the transaction
        //Pass in the request object (created above), and receive a response object.
        //If the resources required by the Transaction Processor have been exhausted,
        //this code will block until the resources become available.
        //The "TransactionProcessor.poolSize" configuration property specifies how many resources
        //will be available.  The TransactionProcessor acts as a governor, only allowing
        //up to "poolSize" transactions outstanding at any point in time.
        //As transactions are completed, their resources are placed back in the pool.

        ResponseIF response = null;
        try {
           response = tp.process(request);
        } catch (TransactionException tex) {
           System.err.println("Transaction failed, including retries and failover");
           System.err.println(tex.getMessage());
           tex.printStackTrace();
           System.exit(ERROR_EXIT);
        }

        //Display the response
        //This line displays the entire xml response on the java system console.
        System.out.println("\nResponse:\n" + response.toXmlString() + "\n");
        //The lines below report all the various attributes of the response object.
        //It is not necessary to use all of these attributes - use only the ones you need.
        //Also, some of the attributes are meaningful only for specific types of transactions,
        //but for consistency and simplicity in the sample code we dump them all for every transaction type.
        System.out.println("Response Attributes:");
        System.out.println("isGood=" + response.isGood());
        System.out.println("isError=" + response.isError());
        System.out.println("isQuickResponse=" + response.isQuickResponse());
        System.out.println("isApproved=" + response.isApproved());
        System.out.println("isDeclined=" + response.isDeclined());
        System.out.println("AuthCode=" + response.getAuthCode());
        System.out.println("TxRefNum=" + response.getTxRefNum());
        System.out.println("ResponseCode=" + response.getResponseCode());
        System.out.println("Status=" + response.getStatus());
        System.out.println("Message=" + response.getMessage());
        System.out.println("AVSCode=" + response.getAVSResponseCode());
        System.out.println("CVV2ResponseCode=" + response.getCVV2RespCode());
    }
}
```

7.8.5.1 Import the Required Classes

In order to use the Java SDK API, you must import the classes indicated below. These classes are contained in the PaymentechSDK.jar file, which is included in the java classpath.

For Java SDK 5.x and higher, all API related classes are located in the "com.Paymentech.orbital.sdk" package.

```
import com.paymentech.orbital.sdk.configurator.Configurator;
import com.paymentech.orbital.sdk.configurator.ConfiguratorIF;
import com.paymentech.orbital.sdk.interfaces.RequestIF;
import com.paymentech.orbital.sdk.interfaces.ResponseIF;
import com.paymentech.orbital.sdk.interfaces.TransactionProcessorIF;
import com.paymentech.orbital.sdk.request.FieldNotFoundException;
import com.paymentech.orbital.sdk.request.Request;
import com.paymentech.orbital.sdk.request.XMLTemplateNotFoundException;
import com.paymentech.orbital.sdk.transactionProcessor.TransactionException;
import com.paymentech.orbital.sdk.transactionProcessor.TransactionProcessor;
import com.paymentech.orbital.sdk.util.exceptions.InitializationException;
```

7.8.5.2 Create the Configurator

The Configurator is a singleton class that contains global configurations, the XML Templates and the eCommerce and engine loggers.

The Configurator loads the properties from the linehandler.properties file, then loads the XML Templates and then initializes the loggers. It then dumps its contents into the engine log, so the configuration can be confirmed by visually examining the log.

A benefit of the Configurator architecture is that all required file I/O is performed up-front during initialization. Other than logging, the Java SDK performs no file I/O after the Configurator is initialized. Another benefit is that all global configurations and loggers are included in a single object that is accessible by all subsystems of the Java SDK.

Since the Configurator is a singleton class, its constructor is private. You create a Configurator by calling its getInstance() method, passing in the full path specification of your configuration file. For historical and support purposes, the configuration file should always be named "linehandler.properties".

```
try {
   //PAYMENTECH_HOME must be passed in as a system property via the java "-D" option
   String paymentechHome = System.getProperty("PAYMENTECH_HOME");
   //Create the config file fully qualified path name
   String configFileFullPath = paymentechHome + "/" + configFile;
   configurator = Configurator.getInstance(configFileFullPath);
} catch (InitializationException ie) {
   System.err.println("Configurator initialization failed.");
   System.exit(ERROR_EXIT);
}
```

7.8.5.3 Create the Request

A Request object is always declared as a RequestIF type. RequestIF is the supported “published interface”, and contains all supported method signatures. (Only the method signatures contained in the RequestIF interface are guaranteed not to change in future releases.)

After declaring the request object, you instantiate it by invoking its constructor, passing in the desired transaction type. Transaction types are included in the RequestIF interface definition (do not use deprecated constants – these only provide backward compatibility with 4.x versions). The request object uses the transaction type to locate the XML Template that will be used for the field substitutions and validation.

After instantiating the request object, you set the data fields by invoking the request.setFieldValue() method, passing in the XML field element name, and the value, as indicated in the example. If the field cannot be found in either the required or optional fields, then a FieldNotFoundException will be thrown.

To enable the automatic retry of a transaction, all you must do is set a Trace Number into the request using the setTraceNumber() method. The Trace Number must be a numeric value (a String representation of an integer).

Two types of exceptions may occur when creating a request object. Your application must catch and appropriately handle these exception types.

1. An InitializationException() will be thrown if there is a problem initializing the request object (for instance, if there is a problem with the Configurator). It is important to note that if the appropriate template can not be found an IntializationException will be thrown with the message content indicating which template could not be found.

2. A FieldNotFoundException() is thrown if an invalid XML field element name is passed into the setFieldValue() method.

```
RequestIF request = null;
    try {
       //Tell the request object which template to use (see RequestIF.java)
       // If the template can not be an IntializationException will be thrown
       request = new Request(RequestIF.NEW_ORDER_TEMPLATE);

       // use the request object’s setFieldValue to fill the XML data
       // this method will throw a FieldNotFound exception if the named field
       // can’t be found in the transaction’s template file
       request.setFieldValue (“MerchantID”, “041756”);

    } catch (InitializationException ie) {
       System.err.println("Unable to initialize request object");
       System.err.println(ie.getMessage());
       ie.printStackTrace();
       System.exit(ERROR_EXIT);
    } catch (FieldNotFoundException fnfe) {
       System.err.println("Unable to find XML field in template");
       System.err.println(fnfe.getMessage());
       fnfe.printStackTrace();
       System.exit(ERROR_EXIT);
    } catch (Exception ex) {
       ex.printStackTrace();
    System.exit(ERROR_EXIT);
  }
```

7.8.5.4 Create a Transaction Processor

The Transaction Processor shields the application developer from all the details of resource management, governing, communication protocol, and error handling, including retries and failover.

Each time you execute a transaction you create a new Transaction Processor. The Transaction Processor is thread-safe; it is intended that multiple application threads will be concurrently executing Transaction Processors.

A Transaction Processor object is always declared as a TransactionProcessorIF type. TransactionProcessorIF is the supported "published interface", and contains all supported method signatures. (Only the method signatures contained in the TransactionProcessorIF interface are guaranteed not to change in future releases.)

The application must catch and handle one exception – the InitializationException(), which occurs if there is any error while initializing the Transaction Processor. This error would most likely be the result of a configuration error.

```
//Create a Transaction Processor
//The Transaction Processor acquires and releases resources and executes transactions.
//It configures a pool of protocol engines, then uses the pool to execute transactions.
TransactionProcessorIF tp = null;
try {
    tp = new TransactionProcessor();
} catch (InitializationException iex) {
    System.err.println("TransactionProcessor failed to initialize");
    System.err.println(iex.getMessage());
    iex.printStackTrace();
    System.exit(ERROR_EXIT);
}
```

7.8.5.5 Process the Transaction

After creating the Configurator, the Request and the Transaction Processor, sending the request is simple. Just invoke the process() method, passing in the request object; the Transaction Processor takes care of everything else, and returns a response.

The application should always declare the response object returned by the Transaction Processor to be of type ResponseIF (this is the "published" interface for the response object).

In the event that the Transaction Processor is unable to complete the Transaction (usually due to catastrophic network failure), it throws a TransactionException(). This exception is thrown only after all retry attempts both in normal and failover mode have failed. The application should always handle a TransactionException, since it indicates the Transaction did not complete.

```
//Process the transaction
//Pass in the request object (created above), and receive a response object.
//If the resources required by the Transaction Processor have been exhausted,
//this code will block until the resources become available.
//The "TransactionProcessor.poolSize" configuration property specifies how many resources
//will be available.  The TransactionProcessor acts as a governor, only allowing
//up to "poolSize" transactions outstanding at any point in time.
//As transactions are completed, their resources are placed back in the pool.
ResponseIF response = null;
try {
   response = tp.process(request);
} catch (TransactionException tex) {
   System.err.println("Transaction failed, including retries and failover");
   System.err.println(tex.getMessage());
   tex.printStackTrace();
   System.exit(ERROR_EXIT);
}
```

7.8.5.6 Interpret the Response

The ResponseIF provides full access to the raw XML content of the message, and additionally provides "helper methods" that indicate various message attributes, as illustrated below.

Remember it is against regulations to save real card security values (such as CVV2). If you must save responses, ResponseIF provides a method signature, toMaskedXmlString(), that will return the XML with those fields masked. The eCommerce logger and engine logger both use toMaskedXmlString().

```
    //Display the response
    //This line displays the entire xml response on the java system console.
    System.out.println("\nResponse:\n" + response.toXmlString() + "\n");
    //The lines below report all the various attributes of the response object.
    //It is not necessary to use all of these attributes - use only the ones you need.
    //Also, some of the attributes are meaningful only for specific types of transactions,
    //but for consistency and simplicity in the sample code we dump them all for every transaction
type.
    System.out.println("Response Attributes:");
    System.out.println("isGood=" + response.isGood());
    System.out.println("isError=" + response.isError());
    System.out.println("isQuickResponse=" + response.isQuickResponse());
    System.out.println("isApproved=" + response.isApproved());
    System.out.println("isDeclined=" + response.isDeclined());
    System.out.println("AuthCode=" + response.getAuthCode());
    System.out.println("TxRefNum=" + response.getTxRefNum());
    System.out.println("ResponseCode=" + response.getResponseCode());
    System.out.println("Status=" + response.getStatus());
    System.out.println("Message=" + response.getMessage());
    System.out.println("AVSCode=" + response.getAVSResponseCode());
    System.out.println("CVV2ResponseCode=" + response.getCVV2RespCode());
  }
```

8 Building the PaymentechSDK.jar

This section of the Java SDK Developer's Guide is intended for advanced SDK users that may want to customize components in the PaymentechSDK.jar file. The developer must have an advanced understanding of Java application development.

Note: Chase Paymentech will not support builds produced and installed from this process.

8.1 Prerequisites

Two software tools are required to build the Chase Paymentech Java SDK – the Sun Java Development Kit (JDK) and Apache Ant.

A Java Development Kit (JDK) must be installed in the development environment, and the JAVA_HOME environment variable must be set to point to the JDK installation directory. The java compilers provided in JDK Version 1.4.x or 1.5.x are recommended, and supported, by Chase Paymentech. The Sun JDK is available for free download at the following URL:

http://java.sun.com

Apache Ant must also be installed, and the ANT_HOME environment variable must be set to the ant installation directory. Ant version 1.5.x or higher is required. Ant is available for free download at the following URL:

http://ant.apache.org/

8.2 Configuring the Build Environment

- Set the **JAVA_HOME** and **ANT_HOME** environment variables to point to their respective installation directories.
- Set the **PAYMENTECH_HOME** environment variable to point to the PaymentechSDK installation directory.
- In a Windows environment, remove the $PAYMENTECH_HOME/build directory if it exists
- For a Windows environment, test the environment variable settings by executing the following commands and observing the results:
    - %ANT_HOME%/bin/ant –version (should display the ant version)
    - %JAVA_HOME%/bin/java -version (should display the java version)
- If you are working in UNIX or Linux, execute these commands:

  $ANT_HOME/bin/ant – version (should display the ant version)

  $JAVA_HOME/bin/java - version (should display the java version)

8.3 Edit the Build File

8.3.1 Windows Environment

1. Edit the $PAYMENTECH_HOME/ant/buildwindows.xml file.
2. Enter the name of the directory containing the source directory.
    a. Example: <property name="projectname" value="PaymentechSDK_6.3.0"/>

8.3.2 UNIX Environment

3. Edit the $PAYMENTECH_HOME/ant/buildunix.xml file.
4. Enter the name of the directory containing the source directory.

a. Example: <property name="projectname" value="PaymentechSDK_6.3.0"/>

8.4 Executing the Build

Ant build files are provided in the %PAYMENTECH_HOME%/ant directory. In that directory you should see a build script for your environment.

To build for Windows, type the following:

```
%ANT_HOME%/bin/ant –buildfile buildwindows.xml
```

To build for UNIX or Linux, type the following:

```
$ANT_HOME/bin/ant –buildfile buildunix.xml
```

These builds compile the entire Java SDK source, and then create a jar file, overwriting the PaymentechSDK.jar in the %PAYMENTECH_HOME%/lib directory. At this point, the build is complete.

9 SDK Functionality

This section outlines some of the unique functionality available through the Orbital Gateway.

9.1 Logging

The Java SDK framework implements Apache Commons-Logging (http://jakarta.apache.org/commons/logging/). This means that any on of the Apache Commons-Logging supported logging types can be used with the SDK without any code changes, but may require logging configuration changes if you decide to use a logging implementation other then the implementation provided.

By default, the Java SDK will use Log4J to perform its engine and transaction logging. The Log4J configuration properties can be found in the linehandler.properties file.

9.1.1 Configuring the Java SDK to work with an existing logging implementation

9.1.1.1 Using Log4J

Since Log4J operates as a Singleton within a JVM, the Java SDK first checks if the Log4J Singleton has been instantiated. If Log4J has been configured, the SDK will try to use the existing instance (and not try to reconfigure which could mean the loss of the previously loaded Log4J configurations). In this case, the best implementation strategy would be to combine the SDK's Log4J settings with the implementing application's Log4J setting. This can be done by copying the "eCommerce Logger Configuration" and "Engine Logger Configuration" Log4J properties from the linehandler.properties file and placing them in the application's Log4J configuration file.

NOTE: If you decide to configure the SDK logging through your Log4J properties file, it is highly advised that you use the Log4J VM argument "-Dlog4j.configuration". This will ensure that Log4J finds the Log4J properties file and it is loaded prior to the SDK needed it.

Example:

```
java -Dlog4j.configuration=/sdkhome/config/customLog4J/properties ….
```

The PAYMENTECH_HOME/config directory includes 2 sample implementations of custom Log4J configuration file (properties and xml). These samples will configure Log4J to send logging information to 3 log files, 1 custom and 2 Paymentech. The logging will not be mixed.

9.2 Request Generation

The Request object is used to load the transaction data for all types of transactions. A Request object is created by passing the name of the transaction you wish to create in the object constructor.

```
RequestIF newRequest = new Request (<name of transaction type>)
```

NOTE: *Whenever possible, it is suggested that you use the published Chase Paymentech SDK interfaces.*

The possible values for a transaction type are:

- NewOrder - Auth, Auth Capture, Prior Auth/Capture and Refunds
- EOD
- Gift Card
- MFC
- Profile
- Reverse – [aka Voids]

The RequestIF interface has constants defined for all of these transaction types.

9.2.1 *Complex Types*

Complex types are basically how the SDK represents XML elements that have children nodes and allows for dynamic XML generation. In the following example, the node "xml" has a child node of "data1" and "data1" has a child node of "data2".

```
<xml>
        <data1>
                <data2>some data</data2>
        </data1>
</xml>
```

Complex types consist of 2 types:

- Complex Root Elements (created from getComplexRoot method of RequestIF and TemplateIF) – pieces of XML that are either added to the transaction or removed based on the presence of needed.
- Recursive Elements (created from getRecursiveElement of TemplateIF) – repetitive pieces of XML that can be continuously added such as line items.

9.2.1.1 Purchasing Card 3 Support (with recursive line items)

Transactions that need Purchasing Card 3 (PC3) support must create both Complex Roots and Recursive Elements. *Currently, only the New Order and MFC transaction types support PC3.*

The first step to adding PC3 data is to create the top level PC3 elements. You get the proper PC3 implementation of a Template (TemplateIF) object by calling the "getComplexRoot" method of the request object. The "getComplexRoot" method takes one parameter. *The possible valid parameters are all defined in the RequestIF interface.*

```
// create core Pcard 3 elements
TemplateIF pc3Root = request.getComplexRoot(RequestIF.PC3_CORE);

// add data to the top Pcard 3 elements
pc3Root.setFieldValue("PC3FreightAmt", "100");
pc3Root.setFieldValue("PC3DutyAmt", "100");
pc3Root.setFieldValue("PC3DestCountryCd", "USA");
pc3Root.setFieldValue("PC3ShipFromZip", "34667");
pc3Root.setFieldValue("PC3DiscAmt", "100");
pc3Root.setFieldValue("PC3VATtaxAmt", "100");
pc3Root.setFieldValue("PC3VATtaxRate", "2");
pc3Root.setFieldValue("PC3AltTaxInd", "1");
pc3Root.setFieldValue("PC3AltTaxAmt", "100");
```

Now that the core elements are filled we can start adding the line items. The PC3 line items are children of the PC3 core and are created by calling the "getRecursiveElement" method of the PC3 core Template object (pc3Root in the above example).

```
            // add 5 line items (recursive elements)
            TemplateIF lineItem = null;

            for (int i = 0; i < 5; i++) {

              // create the line item
              lineItem = pc3Root.getRecursiveElement(RequestIF.PC3_LINE_ITEMS);

              // fill the line item with data
              lineItem.setFieldValue("PC3DtlDesc", "1234567890123456789");
              lineItem.setFieldValue("PC3DtlProdCd", "123456789");
              lineItem.setFieldValue("PC3DtlQty", "1");
              lineItem.setFieldValue("PC3DtlUOM", "LBR");
              lineItem.setFieldValue("PC3DtlTaxAmt", "0");
              lineItem.setFieldValue("PC3DtlTaxRate", "0");
              lineItem.setFieldValue("PC3Dtllinetot", "100");
              lineItem.setFieldValue("PC3DtlDisc", "100");
              lineItem.setFieldValue("PC3DtlCommCd", "3");
              lineItem.setFieldValue("PC3DtlUnitCost", "100");
              lineItem.setFieldValue("PC3DtlGrossNet", "Y");
              lineItem.setFieldValue("PC3DtlTaxType", "Y");
              lineItem.setFieldValue("PC3DtlDiscInd", "Y");
              lineItem.setFieldValue("PC3DtlDebitInd", "D");
      }
```

9.2.1.2 Adding Prior Authorization ID

Transactions that need a prior authorization id will need to add a single Complex Root. *Currently, only the New Order and Flex Cache transaction types support prior authorization id.*

The first step to adding prior authorization data is to create the top level PriorAuthID element. You get the proper PriorAuthID implementation of a Template (TemplateIF) object by calling the "getComplexRoot" method of the request object. The "getComplexRoot" method takes one parameter. *The possible valid parameters are all defined in the RequestIF interface.*

```
              // create core Pcard 3 elements
              TemplateIF priorAuthID = request.getComplexRoot(RequestIF. PRIOR_AUTH_ID);
```

You can then use the newly created TemplateIF object to populate the value of the PriorAuthID element in the XML transaction using the setFieldValue method.

```
              PriorAuthID.setFieldValue ("PriorAuthID", "123456");
```

9.2.1.3 Settling Rejected Transactions

To settle your reject bin, you must include the SettleRejectBin XML element in an End of Day request transaction.

The only step to adding the SettleRejectBin XML element to the EOD transaction is to create the proper SettleRejectBin implementation of a Template (TemplateIF) object by calling the “getComplexRoot” method of the request object. The “getComplexRoot” method takes one parameter. *The possible valid parameters are all defined in the RequestIF interface.*

NOTE: It is not needed to set any value for the RejectSettleBin element. Creating the TemplateIF is sufficient.

```
TemplateIF settleRejectedBin= request.getComplexRoot(RequestIF. SETTLE_REJECT_BIN);
```

10 Business Rules

This section provides the business rules for the functionality that is more complex.

10.1 Account Verification

Account Verification transactions provide the ability to verify accounts without financially impacting the accountholder’s open-to-buy. Address Verification Service (AVS) and Card Security Value can be verified along with the account number.

Some key points regarding Account Verification messages are:

- New Order request must be used
- Transaction type must be an Authorization Only ('A')
- Amount must be '0'
- The minimum of AVS ZIP is required
- Card Security Value is optional
- All mandatory fields must be submitted

Account Verification is supported in all currencies

Account Verification for Salem is supported by:
Visa, MasterCard

10.2 Reversals

The Reversal message is used to void a transaction either in the full amount or partial amount. It can be extended to also reverse the authorization at the issuer.

A void does not reverse the original authorization for any card type other than Gift Card and PINless Debit. An authorization reversal frees-up the accountholder’s open-to-buy, which has been reserved by the original authorization. This is done at the Issuer’s discretion.

To indicate that an authorization reversal is being requested in addition to the void, the Online Reversal Indicator element must be submitted in the Reversal message. A value of 'N' or NULL indicates that a void is being requested. A value of 'Y' extends the void request to also include the authorization reversal.

Merchants can submit the Online Reversal Indicator element via the Reversal request or they can be configured at the Host to allow the Gateway to submit the indicator on their behalf.

When allowing the Gateway to submit the indicator, the Gateway will attempt an authorization reversal wherever applicable. In the event the original authorization doesn't meet the requirements for an authorization reversal or an error occurs while attempting an authorization reversal, the Gateway will perform a void instead. For each occurrence where the indicator is sent in the Reversal request by the merchant, the indicator value in the message will take precedence over the indicator selection that is configured on the Gateway.

The following requirements must be met in order to perform a void:
1. The transaction must not have been settled
2. The Transaction Reference Number from the response message of the original request must be provided. If the Transaction Reference Number is not known, merchants can submit in its place the Retry Trace Number of the original request within the Reversal Retry Number element.
3. The full or a partial amount must be submitted. A void for a partial amount creates a split of the original

transaction into two components. A voided transaction in the amount of the partial void request and the remainder of the previous transaction in the same state the full amount was previously in (Authorized or Marked for Capture).

The following requirements must be met when extending the void request to include an authorization reversal:
The original authorization must have been obtained through Chase Paymentech, or the transaction will decline
The original authorization cannot be greater than 72 hours old
Reversal must be for full amount that was received in the authorization

Authorization Reversals for Salem is supported by:
Visa, MasterCard, Discover

10.3 Inquiry

An Inquiry transaction returns the response of any specified request. This is useful when a merchant needs to know the result of a transaction in the case of, for example, a communication error or unexpected reult. An InquiryRetryNumber value, which corresponds to the Retry Trace Number of the originating transaction, must be passed in the Inquiry request message in order to obtain the response. If there is no matching result, an error message is returned. Similar to the Retry Trace Number, the Inquiry Retry Number is valid within a 48-hour window from the time of the original transaction.

The basic process flow for an Inquiry is as follows:

- A transaction is submitted with a Retry Trace Number and Merchant ID in the request
- The merchant does not receive a response and subsequently submits an inquiry using the Retry Trace Number (as the Inquiry Retry Number) and Merchant ID
- The Gateway validates the Inquiry Retry Number and Merchant ID to determine if it has processed a transaction using that value pair within a 48-hour window
- The Gateway returns the transaction response details for the original request if the transaction was found

For more information about the implementation of Retry Trace Numbers, please see the Retry Logic section of this manual.

Complex Type Name:
Inquiry Request = Inquiry
Inquiry Response = InquiryResp

10.4 Retry Logic

10.4.1 Overview

When processing transactions over the Internet, there is always the risk that a response to a request will not be received. When processing credit card authorizations especially, clients require a means to determine the result of the transaction through a programmatic interface.

To address this problem, the Orbital Gateway offers functionality, referred to as "Retry Logic", which allows a customer to retry a transaction without duplicating the original request.

The result is any Client properly utilizing Retry Logic can reprocess transactions with an unknown result without:

- Risk of double authorizing a transaction against a cardholder's available balance.
- Prevent duplicate [or more] settlement items.

The purpose of this documentation is to provide a developer with the necessary information on how to correctly implement Retry Logic both in terms of the necessary coding changes and the necessary business logic.

10.4.2 Business Rules

Retry is available to any merchant interfacing to the Orbital Gateway using the Java SDK [version 3.2 and higher] by simply setting one new value, a TraceNumber. The Orbital Gateway uses a combination of the TraceNumber and Merchant ID to determine the uniqueness of a transaction in determining how to process the transaction.

The basic process flow of Retry Logic is as follows:

- A Request is submitted with a TraceNumber and Merchant ID.
- The Gateway validates the TraceNumber and Merchant ID to determine if it has processed a transaction using that value pair within a 48-hour window.
  - If the transaction is a decline or error on the initial response, the next request will be treated as a new request.
  - If it has not processed the pair, the Gateway will treat that transaction as a new request and process it accordingly.
  - If it has processed the pair and the Request has either already been processed [the initial response is an approval] or is in process, the Orbital Gateway will immediately echo back the exact response XML Document from the initial request.
  - If the initial Request is still in process, the Orbital Gateway will block and wait until that original response is completed. As soon as that is done, it will then echo back the same response as the original request.

For more information on Retry Logic, please reference the Retry Specification v2.0 [or higher].

10.4.3 TraceNumber rules:

The Trace-number Field rules are:

1. Data Type = Numeric
2. Minimum Length = 1
3. Maximum Length = 16
4. As such, Valid Values are from 1 – 9999999999999999
5. Submitting an invalid value will result in an XML Quick Response with a ProcStatus Code of 9714.

10.4.4 Enabling Automatic Retries in the Orbital Java SDK

To enable the automatic retry of a transaction, you must:

Set a Trace Number into the request object using the setTraceNumber() method before submitting it to the Transaction Processor. The Trace Number must be a numeric value (a String representation of an integer).

Set the connection_timeout_seconds in the linehandler.properties to a value greater then or equal to 90.

10.5 Cardholder Authentication

10.5.1 Address Verification

Address Verification, also known as AVS, is a cardholder authentication mechanism available to merchants. In addition to providing merchants with an additional risk management tool, it is required by Visa and MasterCard to qualify for the lowest interchange rates and protects against certain chargeback conditions. As such, it is highly recommended by Chase Paymentech that all transactions include this information. Two keys on AVS are:

- The minimum required data for AVS is the Cardholders billing ZIP.
- AVS is only supported by credit cards issued in the United States. Foreign card issuers, including but not limited to Canada and Europe, do not issue it.
- For Tampa routed accounts [BIN 000002], the Orbital Gateway does not support ZIP formats outside the US Convention of NNNNN and NNNNN-NNNN. If a transaction is submitted with a ZIP outside this data convention, the entire transaction will reject for invalid ZIP structure.
- For Salem routed accounts [BIN 000001], the Orbital Gateway will accept ZIPs formatted alpha-numeric with a length between 1 and 10 in length. These ZIP's will be forwarded to the Salem authorization host. However, no ZIPs will actually be processed except US based ZIPs. International ZIPs will receive a response stating "International Zip – Issuer not supported."
- The Card Types that support AVS are:
  - Visa
  - MasterCard
  - American Express
  - Discover

10.5.2 *Card Verification Numbers*

The Orbital Gateway supports the submission of Card Verification Numbers [see more detailed definition below] the specific field names for this value in the SDK interface are CardSecVal & CardSecValInd.

10.5.2.1 Visa CVV2/MasterCard CVC2/Discover CID Programs

The Orbital Gateway supports Visa's CVV2 (Card Verification Value 2), MasterCard's CVC2 (Card Validation Code 2) and Discover's CID fraud reduction programs. This section will provide some background information on supporting these programs.

The value for these cards is 3 digits and can be found on the signature panel on the reverse side of the credit card and is represented by the three digits following the account number.

This value may not be stored at all. Not even for future transactions, as it is against regulations to do so.

The use of this value provides an important security check due to the fact that only the individual in possession of the actual credit card will be able to provide the value to the merchant. Statistics validate those individuals who may know the account number, but are not in possession of the actual credit card perpetrating much of the fraud occurring in the non-face-to-face environment.

When a merchant collects this value and passes it in the authorization request, Chase Paymentech passes this data through the authorization system to the card issuer. In the authorization response, the card issuer validates the accuracy of CVV2/CVC2/Discover CID value for the specific card. Used in conjunction with the valid expiration date, this service provides a valuable tool for assessing the true cardholder has placed the order with you for your services or product.

10.5.2.2 American Express CID Merchant Processing Requirements

American Express provides a similar program to Visa, MasterCard and Discover except with a few differences:

- The value for these cards is 4 digits and is printed, not embossed, on the front of all cards. On the American Express card it appears on the right border of the card, however on Optima cards, it appears on the left border of the card.
- In situations where the CID value is invalid, American Express responds with an authorization decline message; as opposed to the other card types, which respond with a separate CVV2, CVC2 or CID response code.
- To process American Express CID, the merchant must contact their American Express Client Manager to have their American Express service establishment numbers flagged to accept the Amex CID value. American Express will not provide validation of the CID value if the merchant's Service Establishment (SE #) is not enabled.

10.5.2.3 Gift Card/Stored Value Application

The Chase Paymentech Gift Card™ program supports CVD2 (Card Verification Data 2), which is also known as a PIN, as an optional feature determined by the merchant. The four-digit value may be imprinted on the back of the stored value card, and used to facilitate a secure Card Not Present transaction when the consumer wishes to use a Gift Card card as their method of payment.

10.5.2.4 Guidelines for populating Card Security Fields

- The Card Security Fields are called <CardSecValInd> & <CardSecVal>.

10.5.3 Verified By Visa and MasterCard SecureCode

Verified by Visa and MasterCard SecureCode are both solutions designed to authenticate cardholders when paying online. These products offer a mechanism for securing the Internet channel by strongly authenticating the cardholder at the point of interaction by providing a unique transaction-specific token that provides evidence that the cardholder originated the transaction.

10.5.3.1 Verified by Visa

Verified by Visa is based on the 3-D Secure Protocol, which uses Secure Sockets Layer (SSL) encryption to collect and protect payment card information transmitted via the Internet. It uses three domains for the authentication process:

- Issuer Domain – Where the Issuer is responsible for determining whether authentication is available for the card account presented in a purchase.
- Acquirer Domain – Where the Acquirer accepts Internet transaction data from the Merchant and passes it to Visa.
- Interoperability / Visa Domain – This is operated by Visa, where transaction information is exchanged and stored using 3-D Secure as the common protocol.

10.5.3.1.1 Transaction Flow

The cardholder shops at participating Internet Merchants with no changes to the shopping or checkout. The cardholder selects the merchandise to be purchased and proceeds to the checkout. At the checkout, the cardholder may complete the purchase and payment information a variety of ways, including self-entered, an electronic wallet; Merchant one-click, or using other checkout capabilities.

After the purchase and payment information is entered, the cardholder selects the "buy" button. This activates the Merchant Server Plug-In (MPI) software application, which checks its local cache to determine if the Visa Issuer BIN participates in Verified by Visa. If the BIN is participating, the MPI will generate an inquiry to the Visa Directory Server to determine if the cardholder's account is enrolled in Verified by Visa. The Visa Directory Server sends a Verify Enrollment Request message to the Issuer Access Control Server (ACS) to determine if authentication is available for the cardholder's account number. The Visa Directory Server sends the Issuer ACS response to the MPI. If authentication is not available, the Merchant server receives an "authentication not available" message and returns the transaction to the Merchant's commerce server to proceed with a standard Authorization request. If authentication is available, the message response provides the URL for the Issuer ACS where the cardholder can be authenticated. The MPI sends a message and script directing the cardholder's browser to establish a pop up session with the Issuer ACS.

The browser directs the transaction to the URL specified for the Issuer ACS creating an SSL session. The Issuer ACS displays an inline web page to the cardholder. The page includes Issuer-specific and Visa branding, transaction details (including Merchant name and sale amount), and prompts the cardholder to enter their password. The cardholder is allowed a limited number of password attempts, typically 3 to 5, as defined by the Issuer ACS. If unable to correctly enter the password, the cardholder may access the password hint that was established during the registration. If the password is entered correctly, the transaction continues. If the cardholder is not registered, the ACS briefly displays a processing window and the transaction continues as an attempted authentication. If the password is incorrectly entered more times that the Issuer limit, the failed Payer Authentication Response is returned to the Merchant.

The Issuer ACS retrieves the authentication information and compares it against the data that was registered during the initial registration process. If the data matches, a success page is presented to the cardholder and the Issuer ACS sends a message through the browser to the merchant, thus providing evidence of cardholder authentication. Using the Issuer's encryption keys and transaction data, the Issuer server calculates the Cardholder Authentication Verification Value (CAVV), which will be included with the Electronic Commerce Indicator (ECI), as provided at the time of authentication by the MPI, in the response to the Merchant.

The Issuer ACS creates, digitally signs, and sends a Payer Authentication Response to the cardholder's browser, and sends transaction information to the Visa Authentication History Server (AHS) for storage. All Payer Authentication Response messages (successful, unable to authenticate, failed, and attempted authentications) are transmitted and stored in the AHS. The browser routes the Payer Authentication Response back to the MPI, which validates the digital signature in the response, verifying that it is from a valid participating Issuer. If the digital signature is verified and the Issuer has sent an approved Payer Authentication Response, the cardholder is deemed authenticated and the MPI returns the transaction to the storefront software. The Merchant starts processing the order, determining whether it can be fulfilled, and calculating taxes and shipping for the total transaction amount.

The Merchant will send the CAVV, ECI of 5 (authenticated transaction) or 6 (attempted authentication) to the Orbital Gateway. The CAVV must be sent in Base 64 encoding within the XML Document. If the CAVV is not submitted in Base 64 encoding or if the CAVV is sent with a non-eCommerce transaction, a response code of 37 in the respCode field will be returned.

Chase Paymentech will pass the CAVV and ECI along to Visa with the authorization request. These fields are used during authorization processing to verify that authentication, or attempted authentication, was performed and to qualify for the eCommerce Customer Payment Services.

The Issuer receives the authorization request, validates the CAVV and responds with a CAVV Response Code as well as an approval or a decline of the authorization. If the CAVV does not match, the Issuer should decline the transaction.

Visa has not implemented any new decline codes for Verified by Visa. The standard decline codes should apply.

NOTE: A Merchant may not submit for authorization a purchase transaction that has failed authentication.

10.5.3.2 MasterCard SecureCode

1. MasterCard SecureCode is a solution designed to authenticate cardholders when paying online. SecureCode offers a mechanism for securing the Internet channel by strongly authenticating the cardholder at the point of interaction by providing a unique transaction-specific token that provides evidence that the cardholder originated the transaction. SecureCode uses MasterCard's Universal Cardholder Authentication Field (UCAF) infrastructure to communicate the authentication information among the cardholder, Issuer, merchant and Acquirer.
2. MasterCard SecureCode supports the 3-D Secure Protocol (same as Verified by Visa). MasterCard SecureCode requires merchants to install a 3-D Secure v1.0.2 compliant Merchant Server Plug-in (MPI) software.

10.5.3.2.1 Transaction Flow

The cardholder shops at a participating SecureCode Internet Merchant with no changes to the shopping or checkout. The cardholder selects the merchandise to be purchased and proceeds to the checkout. At the checkout, the cardholder may complete the purchase and payment information in a variety of ways, including self-entered, an electronic wallet; Merchant one-click, or using other checkout capabilities.

After the purchase and payment information is entered, the cardholder selects the “buy” button. The customer shopping experience is the same for both of the Issuer platforms up until the time that the Merchant Order Confirmation page is displayed.

The MPI activates and checks its local cache and the MC Directory Server to determine if the customer card number is part of a participating MasterCard SecureCode BIN range. If so, a Verify Enrollment Request message will be sent from the MPI, to the MC Directory Server and forwarded to the Issuer Access Control Server (ACS) to determine if authentication is available for the cardholders account number. The MC Directory Server sends the ACS response to the MPI. If authentication is available, the message response provides the web address for the Issuer ACS where the cardholder can be authenticated. If authentication is not available, the Merchant server receives an “authentication not available” message and returns the transaction to the Merchant’s commerce server to proceed with a standard Authorization Request. There is no “attempted SecureCode” transaction type unlike Verified by Visa [ECI = 6].

The MPI sends a message and script directing the cardholder’s browser to establish an inline web page session with the Issuer ACS. The window displays Issuer-specific and MasterCard branding, transaction details – including Merchant name and amount, and prompts the cardholder to enter their SecureCode (e.g. password). If the password is entered correctly, the transaction continues. The cardholder is allowed a limited number of password attempts, typically 3 to 5, as defined by the Issuer ACS. If unable to correctly enter the password, the cardholder may access the password hint that was established during the registration. If the password is incorrectly entered more times than the Issuer limit, a failed Payer Authentication Response is returned to the Merchant.

The Issuer ACS retrieves the authentication information and compares it against the data that was registered during the initial cardholder registration process. If the data matches, a success page is presented to the cardholder and the Issuer ACS sends a message through the browser to the Merchant providing evidence of cardholder authentication, including a 28-byte Account AAV. This AAV is generated cryptographically using Issuer-specific secret keys that are synchronized with keys at the Issuer’s authorization platform.

10.5.4 Merchant Submission of AAV Data

The merchant will then send the transaction to Chase Paymentech, along with the 28-byte AAV in Base 64 encoding, within the Orbital Gateway XML Interface Specification.

If the AAV is not submitted in Base 64 encoding or if the AAV is sent with a non-eCommerce transaction, a response code of 37 in the respCode field will be returned. Also, if the Merchant has not tested and certified with Chase Paymentech to participate in MasterCard SecureCode, and an AAV is sent with the e-Commerce transaction, a response code of 38 in the respCode field will be returned, which indicates the Merchant should contact their Chase Paymentech Representative to become SecureCode enabled.

Chase Paymentech will forward the transaction, including the AAV in the MC authorization request. The Issuer receives the authorization request, validates the AAV and responds with an approval or a decline of the authorization. If the AAV does not match, the Issuer should decline the transaction.

NOTE: MasterCard has not implemented any new decline codes for SecureCode. Standard decline codes apply.

10.5.4.1 Merchant Plug-in Software

Install a Certified 3-D Secure Merchant Plug-in Software Application or code to the 3-D Secure Protocol.

Verify that Merchant Plug-in will provide the CAVV and or AAV in Base 64 encoding before sending to Chase Paymentech. If not, Merchant will need to convert to Base 64 before sending to Chase Paymentech.

10.5.4.2 Business Rules

There are a number of business rules in terms of when a CAVV and or AAV should be presented on aged transaction, reauthorizations, split transactions, etc. The Orbital Gateway abstracts your interface from many of these issues. This section will outline what these rules are and what is necessary to understand from an Interface Perspective.

10.5.4.2.1 Authorizations

Merchants are required to request authorization for all Verified by Visa and MasterCard SecureCode eCommerce transactions. Merchants must supply the CAVV and ECI on all Visa authorization attempts and the AAV on all MasterCard Authorization attempts.

10.5.4.2.2 Failed Authorizations

Merchants are prohibited from submitting transactions for authorization that have failed authentication.

10.5.4.2.3 Late Fulfillment

When a participating merchant splits the shipment of an order, each authorization component may be submitted with the authentication data (ECI of 5 or 6 [6 is for VbV only] and the CAVV or AAV) of the original purchase. In the event of a dispute, the Acquirer must be able to establish that the authorization requests were related to a single customer authenticated purchase. Furthermore, if a Deposit record is sent for the subsequent shipment, the authorization will already have been tagged as “used”, therefore, in order to receive the full benefit of Verified by Visa and MC SecureCode, a merchant must send the authentication data with the subsequent deposit record so that when Chase Paymentech reauthorizes, the authentication data can be sent as well.

10.5.4.2.4 MasterCard SecureCode

Initial SecureCode authorization requests with AAV’s older than 30 calendar days may be declined by the Issuer.

10.5.4.2.5 Recurring Transactions

When a participating Merchant offers services of an ongoing nature to a cardholder for whom the cardholder pays on a recurring basis (for example, insurance premiums, subscriptions, Internet service provider fees, membership fees, tuition, or utility charges), the cardholder payments are considered recurring payments.

If the first payment originated as an Electronic Commerce Transaction via the Internet, it must be submitted with the appropriate Electronic Commerce Indicator (ECI) value, including Verified by Visa or MasterCard Secure Code authentication data (CAVV or AAV respectively), if applicable.

All subsequent payments must be submitted as Recurring transactions [see Orbital Gateway Interface Specification and Message Templates documents). The merchant must not store and submit the CAVV with any subsequent transaction.

10.5.4.2.6 Currencies Supported

All Currencies

10.5.5 Chargeback Liability Sift Exclusions

10.5.5.1 Verified by Visa

There are certain exclusions from the Chargeback provisions related to attempted authentications. They are:

- All Visa Commercial Cards (Visa Business, Visa Purchasing and Visa Corporate Cards), anonymous Prepaid Cards (such as gift cards), and transactions from new channels (such s mobile devices) are excluded from chargeback protections for attempted authentications. If these cards are enrolled in Verified by Visa and the Issuer authenticates the cardholder, the Issuer is not permitted to submit a chargeback for unauthorized usage disputes (reason codes 23, 61, 75, and 83). Either the Issuer ACS or Visa may designate excluded transactions; however, the Visa Directory Server will over-ride excluded responses from an Issuer ACS if the BINs are not also designated as excluded BINs in the Visa Directory. The designation of BINs as Commercial or anonymous Prepaid Cards must be consistent with VisaNet.
- Transactions conducted in new channels (such as mobile or wireless devices).
- Merchants named in the Global Merchant Chargeback Monitoring Program are not eligible for Chargeback protection for attempted authentications during the time that they are required to participate in the program and three months thereafter. Visa will work with Acquirers to ensure compliance with this requirement. There are no additional steps for Issuers regarding this provision.

10.6 Profiles

The Orbital Gateway includes functionality called Customer Profile Management, which allows Cardholder data to be stored with the Orbital Gateway. A merchant can process transactions by simply passing a token value that represents that cardholder.

Once a Profile is created, transactions can be processed, using either the on-line interface or the Orbital Virtual Terminal ["VT"], simply by referencing the Customer Profile and filling in any additional information not stored in the profile. This feature is only available to Merchants using the Chase Paymentech Orbital Interface.

Released in March of 2008, Managed Billing extends the capabilites of Profiles to include Recurring, Installment, and Deferred billing. Using this feature, merchants can configure future payments which the Orbital Gateway will initiate on the desired date.

10.6.1 Supports both Recurring and non-Recurring charges

By default, Profiles do not provide a full recurring service. The Orbital Gateway will store all the relevant information for processing a transaction; it will not automatically process it. When using standard Profiles, merchants are required to initiate a Profile request to the Orbital Gateway and retrieve the result of that request.

Profiles can also be configured to bill automatically via a process known as Managed Billing. Merchants wishing to use Managed Billing to support recurring, installment, or deferred charges will need to have the Managed Billing feature enabled for their account. A merchant contract addendum is required to enable this feature, so interested merchants should contact their sales rep or Account Executive. See the Managed Billing section below for more info. Additionally, please reference the supplemental "Managed Billing 101" document for more information about the overall product, it's features, and how merchants can use the Managed Billing features.

10.6.2 Benefits

There are a number of potential benefits when using the Profiles feature:

- It simplifies transaction processing. When making a transaction request, one simply references the Profile ID [a.k.a. the Customer Reference Number] and fills in any of the missing information.

- It eliminates risk. Since it eliminates the need to store sensitive information about your customer on your database, merchants can focus on their business, and Chase Paymentech can focus on securely processing your transactions.
- It can eliminate data entry errors when using the Virtual Terminal. By retrieving a pre-existing Profile and validating the data, it eliminates the risk of "keying" the wrong customer information such as Order # [which may equate to a Membership ID] or credit card #.

10.6.3 Setup Information

For any Orbital Gateway Merchant ID to support Profiles, it must be configured on the Orbital System to do so. There are several different configuration aspects that must be setup.

- Enablement: First the Merchant ID must be configured to allow Profile functionality. Any Merchant ID that is not configured to use Customer Profiles and attempts to process a Profile Action will receive an error. A Profile Error Code of 9578 or "Merchant-Bin combination is not allowed to perform profile transactions" will be received.
- Customer Profile Hierarchy Support: Each Merchant ID must be configured to support Profiles at Chain ID (Company) level or Merchant ID level.
  - Note: Managed Billing requires that Profiles be configured at the Merchant ID level.
- Virtual Terminal Users: If your organization will utilize Profiles on the VT in addition to the XML interface, there are a few important considerations:

Profile User Management

- *Profile Administration:* For any VT User to administer Profiles [i.e. add, delete, update], that User must be provided the "right" to administer Profiles. Any existing User can be granted this additional User permission.
- *Profile Usage:* For any VT User to use Profiles for processing a transaction, permission needs be granted to use profiles. Any existing User can be granted this additional User permission. The User will not be able to administer profiles, just use existing ones.
- *Profile Disabled*: If the VT User is not enabled for any Profile access level, they will not see any of the functionality. Profiles can be disabled for one user and enabled for another user.

General Access Rights:

- *Card Masking:* The same card masking rules apply to Profile management or usage as they do to any card number viewing in the VT previously.
  - If a User's permission allows the viewing of the credit card number, then during usage or management, that User will be allowed to see any credit card number whether maintaining a profile or using it.
  - Conversely, if that User's permission level does not allow the number to be viewed, then it cannot be viewed whether they have the right to maintain a profile or use it. However, the card can be changed or updated regardless of masking.
- *Access Levels:* All existing access levels will not be impacted regardless of Profile user rights. For instance, if a User cannot submit credits, they will not be able to submit credits using Profiles.

10.6.4 Business Rules

10.6.4.1 How it works

Profile Management is a very simple product. The first step is to create a Profile. This can be done in two different fashions:

- Adding a Profile as a distinct action
- Or adding Profile as a part of an authorization request.

Once that Profile exists, it can be utilized to complete a sale or refund with any of the data elements stored in the profile. Additionally, any part of the Profile can be overridden during the subsequent transactions.

Finally, the Profile can be updated [or even deleted] at any point.

10.6.4.2 Customer Reference Number Options

The Customer Reference Number [<CustomerRefNum> in the schema or 'Profile Number' in the Virtual Terminal] is the referential data element to a Profile.

Key Customer Reference Numbers facts:

- Must be unique [either by Merchant ID or Chain ID – see notes below].
- Can be from 1 to 22 bytes in length.
- Valid Characters are: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-,$@& and a space character.

- Please note that although lowercase characters can be submitted, all Alpha characters are stored in uppercase by the Orbital system. Therefore, users cannot use upper and lower case values to differentiate profile IDs.
- Because the ampersand (&) has unique properties within XML, an ampersand must be sent as: &amp;

10.6.4.2.1 Setting the Customer Reference Number

The merchant can either set or request that the Orbital Gateway set the Customer Reference Number.

The field customerProfileFromOrderInd controls this behavior as follows:

A - Auto Generate the Customer Reference Number. In other words, the Orbital Gateway will assign the Customer Reference Number and return it in the response.

S - The Orbital Gateway will use the value passed in the <CustomerRefNum> Element as the Customer Reference Number.

O - This option only relates to when a Profile is added as a part of an authorization request. In this circumstance, the value passed in the <OrderID> element will be used as the Customer Reference Number. This would be used in circumstances wherein the order ID also represents the Customer ID in your system, such as a Policy Number for an insurance company.

D - This option only relates to when a Profile is added as a part of an authorization request. In this circumstance, the value passed in the <Comments> element will be used as the Customer Reference Number.

Note: Set field to "EMPTY" when using a customer profile. <CustomerProfileFromOrderInd="EMPTY">. This value is case sensitive.

10.6.4.2.2 Using the Customer Reference Number to set other data elements

The Orbital Gateway has options in configuring the Profile setup in terms of how the Profile ID can be leveraged to populate other data sets using the <CustomerProfileOrderOverideInd> value.

The options are:

NO - No mapping to order data.

OI - Pre-populate the following fields with the Customer Reference Number:

- <OrderID>
- <ECOrderNum> if it is an eCommerce Industry Type Transaction
- <MailOrderNum> if it is an Mail Order or Recurring Industry Type Transaction

OD - Pre-populate the <Comments> field

[**Note**: this field is called Order Description in the Virtual Terminal] with the Customer Reference Number. The relevance of this feature is on the PNS platform [BIN 000002] the <Comments> field is what populates the Customer Definable Data. This data can then be made available on certain Resource Online Reports. Any questions about your reports should be directed to your Relationship Manager.

OA - Pre-populate the following fields with the Customer Reference Number:

- <OrderID>
- <ECOrderNum> if it is an eCommerce Industry Type Transaction
- <MailOrderNum> if it is an Mail Order or Recurring Industry
- <Comments>

10.6.4.3 Customer Reference Number hierarchy setup and usage

As stated earlier, Profiles can be created at Merchant ID Level or at the Chain level.

If a MID is configured to use Profiles at a Chain ID level, any profiles setup by any Merchant ID are available to be used by any other Merchant ID's tied to that chain. However, if the MID is setup to manage Profiles at the merchant level, any Profile setup by that Merchant ID can only be used by that Merchant ID.

For example:

- Let's assume there is a single customer with two Merchant ID's on the Orbital Gateway, 11111 and 222222.
- These two merchant ID's are tied to the same Chain ID, 333333.
- Merchant ID 111111 sets up a new customer profile ABC.
- If both Merchant ID 111111 and Merchant ID 222222 are setup to manage Profiles at a chain level, then Merchant ID 222222 will be able to use profile ABC.
- If either one of them is not, the Merchant ID 222222 will not be able to use profile ABC.

Notes:

- All Profile Setups are performed at a Merchant ID. So this cross Chain ID sharing can only be facilitated via Orbital Setup.
- In addition, given that all setup and usage of Profile ID's is done using a specific Merchant ID, there is requirement that the Chain ID be known to take advantage if this feature. As long as all the Merchant ID's are properly linked to the same chain, it will simply work. If the Merchant ID's are not correctly mapped to the same Chain ID, Merchant ID's can be remapped to new Chain ID's easily. If this feature will be used, it is recommended that the correct chaining be validated prior to going live.
- Whatever level is defined as the Storage level, there can only be one version of a Customer Reference Number. Therefore if two different Merchant ID's have different customers who share the same Customer ID, it would be recommended that the Profile storage and usage be maintained at the Merchant ID level as opposed to the Chain level. If the second store tried to establish the same Customer Reference Number and the setup dictated a chain level storage, then a Duplicate Customer Reference Number error [<ProfileProcStatus> error code of 9582].

10.6.5 Salem Hierarchy

For Salem Orbital Gateway customers, the Orbital Gateway hierarchy closely emulates the Salem hierarchy.

- The Orbital Gateway MID will be the same as your Salem Division [or TD] number.
- And the Orbital Gateway Chain ID will be the same value as your Company Number [formerly known as the MA].

If the Salem Division numbers are all linked to a specific Company number that is in fact how it will be setup on the Orbital Gateway.

10.6.6 PNS Hierarchy

For PNS Orbital Gateway customers, the Orbital Gateway hierarchy is tied to the PNS Authorization Host hierarchy. As such:

- The Orbital Gateway MID will be the same as your PNS Authorization Merchant ID [MID] – Terminal ID [TID].
- However, there is no PNS Chain value. So the Orbital Gateway assigns the next available chain value when setting up accounts for the first time.

If an organization has multiple Merchant IDs, there is no guarantee that all of those Orbital Gateway Merchant ID's will be linked under a single Chain ID. However, Merchant IDs can be moved under one chain to take advantage of this feature.

10.6.7 Profile Transaction Types

There is a number of transaction types associated with Profiles. Some of these are extensions of existing transaction types and some are new to profiles. This section will detail from an XML perspective how to support all those Profile transaction types and some of the specific rules associated with each of them. Again, all of the functionality identified within this document is possible through the Virtual Terminal as well.

10.6.8 Managing Profiles

There are a set of transactions specifically set up for managing the Profile; in other words, adding, updating, deleting and retrieving the information.

10.6.9 Adding Profiles

First and foremost, a profile needs to be added to the Orbital Gateway. There are two different transaction actions that can be performed to add a profile.

10.6.10 Adding Profiles as a Stand Alone Transaction

The simplest mechanism is to add a Profile is to simply make a Profile Add Request Type. This document includes both the definition of the XML elements necessary to complete this transaction and an example template of an Add Profile Request.

There are new response data elements that need to be interpreted to determine the success of this Add request.

10.6.11 Adding Profiles during an Auth

Given that in many circumstances, the fist time a customer is setup an authorization needs to be performed, the Orbital Gateway has extended the traditional Authorization transaction to enable adding a Profile in the same request.

- Any data included in the Authorization that can be saved as a part of the profile will be.
- The minimum data to create a profile must be included or no profile will be created.
- The result of the authorization is separate from the result of the profile add step. As such, an authorization can be successful and the Profile Add component can be an error on the same transaction and vice-versa. These results are mutually exclusive and should be interpreted from a response management process as such.
- Add profiles can only be included with Auth Only, Auth-Capture and Prior Auth [i.e. Force] transactions. It cannot be completed as a part of a Refund, Void, or Mark for Capture.

10.6.12 Information saved in a profile

Whether a Profile is added via a Profile Add transaction or added via an authorization or updated later via a Profile Update transaction, the following list of items defines what data elements can be saved as a part of a Profile.

- Customer Reference Number – Required and un-editable (Also referred to as Profile ID)
- Customer Name
- Customer Email – NOTE: Only Available as a Profile Add or Update. This XML element does not exist yet in the Authorization message set and cannot be sent as a part of an Auth Request profile Add as a result.
- Address Information:
    - Address 1
    - Address 2
    - City
    - State
    - ZIP
    - AVS Country Code

    - Phone
- Amount
- Description: This can be set in two ways. Either by sending a specific description message or by setting the <CustomerProfileOrderOverideInd> to populate the <Comments> tag, which is the Order Description.
- All Order # fields: This can be accomplished by setting the <CustomerProfileOrderOverideInd>
- Payment Information:
    - Card Type:
        - Credit Card
            - Card #
            - Expiration Date
        - ECP (Salem Host Only – BIN 000001)
            - DDA
            - R/T
            - Account Type
            - Payment Delivery Method
        - Switch Solo (Salem Host Only – BIN 000001)
            - Card #
            - Expiration Date
            - Start Date
            - Issue Number

10.6.13 Information NOT saved in a profile

There are a number of items that will not be added to Profile regardless of how it is done. This includes but is not limited to:

- Purchasing Card Data
- Card Verification Number [CVV2, CVC2, and CID]. This is because this information cannot be stored by card association regulation. It must be requested from a cardholder on a transaction-by-transaction basis.
- VbV and MasterCard SecureCode Data

10.6.14 Updating Profiles

Once a Profile has been added, any information about a Profile can be updated, except the Profile Keys [which again includes the Customer Reference Number, Merchant ID, and BIN]. This can be accomplished by sending a Profile Update transaction.

Some important keys to performing an Update:

- All Profile Update requests must include the correct Profile keys or an error message will be returned. (A list of the error messages can found in the appendix)
- An update requires the Tags to be sent for both:
    - The Data that should be updated.
    - And for any fields that should be cleared.
- To clear any legacy data, the XML Tag is submitted with nothing but a Tilde ['~']. See below example for performing this step:
    - <CCExpireDate>~</CCExpireDate>
- For example, if the Customer Profile included an amount and update was sent with the Amount Tag present, but filled with a Tilde character, the amount stored in the profile would be changed to null in the database.
- If an XML tag is sent with a Null value [such as <CCExpireDate></CCExpireDate>], it will be ignored as a part of the update process [i.e. no update would occur on the CCExpireDate value from the example].

- When changing Card Types, such as from an ECP to a Credit Card what is required is:
  - Send the XML tag representing the new card type
  - Submit the appropriate data for that card type.
  - Null the old card type data elements using the Tilde process described above. For example, changing from an ECP transaction type to a Credit Card type, the Profile Update message should:
    - Have the Card Type defined as Credit.
    - The Update message should include the Credit Card # and Exp Date
    - And it should send a Tilde for the four ECP data elements [DDA, R/T, Account Type, and Payment Delivery Method]

10.6.15 Retrieving a Profile

At any given time, there may be a need to retrieve the data on an existing Profile. There is a very simple Retrieve Profile transaction type available to perform this action.

10.6.16 Deleting a Profile

Any Profile can be deleted at any time with a Delete Profile transaction.

At this stage, even after a Profile has been deleted a Customer Profile Reference Number still may not be used again.

10.6.17 Using Profiles

One of the key transaction types is using a Profile to process a transaction. This is accomplished by:

- Inserting the Customer Reference Number into one of the existing message types.
- All data that can be pre-populated by the Profile will be.
- Any relevant data, such as CVD for eCommerce transactions, should be included in the request.
- The transaction request should be completed per the normal spec in terms of which tags are mandatory. If the data exists in the Profile and the Tag is mandatory, simply null fill the tag.
- This means that the correct XML Message type should be used based on the card type of the profile. For example, if the card type of a Profile is a credit card, then the base credit card message structure should be used to use the profile. The credit card data, again, would simply be null filled.

10.6.18 Overriding Profile Data

Almost any data set in the Profile can be overridden during a transaction that is using the Profile. For instance, if a Profile included a fixed amount, but a particular transaction was for a different amount, it could be changed for that transaction by including a specific amount in the use profile request.

The one exception to the override rule is the payment type, such as Credit Card versus ECP, cannot be overridden. If the Payment is different, then the Profile should either be Updated [if that change is permanent] or not used [if it is temporary].

By the same token, if the payment type is the same, but the data is different, it can be overridden on a single transaction, if desired.

Finally, overriding Profile data does not update the profile. If the change is permanent, an Update Profile request should be sent in.

10.6.19 Expiration Date

One scenario to take into consideration when overriding data has to do with the usage of expiration dates. As defined in the spec, for a Salem customer, a null Expiration date is one mechanism to submit transactions for authorization when the expiration date is unknown. By the same token, Expiration Date is a required tag for credit card transactions and must be present when Using Profile. And it must be null filled to not override the expiration date that might be set in the profile.

As such, if an Expiration date is saved in a Profile, and the desire is to override that, but submit nothing because the new expiration date is unknown, the transaction should use one of the other two supported mechanisms for supporting unknown expirations dates:

- Send four spaces - <Exp>   </Exp>
- Zero fill the XML Element - <Exp>0000</Exp>

10.6.20 Transaction Types

Profiles may be used on Authorization, Authorization-Capture, Prior Authorizations, and Refund transactions. It is not functional (or necessary) for Voids, Mark for Capture, or End of Day transactions.

10.6.21 Mark for Capture Transactions

While the correct Credit Card or Switch / Solo Mark for Capture ['MFC'] transaction never really required the card number from a business perspective, the Orbital Gateway DTD mandated the presence of the XML element. So many customers are either sending the actual credit card number, a dummy number or null filling this tag. Effective with the release of Profile Management, the Orbital Gateway has corrected this design and no longer mandates this tag to be a part of the MFC request. This was done to ensure that a customer using a Profile could complete a MFC without having to include this tag.

10.6.22 Industry Types

All the Industry Types are supported by the Orbital Gateway [eCommerce, Mail Order, and Recurring] are all supported within Profiles. The transaction in question should set the correct tags as identified in the XML Message Specification.

10.6.23 Currencies

All currencies supported by the Orbital Gateway are supported as a part of Profiles. Simply set the correct currency code and exponent on the transaction being processed.

10.6.24 Managed Billing Profiles

Managed Billing enables merchants to configure Profiles so that Chase Paymentech will automatically run transactions in the future. Managed Billing supports Recurring, Installment, and Deferred Billings. Note that a merchant account can only be configured for one type of Managed Billing at a time.

10.6.25 Recurring Billings

Recurring billings bill cardholders for future payments according to a predefined schedule. Recurring billings can be configured to happen on a weekly, monthly, or yearly basis. Attributes such as Start Date, End Date, and Recurring Frequency must be set so that the Managed Billing system can schedule payments.

Also, since Chase Paymentech will be initiating the future transaction instead of the merchant, a choice regarding Order ID generation must be made.

10.6.26 Installment Billings

Installment billings are handled exactly like Recurring, except that the End trigger is configured using the "MBRecurringMaxBillings" tag. However, this behavior is not enforced by the Orbital Gateway.

Deferred Billings

Deferred Billings are one-time billings that occur on a future date. The key element that needs to be set for a Deferred billing is the Deferred Billing date.

As above, since Chase Paymentech will be initiating the future transaction instead of the merchant, a choice regarding Order ID generation must be made.

10.6.27 Setting a Managed Billing Frequency Pattern

Frequency patterns for Managed Billing are configured using a subset of a standard "CRON" expression, comprised of 3 fields separated by a white space.

Managed Billing Frequency Pattern fields:

Field Name	**Allowed Values** (not case-sensitive)	**Allowed Special Chars** (not case-sensitive)
Day-of-Month	1 – 31	, - * ? L W
Month	1 – 12 or JAN-DEC	, - *
Day-of-Week	1 - 7 or SUN-SAT	, - * ? L #

Frequency Pattern Special Characters:

The asterisk (*) character is used to specify all values. For example, "*" in the Month field means "every month."

The question mark (?) character is allowed for the Day-of-Month and Day-of-Week fields. It is used to specify "no specific value" for the given field. This is useful when you need to specify something in two of the fields but not the third. See the examples section below for clarification.

The dash (-) character is used to specify ranges. For example, "10-12" in the Month field means "the months October, November, and December."

The comma (,) character is used to specify additional values. For example, "MON,WED,FRI" in the Day-of-Week fields means "the days Monday, Wednesday, and Friday."

The capital "L" character is allowed for the Day-of-Month and Day-of-Week fields. This character is short-hand for "last", but it has a different meaning in each of the two fields. For example, the value "L" in the Day-of-Month field means "the last day of the month"(day 31 for January, day 28 for February on non leap years, etc). If used in the Day-of-Week field by itself, it simply means "7" or "SAT." But if used in the Day-of-Week field after another value, it means "the last xxx day of the month" (for example, "6L" means "the last Friday of the month." When using the "L" option, it is important not to specify lists or ranges of values as you'll get confusing results.

The capital "W" character is allowed for the Day-of-Month field. This character is used to specify the weekday (Monday-Friday) nearest the given day. As an example, if you were to specify "15W" as the value for the Day-of-Month field, the meaning is "the nearest weeked to the 15th of the month." So if the 15th is a Saturday, the billing will occur on Friday the 14th. If the 15th is a Sunday, the billing will occur on Monday the 16th. If the 15th is a Tuesday, then the billing will occur on Tuesday the 15th. However, if you specify "1W" as the value for Day-of-Month", and the first is a Saturday, the billing will occur on Monday the 3rd, as it will not 'jump' over the boundary of a month's days. The "W" character can only be specified when the Day-of-Month is a single day, not a range or list of days.

The "L" and "W" characters can also be combined for the Day-of-Month expression to yield "LW", which translates to "last weekday of the month."

The pound sign (#) character is allowed for the Day-of-Week field. This character is used to specify "the nth" xxx day of the month." For example, the value of "6#3" means "the third Friday of the month" (day 6 = Friday and "#3 = the 3rd one of the month). Other examples: "2#1" means "the first Monday of the month" and "4#5" means "the fifth Wednesday of the month. Note that if you specify "#5" and there are not five occurrences of that day in the given month, no billings will occur that month.

Managed Billing Frequency Pattern examples:

Recurrence Pattern needed	**Corresponding CRON expression** (these are examples only – there are multiple ways to express most patterns) *Note: the double-quotes below will NOT be sent in the actual tags/messages*
Weekly	
Every Wednesday in the month of March	`“? MAR WED” or “? 3 WED” or “? 3 4”`
Every Sunday, June through August	`“? JUN-AUG SUN”`
Every Monday	`“? * MON”`
Monthly	
First day of each month	`“1 * ?”`
First day of every other month (odd months)	`"1 1,3,5,7,9,11 ?"`
First day of every other month (even months)	`“1 2,4,6,8,10,12 ?”`
15th day of every month	`“15 * ?”`
Last day of every month	`“L * ?”`
Last Friday of every month	`“? * 6L” or “? * FRIL”`
Third Friday of every month	`“? * 6#3”`
Nearest weekday to the first of the month	`“1W * ?”`
Last weekday of the month	`“LW * ?”`
Yearly	
1st of January	`“1 JAN ?”`
1st weekday of January	`“1W JAN ?”`
1st of January, every 2 years	
1st of January, every 3 years	
Last day of May, every year	`“L MAY ?” or “L 5 ?”`

10.6.27.1 Retry Logic Usage

Retry Logic, the function that allows transactions to be processed without risk of duplicating them will not be supported for Profile Management transactions. In other words, Adds, Deletes, Retrieves and Updates.

However, if when performing a Profile Management transaction, an unknown result occurs, simply replay that transaction.

- If the prior transaction was a success, the second attempt will simply result in duplicate response, which not cause any harm.
- If the original request was not successful, the second attempt will create the result desired.

While Retry is not supported for Profile Management transactions, there is no harm in placing the Trace-ID values associated with Retry Logic in the MIME-Header of these request items. In these circumstances, the trace value will simply be ignored.

Note: Using Profile during an Authorization, Retry Logic is fully supported as defined the message specification.

10.7 Gift Card

10.7.1 Transaction Types

This section defines all the Gift Card transaction types supported by the Orbital Gateway.

10.7.1.1 Card Activation Transaction Types

- Activate - This transaction is used to activate one individual card for the first time.
  - Merchants processing to the PNS Host can process Prior Activation Transactions by additionally passing the correct prior approval code. If the valid Prior Approval code is not passed, it will be treated as a new Activation request.
  - Salem Merchants attempting to process a Prior Activation will receive an error response.
- Block Activate - Block activation provides for the ability to activate more than one card at a time. The maximum number of cards that can be activated at a time is 100. In a Block Activate request, the card number of the first card in a series is defined, plus the number of additional sequential cards.
  - If the Block Activation fails, none of the cards in the block will be activated. And the first card number that caused the Block Activation failure will be returned in the response.
  - The Virtual Terminal supports the ability to perform a Block Activation of 10,000 in a single request. However, as indicted above the On-Line interface maximum is only 100 cards per request.

- Deactivate - This transaction is for the deactivation of a live card. Passing an amount is not required for this transaction type.
- Reactivate - There are two mechanisms for reactivating a card once it has been deactivated:
  - Reversing the deactivation transaction - this will return the initial balance prior to the deactivation transaction.
  - Or the card can be reactivated. In a reactivation transaction, a dollar amount must be passed for how much the card for which will be reactivated.

  NOTE: The Orbital Gateway supports $0 activation transactions for PNS [BIN 000002].
- Add Value - This transaction type adds value to an active card. If an Add Value is performed on an inactive card, it will both activate the card and perform the add value action as well.
  - Merchants processing to the PNS Host can process Prior Add Value Transactions by additionally passing the correct prior approval code. If the valid Prior Approval code is not passed, it will be treated as a new Add Value request.
  - Salem Merchants attempting to process a Prior Add Value will receive an error response.

10.7.1.2 Purchase and Refund Transactions

- The following transaction types are for purchases and refunds. There are two different transaction combinations available for purchases.

  - Authorization followed by a Redemption Completion – This transaction combination is only valid for Salem based customers.
  - Redemption

These two combinations allow for different purchase processing behavior on Gift Card cards. This section will define how each transaction type functions.

- Authorization - Almost all Gift Card transaction types immediately affect the card balance, meaning add or reduce the funds based on the result. In some circumstances there might be a desire to perform a sale wherein an authorization is performed and the funds not actually moved. One reason for this, for example, might be a deferred shipment of goods.
  - The Authorization transaction does exactly that. It will reduce the "Available to Buy" amount without actually reducing the actual funds.
  - Once the item has been shipped, performing a Redemption Completion can complete the transaction, see below.

- Generally speaking, an authorization will hold the requested funds for seven ("7") days, after which the funds will be available again.
- **As stated above, this transaction type is only supported for Merchants processing through the Salem Platform.**

- Redemption Completion - As stated above, a redemption completion is to complete an authorization. Redemption Completions are performed on the Orbital Gateway processing a Gift Card element and setting the Action code to RedemptionCompletion and passing the transaction reference number.
  - As stated above, this transaction type is only supported for Merchants processing through the Salem Platform.
  - There is one optional behavior when managing Redemption Completions.
- Partial Redemption - Gift Card supports a functionality called Partial Redemption. If for whatever reason when the Redemption Completion is submitted and the amount of the original authorization exceeds the available balance, the merchant has two options on how to treat this transaction, which is managed by submitting the field FlexPartialRedemptionInd.
  - If the available balance on the card is less than the Redemption Completion Amount:
    - The transaction can be declined with no amount redeemed from the card. If this is the desired behavior on a particular transaction, do not submit this field or null fill it.
    - Or the transaction can be approved, with the maximum amount of the Redemption completion fulfilled, even though it is less. The response in this circumstance would include both the Requested amount and the actual redeemed amount. The behavior can be implemented by passing the FlexPartialRedemptionInd field with a value of "Y".

10.7.1.3 Auto-Authorization of the Remainder

- As stated above, when a Redemption Completion is submitted for an amount that is less than original amount, the entire original balance on hold is removed and the new amount redeemed from the card. If the transaction was a split shipment; however, the merchant might want the card to maintain a hold on the balance for the remaining un-redeemed amount. To re-establish this hold on the funds, a new authorization needs to be submitted.

- This new authorization can either be a whole new request submitted by the merchant.
  - Or, when performing the Redemption Completion, the Element <FlexAutoAuthInd> can be passed in the Mark for Capture request with a value of Y. When this happens, if the initial Redemption Completion is successful, the Orbital Gateway will automatically generate a new Authorization Request for the amount that represents the un-redeemed amount from the initial Authorization.
  - Return a single response identifying the result of the Redemption Completion and, if it was successful, the new authorization request. If there is a new authorization, the following aspects of the response are important to understand:
    - There will only be one set of Current and Previous Card Balances returned and those values will be reflective of the result of both actions.
    - If the new authorization request is successful, the new authorization will be a distinct transaction with a distinct Transaction Reference Number from the original Authorization. All the relevant details of the second request will be returned in the response.

- Redemption - As opposed to an Authorization followed by a Redemption Completion, Redemption is the mechanism to perform an immediate redemption. Once completed, Redemptions can only be reversed.
  - Merchants processing to the PNS Host can process Prior Redemption Transactions by additionally passing the correct prior approval code. If the valid Prior Approval code is not passed, it will be treated as a new Redemption request.
  - Salem Merchants attempting to process a Prior Redemption will receive an error response.
- Refund - This transaction type is for initiating refunds to a Gift Card card.

10.7.1.4 Reversals

All transaction types, excluding Balance Inquiries, can be reversed, thus returning a transaction to the state it was in prior to the action being reversed. There are two restrictions as it relates to processing Reversals:

- For Salem customers, the reversal must be performed within seven ["7"] days of the original transaction.
- For PNS based customers, the reversal must be performed before the next batch close. Batch closes for Gift Card are usually automatically at 5:00am EST, regardless of what the Auto-settle time is on the Gateway.
- Another action has not occurred that no longer makes the reversal possible.
    - For example, a card, that is activated, can no longer have the activation reversed once a transaction has been processed. The card can only be deactivated at that point, if desired.
- A reversal is accomplished by simply processing a 'Void' FlexAction type using the merchant information and the Transaction reference number of the original. This is true of all reversal transaction types. Unlike other transactions, the Complex Type 'Reversal' is not used for Gift Card Void/reversal actions
- Using the same Gift Card element for the request and setting the FlexAction to "Reversal" and passing the transaction reference number of the original accomplish a reversal. This is true of all reversal transaction types.

The response on a Reversal will provide the same information as any other response [Current Balance, Previous Balance, Response Codes, etc]. In addition, it will identify specifically what transaction type is being reversed, such as "Reversal – Redemption" in the <FlexAction> tag. See section 10.7.2 for more information on responses.

10.7.1.5 Balance Inquiry

This transaction will perform a Balance Inquiry.

10.7.2 Responses

The basic authorization response for all Gift Card transactions is the same. In other words, all responses will be returned with the same base minimum data elements. The transactions types that will include more information are:

- Bloc Activations [if they fail]
- Redemption Completions with the Partial Redemption Flag
- Redemption Completions with Auto-Authorization behavior

10.7.3 Settlement

Since transactions affect the balance of a card real time, Gift Card transactions are not affected by the End of Day process options. Instead transactions will automatically fall into one of two buckets when viewed through the virtual terminal:

- Open Gift Card items: This will include all un-settled activity:
    - Authorizations that been redeemed [Redemption Completion]
    - Declined transactions
    - Errors
- In the Review section of the VT, all Redeemed items will be viewable. These items will be grouped on a daily basis on the same timing as the Gift Card System reports activity, which is 5am – 5am.

10.7.4 Reporting

All true Gift Card reporting is available from the Gift Card system, including Re$ource On Line. Any questions about the available reports should be directed to your account manager.

The Virtual Terminal should not be used for Gift Card reconciliation.

10.8 Bill Me Later

10.8.1 *Introduction*

Bill Me Later is an innovative and secure payment solution for card-not-present merchants. The Bill Me Later method of payment is a non-plastic issued credit vehicle that manages the consumer payment function by providing a transactional credit decision in lieu of the standard predetermined credit line and associated authorization process. Bill Me Later allows consumers to make online/mail order purchases without inputting credit card information.

10.8.2 *How it Works*

Using proprietary credit scoring and fraud detection capabilities, I^{4}Commerce screens each Bill Me Later transaction in real time, instantly decisioning all Bill Me Later requests made by customers.

10.8.3 *Processing Requirements*

Merchants must contract with I^{4}Commerce for acceptance of Bill Me Later.

The Orbital Gateway enforces the following data requirements for sale [authorization, authorization-capture] transaction types:

10.8.3.1 Required:

- Account Number
- Bill To Address [see standard AVS elements]
- Ship To Address [see AVSDest elements]
- Shipping Cost [BMLShippingCost]
- Terms and Conditions Version [BMLTNCVersion]
- Customer Registration Date [BMLCustomerRegistrationDate]
- Customer Type Flag [BMLCustomerTypeFlag]
- Item Category [BMLItemCategory]
- Customer Birth Date [BMLCustomerBirthDate]
- Customer Social Security Number [BMLCustomerSSN]
- Product Delivery Method [BMLProductDeliveryType]

10.8.3.2 Optional:

- Customer Source IP [BMLCustomerIP]
- Customer Email [BMLCustomerEmail]
- Pre-approval Invitation Number [BMLPreapprovalInvitationNum]
- Promotional Code [BMLMerchantPromotionalCode]
- Customer Annual Income [BMLCustomerAnnualIncome]
- Customer Resident Status [BMLCustomerResidenceStatus]
- Customer Checking Account [BMLCustomerCheckingAccount]
- Customer Saving Account [BMLCustomerSavingsAccount]

NOTE: Please contact your I^{4}Commerce Bill Me Later Integration Analyst during the requirements definition phase prior to development to determine required fields.

10.8.4 Currencies

- US Currency

10.8.5 Other

10.8.5.1 Virtual Terminal

- All of the functionality supported through this interface for Bill Me Later is additionally available through the Orbital Gateway Virtual Terminal.

10.8.5.2 Platforms

- The Orbital Gateway only supports the Bill Me Later method of payment through the Salem host platform [BIN 000001]. This method of payment is not supported on the PNS host [BIN 00002].

10.9 PINLess Debit

10.9.1 Introduction

- Customers use their ATM/Debit cards to pay for goods or services rather than cash, check, or credit card. It is more commonly known as Debit Bill Payment. This is a debit transaction where neither the magnetic stripe contents nor the PIN is part of the authorization message.
- Debit transactions are always authorized on a “real time” basis with the actual authorization resulting in the debit of the customer’s bank account. These transactions must still be captured and settled to Chase Paymentech to support funding, reporting and associated reconciliation.
- PINLess Debit is currently only supported by the three largest debit networks: Star, NYCE, and Pulse
- The debit network rules for PINLess debit programs are strict and the networks that support these transactions must approve the merchant prior to their accepting PINLess debit transactions. As a result, PINLess debit processing is only available to merchants in selected industries, specifically utilities, telephone companies, cable TV providers, some insurance companies, government entities, and financial institutions. This list could change so check with your account manager for availability rules.
- The customer must initiate these transactions via the Web or IVR and the merchant assumes 100% liability for these payments.
- Please refer to the *Debit Bill Payment User Manual* for card association and debit network regulations.

10.9.2 Processing Requirements

As a result of the specific processing rules associated with PINLess Debit, the Orbital Gateway enforces specific behavior as it relates to PINLess Debit:

- Only allow Authorization-Capture transaction types are allowed. This means:
  - No Auth Only [future fulfillment] transactions.
  - No Mark for Capture
  - No Splits
  - No voids.
  - No refunds.
- All Merchant ID’s [Transaction Divisions] enabled for PINLess Debit must have auto-settle enabled.
- PINLess Debit BIN Ranges are every dynamic. Chase Paymentech makes PINLess Debit BIN files available on a regular basis to determine which cards are eligible to participate on each transaction.

Additionally, the Orbital Gateway imports and stores the most up to date PINLess Card ranges. If a card is submitted as PINLess Debit [as identified by the required card mnemonic] and it is not in an eligible card range, a ProcStatus error code of 9717 ["PINLess Debit: Card Number Not Eligible for PINLess Debit Processing"]. Optionally, a merchant can retrieve their own Debit BIN files. Please discuss with your account manager the available options.

- A new Industry type of IVR [IV] has been added but is only allowed PINLess Debit method of payment.
- Data set required to submit a PINLess Debit transaction is:
  - IndustryType – IVR or eCommerce Only
  - MessageType – AC only
  - BIN – 000001 only
  - Merchant ID
  - TerminalID – 001 only
  - CardBrand – DP
  - Account Number
  - Expiration Date - null
  - Currency Code – 840 [US Dollar] only
  - Currency Exponent – 2 only
  - Order ID
  - Amount
  - Biller Reference Number
- Approved PINLess Debit transactions may return a Blank or N/A authorization code.

10.9.3 Supported Currencies

- U.S. Currency

10.9.4 Virtual Terminal

- The Orbital Virtual will display and report PINLess debit transactions, but it will not support:

- Creation of new PINLess debit transactions are not supported because PINLess Debit transactions are only supported via IVR or ECommerce channels.
- Adjustments of existing PINLess debit transactions.

10.10 Soft Descriptor Record

10.10.1 Introduction

The Soft Descriptor Records are used to define the merchant name/product description that will appear on the consumer's statement. It allows the merchant greater flexibility in describing the consumer's purchase.

It is subject to issuer discretion whether this descriptor will be displayed on the cardholder statement.

10.10.2 Soft Descriptor Support

Support for Soft Descriptors is not globally available to all customers using the Orbital Gateway:

- Salem - The Orbital Gateway supports Soft Descriptors into the Salem Host. However:
  - Prior Risk Department approval is required.
  - And the Merchant ID/Terminal ID must be enabled for Soft Descriptors on the Orbital Gateway.
  - Please contact you Chase Paymentech Representative.
- Tampa - The Orbital Gateway supports Soft Descriptors into the Tampa Host. However:
  - It is only supported for Chase Paymentech Canada customers.
  - And the Merchant ID/Terminal ID must be enabled for Soft Descriptors on the Orbital Gateway.

- The behavior is different from that of the Salem Interface. Please refer to the Tampa development section to understand how it works.
- Please contact you Chase Paymentech Representative.

10.10.3 SALEM SUPPORT

10.10.3.1 Rules and Guidelines – Credit Card

The description in the *merchant name field* should be what is most recognizable to the cardholder. It should consist of the company name and /or trade name combined with some type of description of the product or service that was purchased.

Chase Paymentech will not generate or segregate reports by the Soft Descriptor. If the Merchant wishes to see Salem reports segregated by product, the Merchant must set up specific reporting divisions and deposit those transactions under that division number.

For those Merchants who need to roll up several merchant names under one corporation, please contact your Chase Paymentech Representative for details on the use and regulation of the Soft Descriptors.

- The Merchant Name can be one of 3 different Lengths:
  - 3 bytes
  - 7 bytes
  - 12 bytes
- And addition Product Description can be appended based on the length of the Merchant Name, such that they are a combined length of 21 bytes. In other words, the options are:
  - 18 bytes
  - 14 bytes
  - 9 bytes

Notes:

- The City Field allows the merchant to identify the business location or provides the cardholder with a Customer Service Phone Number or URL. This is a requirement to qualify for Visa’s lowest Direct Marketing interchange rate.
- If the merchant submits a backslash (\) in the merchant descriptor, then it will be converted into a hyphen (-) on the cardholder statement. If the merchant submits a question mark (?) in the merchant descriptor, then it will be converted into a space on the cardholder statement.
- There are certain American Express card types/programs that ignore the descriptors sent using Soft Descriptors. The Optima card is one of these types. The merchant should contact their American Express representative for more details.
- Non-eCommerce sent with a URL will not qualify for the best interchange.
- For MasterCard MOTO (Transaction Type 1) and Recurring (Transaction Type 2), if the City/Phone field at the division level is not a Customer Service Phone Number, then a Customer Service Phone Number must be populated in the Merchant city/Customer Phone Number field or the transaction will reject with Response Reason Code BP (“Customer Service Phone reqd. on Tran Types 1(MO/TO) and 2(Recurring). MC Only”).
- The Orbital Gateway will apply the asterisks [‘*’] in the necessary locations. Please do not add these to the request.

10.10.3.2 Rules and Guidelines – ECP

The Automated Clearing House (ACH) uses two fields to describe the transaction to the consumer. The Merchant Name, 15 bytes, will always appear on the consumer’s statement, and the Entry Description, 10 bytes, will appear on the consumer’s statement a majority of the time. Both are required fields.

Chase Paymentech recommends that the Merchant Name be used for the Doing Business As (DBA) description and the Company Entry field be used for the product description.

When utilizing the Soft Descriptor for ECP transactions, both the Merchant Name and the Entry Description are mandatory.

10.10.4 A couple of different examples of Soft Descriptors are:

- 3 Byte Merchant Descriptor with Phone #
  ```
  <SDMerchantName>XYZ</SDMerchantName>
  <SDProductDescription>PAYMENT1OF3</SDProductDescription>
  <SDMerchantCity/>
  <SDMerchantPhone>888-888-8888</SDMerchantPhone>
  <SDMerchantURL/>
  <SDMerchantEmail/>
  ```

- 12 Byte Merchant Descriptor with Email
  ```
  <SDMerchantName>XYZCOMPANY</SDMerchantName>
  <SDProductDescription>PYMT1OF3</SDProductDescription>
  <SDMerchantCity/>
  <SDMerchantPhone/>
  <SDMerchantURL/>
  <SDMerchantEmail>suppt@xyz.com</SDMerchantEmail>
  ```

- ECP
  ```
  <SDMerchantName>XYZCOMPANY12345</SDMerchantName>
  <SDProductDescription>PRODUCT123</SDProductDescription>
  <SDMerchantCity/>
  <SDMerchantPhone/>
  <SDMerchantURL/>
  <SDMerchantEmail/>
  ```

10.10.5 TAMPA SUPPORT

10.10.5.1 Rules and Guidelines

Again, the support for Soft Descriptors via the Tampa Host is only for customers processing through Chase Paymentech Canada.

Unlike Salem, the only value that gets passed on the Cardholder statement is the Merchant Name field. And for these customers, it is a maximum of 25 bytes of data.

All other Soft Descriptor fields can optionally be sent, but will not be submitted to the settlement host and will not display on the cardholder statement.

10.10.6 Soft Descriptor Examples

```
<SoftDescriptor>
        <SDMerchantName>XYZ PAYMENT1OF3</SDMerchantName>
        <SDProductDescription/>
        <SDMerchantCity/>
        <SDMerchantPhone/>
        <SDMerchantURL/>
        <SDMerchantEmail/>
</SoftDescriptor>
```

10.11 Purchasing Card

10.11.1 Introduction

The Orbital Gateway supports the processing of procurement cards by fully supporting the enhanced data required by Visa and MasterCard for both Level II and Level III data. Additionally, for American Express, the Orbital Gateway for Salem customers supports Level II and enhanced TAA.

Purchasing Cards with Level III data are typically used in a business-to-business environment providing and collecting funds for outstanding invoices. Merchants have the ability to collect their funds in conjunction with the settlement of their credit card transactions and still provide their customer with the necessary line item detail, thus providing a cleaner process for both the merchant and their customer.

10.11.2 Edit Checks

The Orbital Gateway performs edit checks on incoming data to ensure necessary information is present. In the event necessary information is missing from a transaction, the transaction will result in an error [see list of error codes in the ProcStatus listing in Appendix A]. Data fields that are edited by Chase Paymentech have been marked as “Conditionally Required” in the Transaction Management section of this document. Additionally, there some special edit checks specific to each host described below.

10.11.2.1 PNS

There are two key mathematical data validations specific to PNS processing for Purchasing Card 3 Processing.

- The amount field [pCard3Dtllinetot] of every line item must equal the Unit Cost [pCard3DtlUOM] multiplied by the quantity [pCard3DtlQty] less any discounts [pCard3DtlDisc]. If it does not, then this transaction will receive an error.
- Additionally, the sum of all the Line Item totals [pCard3Dtllinetot] cannot exceed the transaction amount [amount] submitted for an order.

10.11.2.2 Salem

There is no mathematical validation for Purchasing Card II or III for Salem customers.

10.11.3 BIN Ranges

The BIN range assigned by the card associations can identify purchasing cards. BIN ranges are subject to change at the discretion of the card associations.

10.11.4 Processing

Purchasing Card II or III data can be sent either in the original auth [via an Auth or Auth-Capture]. Or it can be appended to the authorization via the Mark for Capture request if it was not originally supplied in the authorization request.

There are different rules for adding and adjusting Purchasing Card data via the Mark for Capture based on whether it is simply level II data or if it is level III data.

Purchasing Card Level II can be sent on Sales and Refunds, but Purchasing Card Level III data can only be sent on Sales.

10.11.5 MFC Adjustment of Level II Data

When processing Purchasing Card level II data, the additional data can be included in either the Auth (and later adjusted) or added altogether via the Mark for capture transaction.

The following scenarios describe the optional behavior:

- If the Purchasing data is submitted on the Authorization.
    - No purchasing Card data is submitted on a Mark for Capture [whether full or partial amount]. The gateway will submit at settlement the purchasing card data presented on the authorization.
    - Purchasing Card Data is submitted on the Mark for Capture [MFC] and the MFC is for:
        - The full amount: The Purchasing Card data submitted on the MFC will override the data submitted on the Auth.
        - Partial Amount: Where the amount of the MFC is less than the auth and a split transaction is generated, whatever data is submitted on the first mark for capture will be used on all splits. If each split should have different data, then each MFC should include the relevant purchasing card data. But that is not required.
- If the data is not submitted on the auth, then it must be included on the MFC for it to be submitted at Settlement.
    - Where the amount submitted on the MFC is equal to the Auth, the transactions is complete and that data will be used.
    - Where the amount is less, just as described above, and a split transaction is generated, whatever data is submitted on the first mark for capture will be used on all splits.

10.11.5.1 Purchasing Card III

Just as with Purchasing Card Level II, when processing Level III data the additional data can included in either the Auth (and later adjusted) or added altogether via the Mark for capture transaction. However because of the Tampa based Purchasing Card Validation rules [see edits above], there is different behavior in terms of what can be done when adjusting the purchasing data via a Mark for Capture. The following scenarios describe the optional behavior:

- If the Purchasing data is submitted on the Authorization.
    - No purchasing Card data is submitted on a Mark for Capture [whether full or partial amount]. The gateway will submit at settlement the purchasing card data presented on the authorization.
    - Purchasing Card Data is submitted on the Mark for Capture [MFC] and the MFC is for:
        - The full amount: The Purchasing Card data submitted on the MFC will override the data submitted on the Auth, as long as the amended data is still consistent with the data validation rules [PNS/Tampa customers only] otherwise the MFC capture request will fail.
        - Partial Amount: Where the amount of the MFC is less than the auth and a split transaction is generated, whatever data is submitted on the first mark for capture will be used on all splits. If each split should have different data, then each MFC should include the relevant purchasing card data. But that is not required, as long as the amended data is still consistent with the data validation rules [PNS/Tampa customers only] other wise the MFC will fail.
- If the data is not submitted on the auth, then it must be included on the MFC for it to be submitted at Settlement.
    - Where the amount submitted on the MFC is equal to the Auth, the transactions is complete and that data will be used. Again, for PNS/Tampa customers, the purchasing card level 3 data must pass the data validation rules or the MFC request will fail.

Where the amount is less, just as described above, and a split transaction is generated, whatever data is submitted on the first mark for capture will only be used for that transaction [for Salem and Tampa customers alike]. All subsequent MFC requests [again both Salem and Tampa] must include the purchasing card level 3 data relevant to that component [and for PNS/Tampa customers it must additionally match the amount, based on the edits, of the MFC.]

10.11.5.2 Virtual Terminal

All of the functionality supported through this interface for Purchase Card 2 and 3 is additionally available through the Orbital Gateway Virtual Terminal.

10.12 European Direct Debit

10.12.1 Overview

European Direct Debit [EU DD] is a popular method of payment for merchants marketing in Europe. While any merchant may want to accept direct debit payments, it is most important and cost effective for those merchants collecting recurring payments. Unlike in the US, many EU customers prefer to pay for recurring services by direct debit to their bank accounts. This is especially true in Germany where almost 40% of all electronic payments are made by direct debit.

10.12.2 How it works

In Europe, each country operates its own direct debit network. Merchants wishing to accept direct debit throughout Europe would face the requirement to establish banking relationships and technical integration for each country in which they wish to market. Chase Paymentech Solutions has created a single technical interface for direct debit processing for multiple countries.

10.12.3 Processing Requirements:

Merchants must contract with Chase Paymentech Solutions for acceptance of European Direct Debit. The Merchant Descriptor is defined on the vendor's system. Sending the Merchant Descriptor record will not alter the descriptor on the accountholder's statement.

The purpose of this document is intended to outline how a developer can code to take advantage of this method of payment within the Orbital Gateway, both in terms of the message layout and the business rules.

10.12.4 Virtual Terminal

All of the functionality supported through this interface for European Direct Debit is additionally available through the Orbital Gateway Virtual Terminal.

10.12.5 Platforms

The Orbital Gateway only supports the European Direct Debit method of payment through the Salem host platform [BIN 000001]. This method of payment is not supported on the PNS host [BIN 00002].

10.13 Electronic Check Processing (e-check)

10.13.1 Overview

Automated Clearing House used for the exchange of paperless entries. Direct deposit utilizes the ACH system. Direct payment is one of the fastest growing payment vehicles today. The ACH system routes transactions to the appropriate endpoint based on the transit routing number assigned financial institutions by the Federal Reserve System.

10.13.2 How it works

ECP Check Verification is done through the SCAN database. SCAN is a leading check verification and guarantee service. The database lists the routing transit and account number of checks that are currently being collected or known to be closed-for-cause. The processing of e-check is fast, efficient, has a lower cost per transaction than credit card transactions with no paper to handle.

Electronic checks are accepted in US and Canadian currencies and useful for web initiated debits, recurring debits, mail order/telephone ordering, just to name a few of its benefits.

10.13.3 Processing Requirements

The merchant must comply with "**Regulation E Electronic Fund Transfer Act"** when using the ACH System. The consumer must give the merchant authorization allowing the merchant to directly debit the consumer's checking account through the ACH Network. This will establish a relationship between the merchant and the consumer and the agreement must be in "writing" or "similarly authenticated" with the date and signature of consumer.

10.14 Switch/Solo

10.14.1 Overview

Chase Paymentech offers processing of Great Britain's Switch and Solo debit cards for US merchants. Whether you sell online, through a catalog or both - you can now offer a card that is held by 70% of UK consumers.

10.14.2 How it works

SWITCH is a United Kingdom only, private label debit card. In terms of number of cardholders, it is the single largest card issued in the UK. Being a UK only card, all transactions are processed in British Pounds Sterling. This allows the Merchant to perform currency exchanges when they determine the rate is favorable. Chase Paymentech supports real time authorization processing or batch authorization file submissions. Being a private label debit card, the merchant incurs no discount fees. The only costs are modest transaction processing fees.

10.14.3 Processing Requirements

The merchant must execute a Chase Paymentech International processing agreement and set up a British Pounds Sterling bank account. Transactions of all currency and card types may be transmitted within a single submission file.

Note: A Switch card number is 19 digits long. Merchant order processing systems will need to support the additional digits.

11 Transaction Types

The Java SDK supports the use of the following XML transaction templates specified in the tables in this section. The tables indicate the field names, default values, and if a field value is required.

The developer must populate the fields that **are Mandatory**; whereas, fields that have default values may optionally be altered.

Example:

```
request = new Request(RequestIF.NEW_ORDER_TRANSACTION);

request.setFieldValue ("MerchantID", "700000000417");
request.setFieldValue ("BIN", "000002");
request.setFieldValue ("OrderID", "122003SA");
```

Note: Request types are ***case sensitive***.

Required [Y / N / C] Key:

Y = Yes, Required
N = No, Optional
C = Conditionally Required

11.1 New Order Transactions – Request Elements

This transaction can be used to construct any transaction that's initiates a new order with exception with Gift Card transactions. To construct this XML transaction type in the Java SDK, pass the constant value "NewOrder" (statically defined in RequestIF.NEW_ORDER_TRANSACTION constant) in the Request object's constructor.

RequestIF request = new Request (RequestIF.NEW_ORDER_TRANSACTION);

NOTE: Due to the flexibility of this transaction, many of the fields are optional. Many individual transactions types (for instance, constructing a credit card authorization only transaction) will require several fields marked as optional. In most cases, the Chase Paymentech Gateway will return descriptive error messages when required fields are missing. For additional help in constructing your transactions, please refer to the supplied samples or call Chase Paymentech Support for assistance.

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
Request	**Required XML Parent Tag**	M	N/A	N/A
NewOrder	**XML tag that defines the transaction as a New Order request**	M	N/A	N/A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
OrbitalConnectionUsername	**Orbital Connection Username set up on Orbital Gateway** - Provide Username that is associated to this MID - Required if merchant is not set up for IP based authentication Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Not case sensitive	C	32	A
OrbitalConnectionPassword	**Orbital Connection Password used in conjunction with Orbital Username** - Provide Password associated with Connection Username Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Password is case sensitive and must match with what is stored on Orbital Gateway exactly	C	32	A
IndustryType	**Defines the Industry type of the transaction:** MO – Mail Order transaction RC – Recurring Payment EC– eCommerce transaction IV – IVR [PINLess Debit Only]	M	2	A
MessageType	**Defines the transaction New Order Transaction Type:** A – Authorization request AC – Authorization and Mark for Capture FC – Force-Capture request R – Refund request	M	2	A
BIN	**Transaction Routing Definition:** Assigned by Chase Paymentech 000001 – Salem 000002 – PNS	M	6	N
MerchantID	**Gateway merchant account number assigned by Chase Paymentech:** This account number will match that of your host platform BIN 000001 – 6 digit Salem Division Number BIN 000002 – 12 digit PNS Merchant ID	M	12	N

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech:** All Salem Terminal IDs at present must be '001'. PNS Terminal ID's can be from '001' – '999'. Most are '001'.	M	3	N
CardBrand	**Defines the Card Type / Brand for the Transaction:** See attached table. Required for: SW – Switch / Solo ED – European Direct Debit EC – Electronic Check BL – Bill Me Later DP – PINLess Debit [Generic Value Used in Requests]	C	2	A
AccountNum	**Card Number identifying the customer.** Should be null for electronic check processing and Profile Transactions Can be null for Refund transactions, provided that the TxRefNum field is filled appropriately. For Bill Me Later transactions, the account number field should be populated with either the customer's Bill Me Later account number or a Bill Me Later Bank Identification Number (BIN) followed by ten zeros (dummy account number) – such as 5049900000000000. The consumer's 16- byte Bill Me Later account number will be returned on all approved transactions.	M	19	N

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
Exp	**Card Expiration Date:** Format: MMYY Mandatory for all card types except ECP, European Direct Debit, Bill Me Later and PINLess Debit; except as defined below: - Can be null for Refund transactions, provided that the TxRefNum field is filled appropriately. - Salem [BIN 000001] allows a blank to be submitted when no known EXP date exists. Please discuss this feature with your certification analyst before implementing. There are three valid mechanisms for submitting a 'Blank' expiration date using Orbital to the Salem Host. They are: - Null fill this XML element - <Exp/> - Send four spaces - <Exp> </Exp> - Zero fill the XML Element - <Exp>0000</Exp>	C	4	N
CurrencyCode	**Defines the transaction currency:** The gateway using the standard ISO defines currency codes. Keys: - Bin 000002 only supports the US Dollar ['840'] and Canadian Dollar ['124']. - See Appendix for complete list of supported currencies.	M	3	N
CurrencyExponent	**Defines the transactions currency exponent:** See Appendix for values.	M	6	N
CardSecValInd	**Supported by Visa and Discover only:** 1 - Value is Present 2 - Value on card but illegible 9 - Cardholder states data not available NOTE: If the transaction is not a Visa or Discover transaction, null fill this attribute or do not submit the attribute at all.	C	1	N

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
CardSecVal	**Card Verification Number** - Visa CVV2 – 3 bytes - MasterCard CVC2 – 3 bytes - American Express CID – 4 bytes - Discover CID – 3 bytes NOTE: It is against regulations to store this value.	O	4	N
DebitCardIssueNum	**Switch – Solo incremental counter for lost or replacement cards.** - Tag Mandatory for Switch – Solo. - An incremental counter of either 1 or 2 characters defined by the issuing bank. If a card is lost, the bank issues a replacement card with the issue number being increased by one. If the card displays '01', submit '01', NOT '1'. If the card displays '1', submit '1'. NOTES: - The DebitCardStartDate field should be submitted only when the card does not have an Issue Number. If the card displays ONLY a Start Date and no Issue Number, the DebitCardStartDate field should contain a value and the DebitCardIssueNum field must be left blank [null filled]. If the card displays both a Start Date and an Issue Number, the DebitCardStartDate field should be left blank [null filled] and the DebitCardIssueNum field must be populated.	C	2	N
DebitCardStartDate	**Switch – Solo card start date:** - Format 'MMYY' - Tag Mandatory for Switch – Solo. NOTE: The card start date should be submitted only when the card does not have an Issue number. If the card displays ONLY a Start Date and no Issue Number, the DebitCardStartDate field must be blank [null filled]. If the card displays both a Start Date and an Issue Number, this field should be left blank and the DebitCardIssueNum field must be populated.	C	4	N

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
BCRtNum	**Bank routing and transit number for the customer.** Conditionally required for Electronic Check processing. NOTE: - All US Bank Routing Numbers are 9 digits - All Canadian Bank Routing Numbers are 8 Digits	C	9	N
CheckDDA	**Customer DDA account number** Conditionally required for Electronic Check processing.	C	17	A
BankAccountType	**Defines the deposit account type:** Conditionally required for Electronic Check processing. C – Consumer Checking [US or Canadian] S – Consumer Savings [US Only] X – Commercial Checking [US Only]	C	1	A
ECPAuthMethod	**Defines the ECP Authorization method:** Code used to identify the method used by consumers to authorize debits to their accounts. If no value submitted, we will default this value. Valid Values: W – Written I – Internet (Web) – DEFAULT T – Telephone	O	1	A
BankPmtDelv	**Defines the ECP payment delivery method:** This field indicates the preferred manner to deposit the transaction. Conditionally required for Electronic Check processing. B – Best Possible Method [US Only] Chase Paymentech utilizes the method that best fits the situation. If the RDFI is not an ACH participant, a facsimile draft will be created. This should be the default value for this field. A – ACH [US or Canadian] Deposit the transaction by ACH only. If the RDFI is not an ACH participant, the transaction is rejected.	C	1	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
AVSzip	**Cardholder Billing Address Zip Code:** - All AVS Requests must minimally include the 5-digit Zip Code. - If sending Zip Code + 4, please separate with a '-' Required for Bill Me Later sale transactions	C	10	A
AVSaddress1	**Cardholder Billing Address line 1** Should not include any of the following characters types: % | ^ \ / Required for Bill Me Later sale transactions	C	30	A
AVSaddress2	**Cardholder Billing Address Line 2** Should not include any of the following characters types: % | ^ \ /	O	30	A
AVScity	**Cardholder Billing City** Should not include any of the following characters types: % | ^ \ / Required for Bill Me Later sale transactions	C	20	A
AVSstate	**Cardholder Billing State** Should not include any of the following characters types: % | ^ \ / Required for Bill Me Later sale transactions	C	2	A

AVSphoneNum	**Cardholder Billing Phone Number**
	AAAEEENNNNXXXX, where
	AAA = Area Code
	EEE = Exchange
	NNNN = Number
	XXXX = Extension
	Required for Bill Me Later sale transactions
		C	14	A

AVSname	**Cardholder Billing Name** Required for Bill Me Later sale transactions and all Electronic Check Transactions.	C	30	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
AVScountryCode	**Cardholder Billing Address Country Code:** Required if processing a U.K. based Address. Valid values: US – United States CA – Canada GB – Great Britain UK – United Kingdom " " – Blank for all other countries Required for Bill Me Later sale transactions	C	2	A
AVSDestzip	**Bill Me Later Cardholder Destination Address Zip Code:** - All AVS Requests must minimally include the 5-digit Zip Code. - If sending Zip Code + 4, please separate with a '-' Required for Bill Me Later sale transactions	C	10	A
AVSDestaddress1	**Bill Me Later Cardholder Destination Address line 1** Should not include any of the following characters types: % | ^ \ / Required for Bill Me Later sale transactions	C	30	A
AVSDestaddress2	**Bill Me Later Cardholder Destination Address Line 2** Should not include any of the following characters types: % | ^ \ / Optional for Bill Me Later Transactions	O	30	A
AVSDestcity	**Bill Me Later Cardholder Destination Billing City** Should not include any of the following characters types: % | ^ \ / Required for Bill Me Later sale transactions	C	20	A
AVSDeststate	**Bill Me Later Cardholder Destination Billing State** Should not include any of the following characters types: % | ^ \ / Required for Bill Me Later sale transactions	C	2	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N

AVSDestphoneNum	**Bill Me Later Cardholder Destination Phone Number**
	AAAEEENNNNXXXX, where
	AAA = Area Code
	EEE = Exchange
	NNNN = Number
	XXXX = Extension
	Optional for Bill Me Later sale transactions
		O	14	A

AVSDestname	**Bill Me Later Cardholder Destination Billing Name** Required for Bill Me Later sale transactions	C	30	A
AVSDestcountryCode	**Bill Me Later Cardholder Destination Address Country Code:** Required if processing a U.K. based Address. Valid values: US – United States CA – Canada GB – Great Britain UK – United Kingdom " " – Blank for all other countries Required for Bill Me Later sale transactions	C	2	A
CustomerProfileFromOrderInd	**Customer Profile Number generation Options:** A – Auto Generate the CustomerRefNum S – Use CustomerRefNum Element When performing an action other than a Customer Profile Action Type = Create, insert "empty" as it will be ignored. But this Attribute is mandatory and cannot be null filled for Customer Profiles.	M	5	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
CustomerRefNum	**Sets the Customer Reference Number that will be used to utilize a Customer Profile on all future Orders.** Mandatory if Customer Profile Action Type = Read, Update, or Delete Or Create and the Customer Profile Number generation option = S [Use CustomerRefNum Element] Keys: - If CustomerProfileFromOrderInd = A, the Customer Reference Number will be defined by the Gateway and any value passed in this element will be ignored. - Given that this value can be the same as the Order Number, the valid characters for this field follow the same convention as the Order ID element. Those valid characters are: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-,$@& and a space character. However, the space character cannot be the leading character. This value may not be changed through a Profile Update action.	C	22	A
CustomerProfileOrderOverrideInd	**Defines if any Order Data can be pre-populated from the Customer Reference Number [CustomerRefNum]:** Mandatory if Customer Profile Action Type = Create Optional if Customer Profile Action Type = Update **NO** No mapping to order data **OI** Use <*CustomerRefNum*> for <*OrderID*> and <*ECOrderNum*> or <*MailOrderNum*> **OD** Use <*CustomerReferNum*> for <*Comments*> **OA** Use <*CustomerRefNum*> for <*OrderID*> and <*Comments*>	C	2	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
Status	**Profile Status flag** This field is used to set the status of a Customer Profile. Valid values: A – Active I – Inactive MS – Manual Suspend	C	Var	A
authenticationECIInd	**Defines transaction type:** Conditionally required for Verified by Visa and MasterCard SecureCode transactions 5 – Verified by Visa/MasterCard SecureCode – Authenticated Transaction 6 – Verified by Visa/MasterCard SecureCode – Attempted Authentication	C	1	N
CAVV	**Cardholder Authentication Verification Value use in Verified by Visa Transactions (CAVV):** - This number must be Base 64 Encoded. - Cryptographic value derived with an algorithm that applies the Issuer's private key to the combination of the Cardholder account number, the Transaction Identifier (XID), and other data.	C	40	A
XID	**Transaction ID used in Verified by Visa Transactions (XID):** ▪ This number must be Base 64 Encoded. ▪ Unique tracking number set by the Merchant and sent to the Issuer Authentication/Service in the Authentication Request message [Optional]	O	40	A
PriorAuthID	**Defines the transaction type as a Prior Auth:** When this value is present, the request is considered a force authorization. No on-line authorization will be generated to the Host systems.	O	6	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
OrderID	**Merchant Defined Order Number:** Field defined and supplied by the auth originator, and echoed back in response. The first 8 characters should be unique for each transaction. Valid Characters: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN OPQRSTUVWXYZ0123456789-,$@ and a space character. However, the space character cannot be the leading character. PINLess Debit transactions can only use upper and lower case alpha [A-Z, a-z] and numeric [0-9] characters and NO special characters.	M	22	A
Amount	**Transaction Amount**: Keys: - Implied Decimal including those currencies that are a zero exponent. For example, both $100.00 (an exponent of '2') and 100 Yen (an exponent of '0') should be sent as <Amount>10000</Amount>.	M	12	N
Comments	**Free form comments:** Merchant can fill in this field and the info will be stored with the transaction details. For PNS customers, this field will populate the Customer Defined Data field, which is displayed in Resource Online.	O	64	A
ShippingRef	**Shipping Tracking Reference Number.** Merchant can fill in this field and the info will be stored with the transaction details.	O	40	A
TaxInd	**Defines the tax type:** Conditionally required for Purchasing Card Level II Data 0 – Not provided 1 – Included 2 – Non-Taxable	C	1	N
Tax	**Tax Amount for the purchase:** - Conditionally required for Purchasing Card Level II Data - Implied decimal including those currencies that are a zero exponent.	C	12	N

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
AMEXTranAdvAddn1	**Amex Purchasing Card Data – Transaction Advice Addendum #1** The TAA Record is used to further identify the purchase that is associated with the charge to the cardholder. It is also used in Purchasing / Procurement card transactions to provide specific details about the transaction to the cardholder for tracking purposes. TAA's should be as concise as possible. A TAA of "Merchandise" for example, would not be acceptable. Salem Only / Conditionally required for Amex Purchasing Card Data	C	40	A
AMEXTranAdvAddn2	**Amex Purchasing Card Data – Transaction Advice Addendum #2** Salem Only / Conditionally required for Amex Purchasing Card Data	C	40	A
AMEXTranAdvAddn3	**Amex Purchasing Card Data – Transaction Advice Addendum #3** Salem Only / Conditionally required for Amex Purchasing Card Data	C	40	A
AMEXTranAdvAddn4	**Amex Purchasing Card Data – Transaction Advice Addendum #4** Salem Only / Conditionally required for Amex Purchasing Card Data	C	40	A
AAV	**Accountholder Authentication Value for MasterCard Secure Code:** Conditionally required for MasterCard SecureCode transactions. This number must be Base 64 Encoded. Unique transaction token generated by the issuer and presented to the merchant each time a cardholder conducts an electronic transaction using MasterCard SecureCode. AAV incorporates elements specific to the transaction and effectively binds the cardholder to a transaction at a merchant for a given sales amount.	C	32	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
SDMerchantName	**Soft Descriptor Merchant Name** Conditionally required for Soft Descriptors The Merchant Name field should be what is most recognizable to the cardholder [Company name or trade name]. The actual length of this field is conditionally tied to Host and the Size of the <SDProductDescription> field used. Salem: - CREDIT – Three options, which conditionally affects the SDProductDescription [see below]: o Max 3 bytes o Max 7 bytes o Max 12 bytes - ECP: o Max 15 bytes PNS: - Max 25 bytes.	C	25	A
SDProductDescription	**Soft Descriptor Product Description** Conditionally required for Soft Descriptors. Provides an accurate product description Salem: - CREDIT: o If SDMerchantName = 3 bytes – then Max = 18 bytes o If SDMerchantName = 7 bytes – then Max = 14 bytes o If SDMerchantName = 12 bytes – then Max = 9 bytes - ECP: o 10 bytes Max PNS: - This field will **not** show on Cardholder statements for PNS Merchants.	C	18	A
SDMerchantCity	**Soft Descriptor Merchant City** Tag conditionally required for Soft Descriptors. Merchant City for Retail. Field required but should be null filled if any Soft Descriptor data is submitted.	C	13	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
SDMerchantPhone	**Soft Descriptor Merchant Phone** Tag conditionally required for Soft Descriptors Only one of the location Soft Descriptor records should be sent [meaning Phone, URL, or Email]. This field will **not** show on Cardholder statements for PNS Merchants. Valid Formats: - NNN-NNN-NNNN - NNN-AAAAAAA NOTE: For MasterCard MOTO (Transaction Type 1) and Recurring (Transaction Type 2), if the City/Phone field at the division level is not a Customer Service Phone Number, then a Customer Service Phone Number must be populated in the Merchant city/Customer Phone Number field or the transaction will reject with Response Reason Code BP (Missing Customer Service Phone).	C	12	A
SDMerchantURL	**Soft Descriptor Merchant URL** Tag conditionally required for Soft Descriptors [can be null filled] Only one of the location Soft Descriptor records should include data [meaning Phone, URL, or Email]. This field will **not** show on Cardholder statements for PNS Merchants.	C	13	A
SDMerchantEmail	**Soft Descriptor Merchant Email** Tag conditionally required for Soft Descriptors [can be null filled] Only one of the location Soft Descriptor records should include data [meaning Phone, URL, or Email]. This field will **not** show on Cardholder statements for PNS Merchants.	C	13	A
RecurringInd	**Recurring indicator:** This tag is conditionally required for Merchants that are: - Located in Canada - Processing on BIN 000002 - Industry Type = Recurring The valid values are: RF – First Recurring Transaction. RS – Subsequent Recurring Transactions.	C	2	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
EUDDCountryCode	**European Direct Debit Country Code** Customer's Country Code. The following is the list of valid country codes. AT – Austria BE – Belgium DE – Germany FR – France GB – United Kingdom NL – Netherlands	C	2	A
EUDDBankSortCode	**European Direct Debit Bank Sort Code** Customer's Bank Sort code. Mandatory for the following Country Codes: AT – Austria DE – Germany FR – France GB – United Kingdom	C	10	A
EUDDRibCode	**European Direct Debit RIB** Bank Account checksum – used in France only	C	2	A
BMLCustomerIP	**Customer's IP Address** Optional for Bill Me Later sale transactions	O	45	A
BMLCustomerEmail	**Customer Email Address** Optional for Bill Me Later sale transactions	O	50	A
BMLShippingCost	**Total Shipping Cost of Consumers Order** Mandatory for Bill Me Later sale transactions	C	8	N
BMLTNCVersion	**Terms and Conditions Number** The Terms and Conditions Number to which the consumer agreed. Mandatory for Bill Me Later sale transactions	C	5	N
BMLCustomerRegistrationDate	**Customer Registration Date** The date a customer registered with the merchant. Mandatory for Bill Me Later sale transactions	C	8	N
BMLCustomerTypeFlag	**Customer Type Flag** New or Existing Customer to the Merchant [not Bill Me Later] Valid Values: N – New E – Existing Optional for Bill Me Later sale transactions	O	2	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
BMLItemCategory	**Item Category** Product Description Code assigned by I^{4}Commerce Mandatory for Bill Me Later sale transactions	C	4	N
BMLPreapprovalInvitationNum	**Pre-Approval Invitation Number** Indicates whether the consumer has been pre-approved for Bill Me Later or not. ▪ Pre-approval from credit bureau should include the 16-digit pre-approval number. This will allow the pre-approval to be matched with the first consumer order. ▪ Internal pre-approval should include the leftmost digit as a 1. ▪ No pre-approval should include all zeros or be blank filled. Optional for Bill Me Later sale transactions	O	16	A
BMLMerchantPromotionalCode	**Merchant Promotional Code** Optional for Bill Me Later sale transactions	O	4	A
BMLCustomerBirthDate	**Customer Date of Birth** YYYYMMDD Format Mandatory for Bill Me Later sale transactions	C	8	N
BMLCustomerSSN	**Customer Social Security Number** Either the full 9 digit or last 4 digits of the customer's Social Security Number. Mandatory for Bill Me Later sale transactions	C	9	N
BMLCustomerAnnualIncome	**Gross Household Annual Income** Implied Decimal. For example, $100,000.00 should be sent as: <BMLCustomerAnnualIncome>10000000</BMLCustomerAnnualIncome> Optional for Bill Me Later sale transactions	O	10	N
BMLCustomerResidenceStatus	**Customer Residence Status** Valid Values: O – Own R – Rent X – Other Optional for Bill Me Later sale transactions	O	1	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
BMLCustomerCheckingAccount	**Customer Checking Account Indicator** Valid Values: Y – Yes, customer has a checking account N – No, customer does not have a checking account Optional for Bill Me Later sale transactions	O	1	A
BMLCustomerSavingsAccount	**Customer Savings Account Indicator** Valid Values: Y – Yes, customer has a savings account N – No, customer does not have a savings account Optional for Bill Me Later sale transactions	O	1	A
BMLProductDeliveryType	**Delivery Type Indicator** Valid Values: CNC – Cash and Carry DIG – Digital Goods PHY – Physical Delivery Required SVC – Service TBD – To Be Determined Optional for Bill Me Later sale transactions	C	3	A
BillerReferenceNumber	**Biller Reference Number [PINLess Debit Only]** Reference Number the Biller (merchant) uses on their system to identify this customer. Conditionally required for PINLess Debit	C	25	A
MBType	**Managed Billing Type** Indicates the type of Managed Billing the merchant is participating in. The value submitted must be in agreement with the type of Managed Billing the merchant is configured for at Chase Paymentech. Valid Values: R – Recurring D – Deferred This field serves to notify the Orbital system that the transaction is a Managed Billing transaction. If this field is not sent in conjunction with a Managed Billing transaction, any other Managed Billing fields will be ignored.	C	1	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
MBOrderIdGenerationMethod	**Managed Billing Order ID generation method** This value is used to set the method that Orbital will use to generate the Order ID for any Managed Billing transactions. This field does NOT influence the Order ID for stand-alone transactions initiated by the merchant, VT transactions, etc. Valid values: IO – Use Customer Reference Number (Profile ID); this value is made up of the capital letters ‘I’ and ‘O’, no numerals. DI – Dynamically generate; this value is made up of the capital letters ‘D’ and ‘I’, no numerals.	C	2	A
MBRecurringStartDate	**Managed Billing Recurring Start Date** Defines the future date that Orbital will begin a recurring billing cycle to the associated Profile. To allow the Managed Billing engine to properly calculate and schedule all billings, this date must be at least one day after the request date (a recurring billing cycle can never begin on the date that the request message is sent to the Orbital system). Format is MMDDYYYY	C	8	N
MBRecurringEndDate	**Managed Billing Recurring End Date** Defines the future date that Orbital will end a recurring billing cycle to the associated Profile. Format is MMDDYYYY This is the first of three possible recurring end triggers. Only one end trigger can be submitted per request message.	C	8	N
MBRecurringNoEndDateFlag	**Managed Billing ‘No End Date’ Indicator** Valid values: Y – this value will schedule recurring transactions for an infinite amount of time. If “Y”, this value will override MBRecurringEndDate (above), even if it is populated with an end date. N (or blank) – Orbital will use MBRecurringEndDate(above) to define the recurring end date This is the second of three possible recurring end triggers. Only one end trigger can be submitted per request message.	C	1	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
MBRecurringMaxBillings	**Managed Billing Max Number of Billings** This value will define the maximum number of billings that will be allowed for a recurring billing cycle. Valid values: 1 – 999999 This is the third of three possible recurring end triggers. Only one end trigger can be submitted per request message.	C	6	N
MBRecurringFrequency	**Managed Billing Recurring Frequency Pattern** This pattern is a subset of a standard CRON expression, comprised of 3 fields separated by white space. Field Allowed Values Allowed Special Chars 1)Day-of-month 1-31 , - * ? / L W 2)Month 1-12 or JAN-DEC , - * / 3)Day-of-week 1-7 or SUN-SAT , - * ? / L # For a full discussion of these three fields, the usage of the special characters, and multiple example values, see the Profiles and Managed Billing section above (prior to the message layouts).	C	Var	A
MBDeferredBillDate	**Managed Billing Deferred billing date** Defines the future date that Orbital will trigger a one-time billing to the associated Profile. This date must be at least one day after the request date (a deferred billing can never take place on the date that the request message is sent to the Orbital system). Format is MMDDYYYY	C	8	N

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
TxRefNum	**Gateway transaction reference number:** A unique value is assigned by the Gateway for each transaction. The only time this field will be used in a New Order is to complete a Return (Refund, Credit) transaction on the card used in the original transaction from which the TxRefNum was issued. If this field is submitted with a Return, the card number and expiration date are no longer required. If no amount is sent, the original amount will be refunded. If an amount is sent, it must be equal to or less than the original amount. If this field is submitted with any other type of New Order transaction other than a Return, it will be ignored.	O	40	A
PCOrderNum	**PO number or Order number from customer:** Required for Purchasing Card Level 2 Data	C	17	A
PCDestZip	**Shipping Destination Zip code for the purchase:** - Required for Purchasing Card Level 2 and Level 3 Data - For Zip Code + 4 please separate with '-'. Required for best Interchange rate and cannot be all zeros or nines.	C	10	A
PCDestName	**Amex Purchasing Card Data – Cardholder Ship To: Name** Salem Only / Required for Amex Purchasing Card Data	C	30	A
PCDestAddress1	**Amex Purchasing Card Data - Cardholder Ship To: Address line 1** Salem Only / Required for Amex Purchasing Card Data	C	30	A
PCDestAddress2	**Amex Purchasing Card Data - Cardholder Ship To: Address line 2** Salem Only / Required for Amex Purchasing Card Data	C	30	A
PCDestCity	**Amex Purchasing Card Data – Cardholder Ship TO: City** Salem Only / Required for Amex Purchasing Card Data	C	20	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
PCDestState	**Amex Purchasing Card Data – Cardholder Ship TO: State** Salem Only / Required for Amex Purchasing Card Data	C	2	A
PC3FreightAmt	**Purchase Card Level 3 freight amount for shipment** Total freight or shipping and handling charges. Implied decimal.	O	12	N
PC3DutyAmt	**Purchase Card Level 3 Duty Amount for Shipment** Total charges for any import and/or export duties included in this transaction. Implied decimal.	O	12	N
PC3DestCountryCd	**Purchase Card Level 3 Destination Country Code** The ISO-assigned code of the country to which the goods are shipped. Conditionally required for all Purchasing Card 3 transactions. If no value is submitted, it will be default to the United States [USA]. Required for Purchasing Card 3. See Table in Appendix A	C	3	A
PC3ShipFromZip	**Purchase Card Level 3 Ship from Zip** The zip/postal code of the location from which the goods are shipped. Required for best Interchange rate and cannot be all zeros or nines.	C	10	A
PC3DiscAmt	**Purchase Card Level 3 Discount Amount from Order** The total amount of discount applied to the transaction by the merchant. Used by the merchant when a price break is given on an entire transaction rather than on unit prices. Typically, this is shown as a credit on a detailed invoice. Implied decimal. Optional for Visa only. Should not be sent for MasterCard.	O	12	N

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
PC3VATtaxAmt	**Purchase Card Level 3 Total Amount of VAT or other tax** The total amount of VAT or other tax included in this transaction. Implied decimal Optional for Visa only. Should not be sent for MasterCard.	O	12	N
PC3VATtaxRate	**Purchase Card Level 3 Rate of VAT or other tax** The total amount of VAT or other tax (expressed in percentage terms) for this line item. 2 decimal implied. Example: 0001 = 1% Optional for Visa only. Should not be sent for MasterCard.	O	4	N
PC3AltTaxID	**Purchase Card Level 3 Alternate Tax ID** Tax ID number for the alternate tax associated with this transaction. Optional for MasterCard only. However, required if an amount is sent in PC3AltTaxAmt. Should not be sent for Visa.	O	15	N
PC3AltTaxAmt	**Purchase Card Level 3 Alternate Tax Amount** Total Amount of alternate tax associated with this transaction. Implied decimal. Optional for MasterCard only. However, required if an amount is sent in PC3AltTaxID. Should not be sent for Visa.	O	9	N
PC3LineItemCount	**Purchase Card Level 3 Number of Line Items** The number of Purchasing Card 3 Line Item Detail items included with this transaction. The maximum number of line items is 98. At least 1 line item must be included to submit Purchasing Card 3 data. Required for Purchasing Card 3.	C	2	N
PC3LineItemArray	**Purchase Card Level 3 Detail Header** Required parent tag for Purchasing Card 3 Line Item Detail components.	C	N/A	N/A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
PC3LineItem	**Parent XML Tag for Individual Purchase Card Level 3 Line Item Details** This XML Tag is the parent for each Line item detail included in this transaction. It should be repeated for each item.	C	N/A	N/A
PC3DtlIndex	**Purchase Card Level 3 Line Item Index** The sequential number [1 – 98] of the Line Item Details included with this transaction. Required for Purchasing Card 3.	C	2	N
PC3DtlDesc	**Purchase Card level 3 Line Item Detail Element – Description** Text description of the item purchased. Cannot be all zeros Required for Purchasing Card 3.	C	35	A
PC3DtlProdCd	**Purchase Card level 3 Line Item Detail Element – Product Code** Product code of the item purchased. Cannot be all zeros Required for Purchasing Card 3.	C	12	A
PC3DtlQty	**Purchase Card level 3 Line Item Detail Element – Number of Units** Number of units of the item purchased. Cannot be all zeros Required for Purchasing Card 3. Implied decimal of 4.	C	13	N
PC3DtlUOM	**Purchase Card level 3 Line Item Detail Element – Unit of Measurement** The unit of measure, or unit of measure code used for this line item. Required for Purchasing Card 3. See Table in Appendix A	C	3	A
PC3DtlTaxAmt	**Purchase Card level 3 Line Item Detail Element – Tax Amount** The tax amount for this item Implied decimal Required for Purchasing Card 3	C	13	N

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
PC3DtlTaxRate	**Purchase Card level 3 Line Item Detail Element – Tax Rate** Tax rate applied for this item. Implied decimal of 3 as a percentage. Ex: Interest rate of 6.25% should be sent as'06250'. Required for Purchasing Card 3	C	5	N
PC3Dtllinetot	**Purchase Card level 3 Line Item Detail Element – Line Item Total** For PNS customers: - This field must equal the Unit Cost [PC3DtlUnitCost] multiplied by the quantity [PC3DtlQty] less any discounts [PC3DtlDisc]. If it does not, then this transaction will receive an error. - Additionally, the sum of all the Line Item totals [i.e., the sum of all these fields] cannot exceed the transaction amount [<Amount>] submitted for this order. Implied decimal Cannot be all zeros for either PNS or Salem. Required for Purchasing Card 3	C	13	N
PC3DtlDisc	**Purchase Card level 3 Line Item Detail Element – Discount Amount for Line Item** Amount of the discount applied to the line item Implied decimal Required for Purchasing Card 3	C	13	N
PC3DtlCommCd	**Purchase Card level 3 Line Item Detail Element – Commodity Code for Line Item** The commodity code used to classify the item purchased. Required for Visa. Should not be sent for MasterCard.	C	12	N
PC3DtlUnitCost	**Purchase Card level 3 Line Item Detail Element – Unit Cost of Item Purchased** Unit Cost of the unit purchased Implied decimal of 4. Required for Purchasing Card 3	C	13	N

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
PC3DtlGrossNet	**Purchase Card level 3 Line Item Detail Element – Gross/Net Indicator** Indicates whether tax amount is included in the item amount Valid values: Y = Item amount includes tax amount N = Item amount does not include tax amount Required for Purchasing Card 3	C	1	A
PC3DtlTaxType	**Purchase Card level 3 Line Item Detail Element – Type of Tax Being Applied** Type of tax being applied	O	4	A
PC3DtlDiscInd	**Purchase Card level 3 Line Item Detail Element – Discount Indicator** Indicates whether the amount if discounted Valid values: Y = Amount is discounted N = Amount is not discounted If value = Y, and Discount Amount Field [PC3DiscAmt] is Blank or Zero Filled, Chase Paymentech will change this field indicator to 'N' before sending the data. Optional for MasterCard only. Should not be sent for Visa.	O	1	A
PC3DtlDebitInd	**Purchase Card level 3 Line Item Detail Element – Item Debit/Credit Indicator** Valid Values: D = Item extended amount is a debit. C = Item extended amount is a credit. Required for Purchasing Card 3 for PNS [BIN 00002] Merchants	C	1	A

11.2 New Order Response Elements

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
Response	**Required XML Parent Tag**	M	N/A	N/A
NewOrderResp	**XML tag that defines the transaction as a New Order Response**	M	N/A	N/A
IndustryType	**Defines the Industry type of the transaction:** This tag will return null results	M	2	A
MessageType	**Defines the transaction New Order Transaction Type:** Echoes the Message Type passed in the request	M	2	A
MerchantID	**Gateway merchant account number assigned by Chase Paymentech Solutions:** Echoes the account number passed in the request	M	12	N
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech Solutions:** Echoes the Terminal ID passed in the request	M	3	N
CardBrand	**Defines the Card Type / Brand for the Transaction:** Echoes the Card Type/Brand passed in the request, except: ▪ If no CardBrand was sent in the request [when optional], such as Visa / MasterCard, the specific Card Brand mnemonic is returned. ▪ For PINLess Debit transactions, the request Card Brand is DP [which is a generic PINLess mnemonic. However the response Card Brand will be one of the three supported PINLess Debit Card Brands: o NP - NYCE PINless Debit o PP - Pulse PINless Debit o SP - Star PINless Debit	M	2	A
AccountNum	**Account Number** Value is conditionally returned for approved Bill Me Later transactions. Other methods of payment will never return the card number.	M	19	N
OrderID	**Merchant Defined Order Number:** Echoes the Order Number passed in the request	M	22	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
TxRefNum	**Gateway transaction reference number:** A unique value for each transaction, which is required to adjust any transaction in the gateway [such as Mark for Capture or void].	M	40	A
TxRefIdx	**Gateway transaction index:** Used to identify the unique components of transactions adjusted more than one time. Required on for void transactions, not for Mark for Captures.	M	4	A
ProcStatus	**Process Status:** The first element that should be checked to determine the result of a request. It is the only element that is returned in all response scenarios. It identifies whether transactions have successfully passed all of the Gateway edit checks. 0 – Success All other values constitute an error condition. See appendix for definition of those error values.	M	6	A
ApprovalStatus	**Approval Status:** Conditional on Process Status returning a '0' or successful response. If so, approval status identifies the result of the authorization request to the host system. 0 – Decline 1 – Approved 2 – Message/System Error	C	1	N
RespCode	**Response Code:** Normalized authorization response code issued by the host system [Salem / PNS], which identifies an approval ('00') or the reason for a decline or error. See appendix for values.	C	2	A
AVSRespCode	**Address verification request response:** - See appendix for values - Conditional on AVS request being sent.	M	2	A
CVV2RespCode	**Card verification value request response:** - See appendix for values - Conditional on card verification request being sent	M	1	A

XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
AuthCode	**Issuer approval Code:** Unique transactional level code issued by the bank or service establishment for approvals. PINLess Debit transactions could return blanks or N/A	C	6	A
RecurringAdviceCd	**Recurring Payment Advice Code [MasterCard Only]** 01=New account information available. Obtain new account information 02=Try again later. Recycle transaction in 72 hours. 03=Do not try to obtain another form of payment.	M	2	N
CAVVRespCode	**Response code to VbV Requests**	M	1	A
StatusMsg	**Text message associated with ProcStatus value.**	C	Var	A
RespMsg	**Messages associated with RespCode.**	C	80	A
HostRespCode	**Actual host response code:** Exact response sent by host authorization system [non-normalized by the gateway]. For those systems that have already coded to the Salem / PNS authorization response values, they are available via this tag.	C	3	A
HostAVSRespCode	**Actual host address verification response code:** Exact address verification response sent by host authorization system [non-normalized by the gateway]. For those systems that have already coded to the Salem / PNS authorization response values, they are available via this tag.	C	2	A
HostCVV2RespCode	**Actual host card verification response code:** Exact card verification response sent by host authorization system [non-normalized by the gateway]. For those systems that have already coded to the Salem / PNS authorization response values, they are available still via this tag.	C	1	A
CustomerRefNum	**The Customer Reference Number** If Customer Profile Action Type = Create and If CustomerProfileFromOrderInd = S, this field will echo the Customer Reference Number sent in the Profile Request.	M	22	A
CustomerName	**Customer Billing Name** Value from the Request Returned.	M	30	A

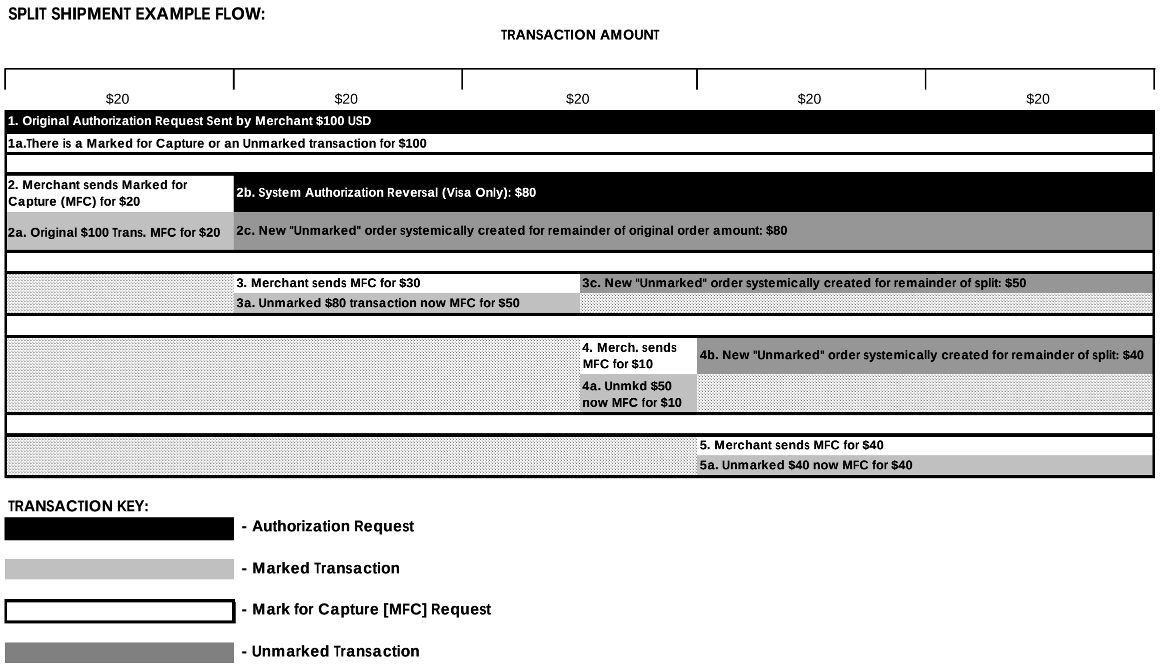
XML Name	Description	Req. M/C/O	Max. Char	Field Type A/N
ProfileProcStatus	**Result Status of Profile Management:** Communicates the success or failure of a Profile Management request. 0 – Success >0 – An error condition, see Appendix A for definition of the specific Profile Management error values.	M	6	A
CustomerProfileMessage	**Verbose Text Description associated with ProfileProcStatus**	M	Var	A
BillerReferenceNumber	**Biller Reference Number [PINLess Debit Only]** Echoed for Request	C	25	A
MBStatus	**Managed Billing Status**	C	Var	A
RespTime	**Time the transaction was processed by gateway:** Format – 'hh24miss'	M	6	N

11.3 Mark For Capture Transactions – Request Elements

This transaction is used to send a complete or partial capture transaction for an existing authorization transaction. To construct this XML transaction type in the Java SDK, pass the constant value “MFC” (statically defined in RequestIF. MARK_FOR_CAPTURE_TRANSACTION constant) in the Request object’s constructor.

RequestIF request = new Request (RequestIF.MARK_FOR_CAPTURE_TRANSACTION);

The table below identifies how Mark for Captures function when the Amount Captured is less than the initial Authorization – i.e. a Split transaction.


SPLIT SHIPMENT EXAMPLE FLOW:
TRANSACTION AMOUNT
$20
$20
$20
$20
$20
1. Original Authorization Request Sent by Merchant $100 USD
1a.There is a Marked for Capture or an Unmarked transaction for $100
2. Merchant sends Marked for Capture (MFC) for $20
2b. System Authorization Reversal (Visa Only): $80
2a. Original $100 Trans. MFC for $20
2c. New "Unmarked" order systemically created for remainder of original order amount: $80
3. Merchant sends MFC for $30
3c. New "Unmarked" order systemically created for remainder of split: $50
3a. Unmarked $80 transaction now MFC for $50
4. Merch. sends MFC for $10
4b. New "Unmarked" order systemically created for remainder of split: $40
4a. Unmkd $50 now MFC for $10
5. Merchant sends MFC for $40
5a. Unmarked $40 now MFC for $40
TRANSACTION KEY:
- Authorization Request
- Marked Transaction
- Mark for Capture [MFC] Request
- Unmarked Transaction

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
OrbitalConnectionUsername	**Orbital Connection Username set up on Orbital Gateway** - Provide Username that is associated to this MID - Required if merchant is not set up for IP based authentication Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Not case sensitive		32A	C

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
OrbitalConnectionPassword	**Orbital Connection Password used in conjunction with Orbital Username** - Provide Password associated with Connection Username Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Password is case sensitive and must match with what is stored on Orbital Gateway exactly		32A	C
OrderID	**Merchant Defined Order Number:** Must use the same OrderID as the original request	None	22 A	Y
Amount	**Transaction Amount**: Keys: - Amount being captured. It can be less than or equal to the original authorization. Anything less than will create a split transaction. - Implied decimal including those currencies that are a zero exponent. For example, both $100.00 (an exponent of '2') and 100 Yen (an exponent of '0') should be sent as <Amount>10000</Amount>. See table for min/max amount for each currency type.	None	12 N	N
TaxInd	**Defines the tax type:** Required for Purchasing Card Level II and Level III Data 0 – Not provided 1 – Included 2 – Non-Taxable	None	1 N	C
Tax	**Tax Amount for the purchase:** - Required for Purchasing Card Level II and Level III Data Implied decimal including those currencies that are a zero exponent.	0	12 A	C
BIN	**Transaction Routing Definition:** Assigned by Chase Paymentech 000001 – Salem 000002 – Tampa	None	6 A	Y

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
MerchantID	**Gateway merchant account number assigned by Chase Paymentech:** This account number will match that of your host platform BIN 000001 – 6 digit Salem Division Number BIN 000002 – 12 digit Tampa Merchant ID	None	15 A	Y
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech:** All Salem Terminal IDs at present must be '001'. PNS Terminal ID's can be from '001' – '999'. Most are '001'.	001	3 A	N
TxRefNum	**Gateway transaction reference number:** A unique value for each transaction, which is required to adjust any transaction in the gateway [such as Mark for Capture or void].	None	40 A	Y
PCOrderNum	**PO number or Order number from customer:** Required for Purchasing Card Level II and Level III Data	None	17	C
PCDestZip	**Shipping Destination Zip code for the purchase:** - Required for Purchasing Card Level II Data For Zip Code + 4 please separate with '-'.	None	10	C
PCDestName	**Amex Purchasing Card Data –** Salem Only / Required for Amex Purchasing Card Data	None	30	C
PCDestAddress1	**Amex Purchasing Card Data - Cardholder Ship To: Address line 1** Salem Only / Required for Amex Purchasing Card Data	None	30 A	C
PCDestAddress2	**Amex Purchasing Card Data - Cardholder Ship To: Address line 2** Salem Only / Required for Amex Purchasing Card Data	None	30 A	C
PCDestCity	**Amex Purchasing Card Data – Cardholder Ship TO: City** Salem Only / Required for Amex Purchasing Card Data	None	20 A	C
PCDestState	**Amex Purchasing Card Data – Cardholder Ship TO: State** Salem Only / Required for Amex Purchasing Card Data	None	2 A	C

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
AMEXTranAdvAddn1	**Amex Purchasing Card Data – Transaction Advice Addendum #1** The TAA Record is used to further identify the purchase that is associated with the charge to the cardholder. It is also used in Purchasing / Procurement card transactions to provide specific details about the transaction to the cardholder for tracking purposes. TAA's should be as concise as possible. A TAA of "Merchandise" for example, would not be acceptable. Salem Only / Required for Amex Purchasing Card Data	None	40 A	C
AMEXTranAdvAddn2	**Amex Purchasing Card Data – Transaction Advice Addendum #2** Salem Only / Required for Amex Purchasing Card Data	None	40 A	C
AMEXTranAdvAddn3	**Amex Purchasing Card Data – Transaction Advice Addendum #3** Salem Only / Required for Amex Purchasing Card Data	None	40 A	C
AMEXTranAdvAddn4	**Amex Purchasing Card Data – Transaction Advice Addendum #4** Salem Only / Required for Amex Purchasing Card Data	None	40 A	C
PC3Core	**Visa/MasterCard Purchasing Card III Data** Complex Type	N/A	N/A	C

11.4 Mark for Capture Response Elements

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
Response	**Required XML Parent Tag**	M	N/A	N/A
MarkForCaptureResp	**XML tag that defines the transaction as a New Order response**	M	N/A	N/A
MerchantID	**Gateway merchant account number assigned by Chase Paymentech Solutions:** Echoes the account number passed in the request	M	12	N
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech Solutions:** Echoes the Terminal ID passed in the request	M	3	N
OrderID	**Merchant Defined Order Number:** Echoes the Order Number passed in the request	M	22	A
TxRefNum	**Gateway transaction reference number:** Echoes the Transaction Reference Number passed in the request	M	40	A
TxRefIdx	**Gateway transaction index:** Used to identify the unique components of transactions adjusted more than one time. Required on for void transactions, not for Mark for Captures.	C	4	A
Amount	**Transaction Amount**: Echoes the Amount passed in the request	M	12	N
ProcStatus	**Process Status:** The first element that should be checked to determine the result of a request. It is the only element that is returned in all response scenarios. It identifies whether transactions have successfully passed all of the Gateway edit checks. 0 – Success All other values constitute an error condition. See appendix for definition of those error values.	M	6	A
StatusMsg	**Text message associated with ProcStatus value.**	C	Var	A
RespTime	**Time the transaction was processed by gateway:** Format – 'hh24miss'	M	6	N

11.5 Void (Reverse) Transactions – Request Elements

This transaction is used to send a reversal (or void) transaction for an existing transaction. To construct this XML transaction type in the Java SDK, pass the constant value "Reverse" (statically defined in RequestIF. REVERSE_TRANSACTION constant) in the Request object's constructor.

```
RequestIF request = new Request (RequestIF.REVERSE_TRANSACTION);
```

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
OrbitalConnectionUsername	**Orbital Connection Username set up on Orbital Gateway** - Provide Username that is associated to this MID - Required if merchant is not set up for IP based authentication Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Not case sensitive	None	32A	C
OrbitalConnectionPassword	**Orbital Connection Password used in conjunction with Orbital Username** - Provide Password associated with Connection Username Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Password is case sensitive and must match with what is stored on Orbital Gateway exactly	None	32A	C
TxRefNum	**Gateway transaction reference number:** A unique value for each transaction, which is required to adjust any transaction in the gateway [such as Mark for Capture or void].	None	40 A	Y
TxRefIdx	**Gateway transaction index:** Used to identify the unique components of transactions adjusted more than one time. Required on for void transactions, not for Mark for Captures.	None	4 N	N

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
AdjustedAmt	**Amount for Partial Voids if necessary:** When a specific amount is included with this tag, that amount will be voided [assuming that the amount is not greater than the transaction amount remaining]. This is a mandatory tag that must be null filled for a full Void.	None	12 N	N
OrderID	**Merchant Defined Order Number:** Should use the same OrderID as the original request	None	22 A	Y
BIN	**Transaction Routing Definition:** Assigned by Chase Paymentech 000001 – Salem 000002 – Tampa	None	6 N	Y
MerchantID	**Gateway merchant account number assigned by Chase Paymentech:** This account number will match that of your host platform BIN 000001 – 6 digit Salem Division Number BIN 000002 – 12 digit Tampa Merchant ID	None	15 N	Y
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech:** All Salem Terminal IDs at present must be '001'. PNS Terminal ID's can be from '001' – '999'. Most are '001'.	001	3 N	N
ReversalRetryNumber	**Retry Trace Number from Original Transaction Request** Provide the Retry Trace Number from the transaction that needs to be voided (in the event the Transaction Reference Number is not known).		16N	C
OnlineReversalInd	**Online Reversal Indicator** Indicates whether an authorization reversal or a void is being requested. This value will override the Orbital Gateway setting on the host, if any. Y Authorization Reversal N Void NULL Void		1A	C

11.6 Void (Reverse) – Response Elements

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
Response	**Required XML Parent Tag**	M	N/A	N/A
ReversalResp	**XML tag that defines the transaction as a New Order response**	M	N/A	N/A
MerchantID	**Gateway merchant account number assigned by Chase Paymentech Solutions:** Echoes the account number passed in the request	M	12	N
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech Solutions:** Echoes the Terminal ID passed in the request	M	3	N
OrderID	**Merchant Defined Order Number:** Echoes the Order Number passed in the request	M	22	A
TxRefNum	**Gateway transaction reference number:** Echoes the Transaction Reference Number passed in the request	M	40	A
TxRefIdx	**Gateway transaction index:** Used to identify the unique components of transactions adjusted more than one time. Required on for void transactions, not for Mark for Captures.	C	4	A
OutstandingAmt	**Remaining Non-voided amount for partial Voids**	M	12	N
ProcStatus	**Process Status:** The first element that should be checked to determine the result of a request. It is the only element that is returned in all response scenarios. It identifies whether transactions have successfully passed all of the Gateway edit checks. 0 – Success All other values constitute an error condition. See appendix for definition of those error values.	M	6	A
StatusMsg	**Text message associated with ProcStatus value.**	C	Var	A
RespTime	**Time the transaction was processed by gateway:** Format – 'hh24miss'	M	6	N

11.7 End of Day - Request Elements

This transaction is used to send an end of day (or batch) transaction. To construct this XML transaction type in the Java SDK, pass the constant value "EOD" (statically defined in RequestIF.END_OF_DAY_TRANSACTION constant) in the Request object's constructor.

```
RequestIF request = new Request (RequestIF.END_OF_DAY_TRANSACTION);
```

NOTE: EOD transactions are slightly different from others transaction types. The response from batch request is not indicated by the approval code; it is indicated by the ProcStatus code, which can be determined by the good () method in the response object.

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
OrbitalConnectionUsername	**Orbital Connection Username set up on Orbital Gateway** - Provide Username that is associated to this MID - Required if merchant is not set up for IP based authentication Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Not case sensitive	None	32A	C
OrbitalConnectionPassword	**Orbital Connection Password used in conjunction with Orbital Username** - Provide Password associated with Connection Username Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Password is case sensitive and must match with what is stored on Orbital Gateway exactly	None	32A	C
BIN	**Transaction Routing Definition:** Assigned by Chase Paymentech 000001 – Salem 000002 – Tampa	None	6 N	Y

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
MerchantID	**Gateway merchant account number assigned by Chase Paymentech:** This account number will match that of your host platform BIN 000001 – 6 digit Salem Division Number BIN 000002 – 12 digit Tampa Merchant ID	None	15 N	Y
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech:** All Salem Terminal IDs at present must be '001'. PNS Terminal ID's can be from '001' – '999'. Most are '001'.	001	3 N	N
SettleRejectBin	**Settle Rejected Item BIN** Complex Type	N/A	N/A	C

11.8 End of Day – Response Elements

XML Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
Response	**Required XML Parent Tag**	M	N/A	N/A
EndOfDayResp	**XML tag that defines the transaction as a New Order response**	M	N/A	N/A
MerchantID	**Gateway merchant account number assigned by Chase Paymentech Solutions:** Echoes the account number passed in the request	M	12	N
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech Solutions:** Echoes the Terminal ID passed in the request	M	3	N
BatchSeqNum	**Sequence Number that References a Settlement Batch**	M	32	A
ProcStatus	**Process Status:** The first element that should be checked to determine the result of a request. It is the only element that is returned in all response scenarios. It identifies whether transactions have successfully passed all of the Gateway edit checks. 0 – Success All other values constitute an error condition. See appendix for definition of those error values.	M	6	A
StatusMsg	**Text message associated with ProcStatus value.**	C	Var	A
RespTime	**Time the transaction was processed by gateway:** Format – 'hh24miss'	M	6	N

11.9 Inquiry Request Elements

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
OrbitalConnectionUsername	**Orbital Connection Username set up on Orbital Gateway** - Provide Username that is associated to this MID - Required if merchant is not set up for IP based authentication Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Not case sensitive	C	32	A
OrbitalConnectionPassword	**Orbital Connection Password used in conjunction with Orbital Username** - Provide Password associated with Connection Username Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Password is case sensitive and must match with what is stored on Orbital Gateway exactly	C	32	A
BIN	**Transaction Routing Definition** Assigned by Chase Paymentech 000001 - Salem 000002 - Tampa	M	6	N
MerchantID	**Gateway merchant account number assigned by Chase Paymentech** This account number will match that of your host platform BIN 000001 - 6 digit Salem Division Number BIN 000002 - 12 digit Tampa Merchant ID	M	15	N

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech** All Salem terminal IDs at present must be '001'. PNS terminal IDs can be from '001' to '999'. Most are '001'.	M	3	N
OrderID	**Merchant Defined Order Number** Use the same Order ID as the original request	M	22	A
InquiryRetryNumber	**Retry Trace Number from original request** Provide the Retry Trace Number from the original request in this tag to return the original response. If the original transaction was not processed successfully, the Gateway will return an error message.	M	16	N

11.10 Inquiry Response Elements

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
InquiryResp	**XML tag that defines the transaction as a New Order request**	M	N/A	N/A
IndustryType	**Defines the Industry type of the transaction** This tag will return null results	M	2	A
MessageType	**Defines the transaction New Order Transaction Type** Echoes the Message Type passed in the request	M	2	A
MerchantID	**Gateway merchant account number assigned by Chase Paymentech Solutions** Echoes the account number passed in the request	M	12	N
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech Solutions** Echoes the Terminal ID passed in the request	M	3	N

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
CardBrand	**Defines the Card Type/Brand for the Transaction** Echoes the Card Type/Brand passed in the request, except: - If no CardBrand was sent in the request [when optional], such as Visa / MasterCard, the specific Card Brand mnemonic is returned. - For PINLess Debit transactions, the request Card Brand is DP [which is a generic PINLess mnemonic. However the response Card Brand will be one of the three supported PINLess Debit Card Brands: NP - NYCE PINless Debit PP - Pulse PINless Debit SP - Star PINless Debit	M	2	A
AccountNum	**Card Number identifying the customer** Echoes the Account Number passed in the request	M	19	N
OrderID	**Merchant Defined Order Number** Echoes the Order Number passed in the request	M	22	A
TxRefNum	**Gateway transaction reference number** Echoes the Transaction Reference Number passed in the request	M	40	A
TxRefIdx	**Gateway transaction index** Used to identify the unique components of transactions adjusted more than one time. Required for Void transactions.	M	4	A
ProcStatus	**Process Status** The first element that should be checked to determine the result of a request. It is the only element that is returned in all response scenarios. It identifies whether transactions have successfully passed all of the Gateway edit checks. 0 – Success All other values constitute an error condition. See Appendix A for definition of those error values	M	6	A

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
ApprovalStatus	**Approval Status** Conditional on Process Status returning a '0' or successful response. If so, approval status identifies the result of the authorization request to the host system. 0 - Decline 1 - Approved 2 - Message/System Error	M	1	N
RespCode	**Response Code** Normalized authorization response code issued by the host system [Salem/PNS], which identifies an approval ('00') or the reason for a decline or error See appendix A for values	M	2	A
AVSRespCode	**Address verification request response** - See appendix A for values - Conditional on AVS request being sent	M	2	A
CVV2RespCode	**Card verification value request response** - See appendix for values - Conditional on card verification request being sent	M	1	A
AuthCode	**Issuer approval Code** Unique transactional level code issued by the bank or service establishment for approvals. PINLess Debit transactions could return blanks or N/A.	M	6	A
RecurringAdviceCd	**Recurring Payment Advice Code [MasterCard Only]** 01 - New account information available. Obtain new account information. 02 - Try again later. Recycle transaction in 72 hours. 03 - Do not try to obtain another form of payment	M	2	N
CAVVRespCode	**Response code to VbV Requests**	M	1	A
StatusMsg	**Text message associated with ProcStatus value**	M	Var	A
RespMsg	**Messages associated with RespCode**	M	80	A
HostRespCode	**Actual host response code** Exact response sent by host authorization system [non-normalized by the gateway]. For those systems that have already coded to the Salem/PNS authorization response values, they are available via this tag.	M	3	A

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
HostAVSRespCode	**Actual host address verification response code** Exact address verification response sent by host authorization system [non-normalized by the gateway]. For those systems that have already coded to the Salem/PNS authorization response values, they are available via this tag.	M	2	A
HostCVV2RespCode	**Actual host card verification response code** Exact card verification response sent by host authorization system [non-normalized by the gateway]. For those systems that have already coded to the Salem/PNS authorization response values, they are available still via this tag.	M	1	A
CustomerRefNum	**The Customer Reference Number** If Customer Profile Action Type = Create and CustomerProfileFromOrderInd = S, this field will echo the Customer Reference Number sent in the Profile Request	M	22	A
CustomerName	**Customer Billing Name** Value from the Request Returned	M	30	A
ProfileProcStatus	**Result Status of Profile Management** Communicates the success or failure of a Profile Management request 0 - Success Greater than 0 - An error condition, see Appendix A for definition of the specific Profile Management error values	M	6	A
CustomerProfileMessage	**Verbose Text Description associated with ProfileProcStatus**	M	Var	A
BillerReferenceNumber	**Biller Reference Number [PINLess Debit Only]** Echoed from Request	C	25	A
RespTime	**Time the transaction was processed by gateway** Format – ‘hh24miss’	M	6	N

11.11 Gift Card Request Elements

This transaction is used to send a Gift Card type transaction. To construct this XML transaction type in the Java SDK, pass the constant value "Gift Card" (statically defined in RequestIF.FLEX_CACHE_TRANSACTION constant) in the Request objects's constructor.

RequestIF request = new Request (RequestIF.FLEX_CACHE_TRANSACTION);

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
OrbitalConnectionUsername	**Orbital Connection Username set up on Orbital Gateway** - Provide Username that is associated to this MID - Required if merchant is not set up for IP based authentication Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Not case sensitive		32A	C
OrbitalConnectionPassword	**Orbital Connection Password used in conjunction with Orbital Username** - Provide Password associated with Connection Username Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Password is case sensitive and must match with what is stored on Orbital Gateway exactly		32A	C
BIN	**Transaction Routing Definition:** Assigned by Chase Paymentech 000001 – Salem 000002 – Tampa	None	6 N	Y
MerchantID	**Gateway merchant account number assigned by Chase Paymentech:** This account number will match that of your host platform BIN 000001 – 6 digit Salem Division Number BIN 000002 – 12 digit Tampa Merchant ID	None	15 N	Y
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech:** All Salem Terminal IDs at present must be '001'. PNS Terminal ID's can be from '001' – '999'. Most are '001'.	001	3 N	N

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
AccountNum	**Customer Gift Card Number** Used in all Gift Card Transactions including Block Activations, wherein this defines the first Card Number in a Block Activation Sequence	None	19 N	N
OrderID	**Merchant Defined Order Number:** Field defined and supplied by the auth originator, and echoed back in response.	None	22 A	Y
Amount	**Transaction Amount**: Keys: - Implied decimal including those currencies that are a zero exponent. For example, both $100.00 (an exponent of ‘2’) and 100 Yen (an exponent of ‘0’) should be sent as Amount = 10000 - See table for min/max amount for each currency type. NOTE: Currency and currency code are not passed in the request. It is implied by the Currency setup for the Merchant ID.	None	12 N	N
CardSecVal	**Card Verification Data [CVD] / PIN** While the CVD value can be submitted on any transaction type, the Gift Card Host will only validate the value on the following transaction types: - Authorize - Redemption - Balance Inquiry	None	4 N	N
Comments	**Free form comments:** Merchant can fill in this field and the info will be stored with the transaction details. For Tampa customers, this field will populate the Customer Defined Data field, which is displayed on ROL.	None	64 A	N
ShippingRef	**Shipping Tracking Reference Number.** Merchant can fill in this field and the info will be stored with the transaction details.	None	40 A	N
IndustryType	**Defines the Industry type of the transaction:** MO – Mail Order transaction RC – Recurring Payment EC– eCommerce transaction	EC	2 A	N

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
FlexAutoAuthInd	**Trigger the auto-authorization of the remainder on a Split Redemption Completion ("MFC") transaction.** Y – Auto Authorize the Remainder Do not submit this Element if the desire is to not generate a new auth for the Remainder. This functionality is only available to Salem customers.	N	1 A	N
FlexPartialRedemptionInd	**Trigger to allow an approval for a MFC transaction if the available balance is less than the requested amount.** Y – Approve Redemption Completion Do not submit this Element if the desire is to not allow an Approval on a Redemption Completion unless the full amount on the card is available. This functionality is only available to Salem customers.	N	1 A	N
FlexAction	**Defines the Transaction [or Action] Type:** The following are the values that can be submitted: - Activate - BlockActivate - DeActivate - ReActivate - AddValue - Auth - Redemption - Refund - BalanceInquiry	None	30 A	Y
StartAccountNum	**Defines the first Card Number in a Block Activation Sequence** Should only used when the FlexAction equals BatchActivate and should be used in conjunction with the ActivationCount	None	19	N
ActivationCount	**Defines the number of Cards in addition to the first Card Number in the sequence** - The maximum number of cards that can be activated at one time is 100 As such, the maximum number for this field is 99.	None	2	N
TxRefNum	**Gateway transaction reference number:** A unique value for each transaction, which is required to adjust any transaction in the gateway [such as Mark for Capture or void]. See rules for MFC's and Void for Gift Card transactions per host.	None	40 A	C

Field Name	Description	Default Value	Max/ Field Type	Req. [Y / N /C]
FlexEmployeeNumber	**Employee Number:** Employee number for person entering transaction. Tampa usage only.	None	10 A	N
PriorAuthID	**Prior Auth Code for Force Transactions** Complex Type	N/A	N/A	C
TraceNumber	**Retry Logic Number** This value is the trigger for implementing Retry Logic Submitting an invalid value will result in an XML Quick Response with a ProcStatus Code of 9714	None	16 N	N

11.12 Gift Card Response Elements

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
Request	**Required XML Parent Tag**	M	N/A	N/A
Gift CardResp	**XML tag that defines the transaction as a New Order request**	M	N/A	N/A
MerchantID	**Gateway merchant account number assigned by Chase Paymentech:** This account number will match that of your host platform BIN 000001 – 6 digit Salem Division Number BIN 000002 – 12 digit PNS Merchant ID	M	12	N
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech:** All Salem Terminal IDs at present must be '001'. PNS Terminal ID's can be from '001' – '999'. Most are '001'.	M	3	N
OrderID	**Merchant Defined Order Number:** Field defined and supplied by the auth originator, and echoed back in response.	M	22	A
AccountNum	**Card Number identifying the customer.** Should be null for electronic check processing	C	19	N
StartAccountNum	**Defines the first Card Number in a Block Activation Sequence**	C	19	N
BatchFailedAcctNum	**Card Number in a Block Activation Sequence that caused a Block Activation Failure** Conditionally returned on a Block Activation failure	C	19	N

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
FlexRequestedAmount	**Transaction amount submitted in the request**: Implied Decimal. Conditionally returned	C	12	N
FlexRedeemedAmt	**Actual amount Redeemed on a Redemption Completion where the flexPartialRedemptionInd = Yes**: Implied Decimal. Conditionally returned Regardless of whether the amount redeemed is equal to or less than the requested amount, it will be identified in this tag.	C	12	N
FlexHostTrace	**Gateway transaction reference number:** A unique value for each transaction, which is required to adjust any transaction in the gateway [such as Mark for Capture or void/Reversal].	C	40	N
FlexAction	**Returns the Transaction [or Action] Type performed from the request:** The FlexAction returned in the response is the same value submitted in the request.	M	30	A
FlexAcctBalance	**Current Balance of the Gift Card Card** The Balance after the result of the request transaction. This information will returned in all Gift Card response messages.	M	12	N
FlexAcctPriorBalance	**Prior Balance of the Gift Card Card** Balance prior to the result of the request transaction. This information will returned in all Gift Card response messages.	M	12	N
FlexAcctExpireDate	**The Gift Card Card Expiration Date** The Expiration Date of the Gift Card will be returned, if one exists. This information will returned in all response messages. Response Format = MMMMYY	M	6	N
CardBrand	**Mnemonic representing the of the request card type:** FC – Gift Card	M	2	A
TxRefNum	**Gateway transaction reference number:** A unique value for each transaction, which is required to adjust any transaction in the gateway.	M	40	A
TxRefIdx	**Gateway transaction index:** Used to identify the unique components of transactions adjusted more than one time. Required on for void transactions.	M	4	N

Field Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
ProcStatus	**Process Status:** The first data set that should be checked to determine the result of a request. It is the only element that is returned in all response scenarios. It identifies whether transactions have successfully passed all of the Gateway edit checks. 0 – Success All other values constitute an error condition and will be returned in a SOAPFault. See appendix for definition of those error values.	M	6	A
StatusMsg	**Text message associated with ProcStatus value.**	C	Var	A
ApprovalStatus	**Approval Status:** Conditional on: • Process Status returning a '0' or successful response. • Only Returned if performing a MFC on a Gift Card Card Type If so, approval status identifies the result of the authorization request to the host system. 0 – Decline 1 – Approved 2 – Message/System Error	M	1	N
AuthCode	**Issuer approval Code:** Unique transactional level code issuer uses to show each request was approved	M	6	A
RespCode	**Response Code:** Normalized authorization response code issued by the host system [Salem / PNS], which identifies an approval ('00') or the reason for a decline or error. Conditionally returned when procStatus = 0	M	2	A
CVV2RespCode	**Card verification value request response:** - See appendix for values - Conditional on card verification request being sent	M	1	A
RespTime	**Time the transaction was processed by gateway:** Format – 'hh24miss'	M	6	N

11.13 Profile Management Transaction – Request Elements

This transaction is used to send a profile management transaction (create, read, update, delete). To construct this XML transaction type in the Java SDK, pass the constant value "Profile" (statically defined in RequestIF.PROFILE_TRANSACTION constant) in the Request object's constructor.

```
RequestIF request = new Request (RequestIF. PROFILE_TRANSACTION);
```

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
OrbitalConnectionUsername	**Orbital Connection Username set up on Orbital Gateway** - Provide Username that is associated to this MID - Required if merchant is not set up for IP based authentication Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Not case sensitive	C	32	A
OrbitalConnectionPassword	**Orbital Connection Password used in conjunction with Orbital Username** - Provide Password associated with Connection Username Formats: - Between 8-32 characters (a-z, A-Z, 0-9 - Minimum 1 numeric digit - No leading, trailing, or embedded spaces - Password is case sensitive and must match with what is stored on Orbital Gateway exactly	C	32	A
CustomerBin	**Transaction Routing Definition:** Assigned by Chase Paymentech 000001 – Salem 000002 – PNS This value may not be changed through a Profile Update action. *This is the equivalent to the <BIN> element used on transactional requests*	M	6	N

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
CustomerMerchantID	**Gateway merchant account number assigned by Chase Paymentech:** This account number will match that of your host platform BIN 000001 – 6 digit Salem Division Number BIN 000002 – 12 digit PNS Merchant ID This value may not be changed through a Profile Update action. *This is the equivalent to the <MerchantID> element used on transactional requests*	M	15	N
CustomerName	**Customer Billing Name** Conditionally required for electronic check Profiles *This is the equivalent to the <AVSname> element used on transactional requests.*	C	30	A
CustomerRefNum	**Sets the Customer Reference Number that will be used to utilize a Customer Profile on all future Orders.** Mandatory if Customer Profile Action Type = Read, Update, or Delete Or Create and the Customer Profile Number generation option = S [Use CustomerRefNum Element] Keys: - If CustomerProfileFromOrderInd = A, the Customer Reference Number will be defined by the Gateway and any value passed in this element will be ignored. - Given that this value can be the same as the Order Number, the valid characters for this field follow the same convention as the Order ID element. Those valid characters are: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKL MNOPQRSTUVWXYZ0123456789-,$@& and a space character. However, the space character cannot be the leading character. This value may not be changed through a Profile Update action.	C	22	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
CustomerAddress1	**Cardholder Billing Address line 1** Optional if Customer Profile Action Type = Create or Update *This is the equivalent to the <AVSaddress1> element used on transactional requests.*	O	30	A
CustomerAddress2	**Cardholder Billing Address line 2** Optional if Customer Profile Action Type = Create or Update *This is the equivalent to the <AVSaddress2> element used on transactional requests.*	O	30	A
CustomerCity	**Cardholder Billing City** Optional if Customer Profile Action Type = Create or Update *This is the equivalent to the <AVScity> element used on transactional requests.*	O	20	A
CustomerState	**Cardholder Billing State** This is the equivalent to the <AVSstate> element used on transactional requests. Optional if Customer Profile Action Type = Create, or Update	O	2	A
CustomerZIP	**Cardholder Billing Address Zip Code:** - All AVS requests must minimally include the 5-digit Zip Code. - If sending Zip Code + 4, please separate with a '-' Conditionally required if Customer Profile Action Type = Create *This is the equivalent to the <AVSzip> element used on transactional requests.*	C	10	A
CustomerEmail	**Cardholder Email** Optional if Customer Profile Action Type = Create or Update *There is no equivalent to this element available on transactional requests.*	O	50	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
CustomerPhone	**Cardholder Telephone Number:** Format = AAAEEENNNNXXXX, where AAA – Area Code EEE – Exchange NNNN – Number XXXX – EXTENSION *Optional if Customer Profile Action Type = Create or Update* *This is the equivalent to the <AVSphoneNum> element used on transactional requests.*	O	14	A
CustomerCountryCode	**Cardholder Billing Address Country Code:** Required if processing a U.K. based Address. Valid values: US – United States CA – Canada GB – Great Britain UK – United Kingdom “ “ – Blank for all other countries *This element is only used for BML sale transactions.* *This is the equivalent to the <AVScountryCode> element used on transactional requests.*	C	2	A
CustomerProfileAction	**Defines the Customer Profile action desired:** C – Create a Customer Profile U – Update a Customer Profile R – Retrieve a Customer Profile’s Attributes D – Delete a Customer Profile *This element is only used for Customer Profile Management actions.*	M	6	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
CustomerProfileOrderOverrideInd	**Defines if any Order Data can be pre-populated from the Customer Reference Number [CustomerRefNum]:** Mandatory if Customer Profile Action Type = Create Optional if Customer Profile Action Type = Update **NO** No mapping to order data **OI** Use <*CustomerRefNum*> for <*OrderID*> and <*ECOrderNum*> or <*MailOrderNum*> **OD** Use <*CustomerReferNum*> for <*Comments*> **OA** Use <*CustomerRefNum*> for <*OrderID*> and <*ECOrderNum*> [or <*MailOrderNum*>] and <*Comments*>	C	2	A
CustomerProfileFromOrderInd	**Customer Profile Number generation Options:** A – Auto Generate the CustomerRefNum S – Use CustomerRefNum Element When performing an action other than a Customer Profile Action Type = Create, insert "empty" as it will be ignored. But this Attribute is mandatory and cannot be null filled.	M	5	A
OrderDefaultDescription	**Order Description** Optional if Customer Profile Action Type = Create or Update If the CustomerProfileOrderOverideInd = OA, do not set this value since the Customer Reference Number has been defaulted as the Order Description. *This is the equivalent to the <Comments> element used on transactional requests.*	O	64	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
OrderDefaultAmount	**Transaction Amount**: Optional if Customer Profile Action Type = Create or Update Keys: - Implied decimal including those currencies that are a zero exponent. For example, both $100.00 (an exp. of '2') and ¥100 (an exp. of '0') should be sent as <OrderDefaultAmount>10000</OrderDefault Amount> - Given that each Orbital Gateway Merchant ID is restricted to one currency, the Currency Code [and Exponent] is defaulted based on Merchant ID in which a transaction presented. *This is the equivalent to the <Amount> element used on transactional requests.*	O	12	N
CustomerAccountType	**Defines the Customers Payment Type:** CC – Credit Card SW – Switch / Solo EC – Electronic Check Mandatory if Customer Profile Action Type = Create Optional if Customer Profile Action Type = Update *This is the equivalent to the multiple XML elements used to define cardholder on transactional requests.*	C	1	A
Status	**Profile Status flag** This field is used to set the status of a Customer Profile. Valid values: A – Active I – Inactive MS – Manual Suspend	C	Var	A
CCAccountNum	**Customer Credit Card Number:** Mandatory if Customer Profile Action Type = Create and the Customer Payment Type = Credit card or Switch / Solo Optional if Customer Profile Action Type = Update and the Customer Payment Type = Credit card or Switch / Solo *This is the equivalent to the <AccountNum> element used on transactional requests.*	C	19	N

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
CCExpireDate	**Customer Credit Card Expiration Date:** Mandatory if Customer Profile Action Type = Create and the Customer Payment Type = Credit card or Switch / Solo Optional if Customer Profile Action Type = Update and the Customer Payment Type = Credit card or Switch / Solo - Formats: MMYY Salem {CustomerBIN 000001] allows a blank to be submitted when no known EXP date exists. Please discuss this feature with your certification analyst before implementing. There are three valid mechanisms for submitting a 'Blank' expiration date using Orbital to the Salem Host. They are: - Null fill this XML element - <Exp/> - Send four spaces - <Exp> </Exp> - Zero fill the XML Element - <Exp>0000</Exp> *This is the equivalent to the <Exp> element used on transactional requests.*	C	4	N
ECPAccountDDA	**ECP [DDA] Account Number:** Mandatory if Customer Profile Action Type = Create and the Customer Payment Type = ECP Optional if Customer Profile Action Type = Update and the Customer Payment Type = ECP *This is the equivalent to the <CheckDDA> element used on transactional requests.*	C	17	A
ECPAccountType	**Defines the ECP deposit account type:** Mandatory if Customer Profile Action Type = Create and the Customer Payment Type = ECP Optional if Customer Profile Action Type = Update and the Customer Payment Type = ECP C – Consumer Checking [US or Canadian] S – Consumer Savings [US Only] X – Commercial Checking [US Only] *This is the equivalent to the <BankAccountType> element used on transactional requests.*	C	1	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
ECPAccountRT	**ECP Bank routing and transit number for the customer.** Mandatory if Customer Profile Action Type = Create and the Customer Payment Type = ECP Optional if Customer Profile Action Type = Update and the Customer Payment Type = ECP NOTE: - All US Bank Routing Numbers are 9 digits - All Canadian Bank Routing Numbers are 8 Digits This is the equivalent to the <BCRtNum> element used on transactional requests.	C	9	N
ECPBankPmtDlv	**Defines the ECP payment delivery method:** Mandatory if Customer Profile Action Type = Create and the Customer Payment Type = ECP Optional if Customer Profile Action Type = Update and the Customer Payment Type = ECP This field indicates the preferred manner to deposit the transaction. B – Best Possible Method [US Only] Chase Paymentech utilizes the method that best fits the situation. If the RDFI is not an ACH participant, a facsimile draft will be created. This should be the default value for this field. A – ACH [US or Canadian] Deposit the transaction by ACH only. If the RDFI is not an ACH participant, the transaction is rejected. *This is the equivalent to the <BankPmtDelv> element used on transactional requests.*	C	1	A
SwitchSoloStartDate	**Customer Switch / Solo cards activation date:** Mandatory if Customer Profile Action Type = Create and the Customer Payment Type = Switch / Solo Optional if Customer Profile Action Type = Update and the Customer Payment Type = Switch / Solo - Format: MMYY *This is the equivalent to the <DebitCardStartDate> element used on transactional requests.*	C	4	N

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
SwitchSoloIssueNum	**Customer Switch / Solo cards Issue Number:** Mandatory if Customer Profile Action Type = Create and the Customer Payment Type = Switch / Solo Optional if Customer Profile Action Type = Update and the Customer Payment Type = Switch / Solo Switch / Solo incremental counter for lost or replacement cards. *This is the equivalent to the <DebitCardIssueNum> element used on transactional requests.*	C	2	N
MBType	**Managed Billing Type** Indicates the type of Managed Billing the merchant is participating in. The value submitted must be in agreement with the type of Managed Billing the merchant is configured for at Chase Paymentech. Valid Values: R – Recurring D – Deferred This field serves to notify the Orbital system that the transaction is a Managed Billing transaction. If this field is not sent in conjunction with a Managed Billing transaction, any other Managed Billing fields will be ignored.	C	1	A
MBOrderIdGenerationMethod	**Managed Billing Order ID generation method** This value is used to set the method that Orbital will use to generate the Order ID for any Managed Billing transactions. This field does NOT influence the Order ID for stand-alone transactions initiated by the merchant, VT transactions, etc. Valid values: IO – Use Customer Reference Number (Profile ID); this value is made up of the capital letters 'I' and 'O', no numerals. DI – Dynamically generate; this value is made up of the capital letters 'D' and 'I', no numerals.	C	2	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
MBRecurringStartDate	**Managed Billing Recurring Start Date** Defines the future date that Orbital will begin a recurring billing cycle to the associated Profile. To allow the Managed Billing engine to properly calculate and schedule all billings, this date must be at least one day after the request date (a recurring billing cycle can never begin on the date that the request message is sent to the Orbital system). Format is MMDDYYYY	C	8	N
MBRecurringEndDate	**Managed Billing Recurring End Date** Defines the future date that Orbital will end a recurring billing cycle to the associated Profile. Format is MMDDYYYY This is the first of three possible recurring end triggers. Only one end trigger can be submitted per request message.	C	8	N
MBRecurringNoEndDateFlag	**Managed Billing 'No End Date' Indicator** Valid values: Y – this value will schedule recurring transactions for an infinite amount of time. If "Y", this value will override MBRecurringEndDate (above), even if it is populated with an end date. N (or blank) – Orbital will use MBRecurringEndDate(above) to define the recurring end date This is the second of three possible recurring end triggers. Only one end trigger can be submitted per request message.	C	1	A
MBRecurringMaxBillings	**Managed Billing Max Number of Billings** This value will define the maximum number of billings that will be allowed for a recurring billing cycle. Valid values: 1 – 999999 This is the third of three possible recurring end triggers. Only one end trigger can be submitted per request message.	C	6	N

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
MBRecurringFrequency	**Managed Billing Recurring Frequency Pattern** This pattern is a subset of a standard CRON expression, comprised of 3 fields separated by white space. <u>Field</u> <u>Allowed Values</u> <u>Allowed Special Chars</u> 1)Day-of-month 1-31 , - * ? / L W 2)Month 1-12 or JAN-DEC , - * / 3)Day-of-week 1-7 or SUN-SAT , - * ? / L # For a full discussion of these three fields, the usage of the special characters, and multiple example values, see the Profiles and Managed Billing section above (prior to the message layouts).	C	Var	A
MBDeferredBillDate	**Managed Billing Deferred billing date** Defines the future date that Orbital will trigger a one-time billing to the associated Profile. This date must be at least one day after the request date (a deferred billing can never take place on the date that the request message is sent to the Orbital system). Format is MMDDYYYY	C	8	N
MBCancelDate	**Managed Billing Cancel Date** This field is used to cancel a single future billing that is already scheduled. The exact date of the scheduled billing must be submitted. Format is MMDDYYYY			
MBRestoreBillingDate	**Managed Billing Restore Billing Date** This field is used to reinstate a cancelled billing. The exact date of the previously scheduled billing must be submitted in order for this action to work. Format is MMDDYYYY			
MBRemoveFlag	**Managed Billing Remove Flag** Valid values: Y – this value is used to remove all Managed Billing settings from the associated Profile. The Profile will become a 'Standard" Profile and any scheduled future billings are removed from the Orbital system and will not occur. N (or blank) – this value has no affect on the Profile.	C	1	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
EUDDCountryCode	**European Direct Debit Country Code** Customer's Country Code. The following is the list of valid country codes. AT – Austria BE – Belgium DE – Germany FR – France GB – United Kingdom NL – Netherlands	C	2	A
EUDDBankSortCode	**European Direct Debit Bank Sort Code** Customer's Bank Sort code. Mandatory for the following Country Codes: AT – Austria DE – Germany FR – France GB – United Kingdom	C	10	A
EUDDRibCode	**European Direct Debit RIB** Bank Account checksum – used in France only	C	2	A
SDMerchantName	**Soft Descriptor Merchant Name** Conditionally required for Soft Descriptors The Merchant Name field should be what is most recognizable to the cardholder [Company name or trade name]. The actual length of this field is conditionally tied to Host and the Size of the <SDProductDescription> field used. Salem: - CREDIT – Three options, which conditionally affects the SDProductDescription [see below]: o Max 3 bytes o Max 7 bytes o Max 12 bytes - ECP: o Max 15 bytes PNS: - Max 25 bytes.	C	25	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
SDProductDescription	**Soft Descriptor Product Description** Conditionally required for Soft Descriptors. Provides an accurate product description Salem: - CREDIT: o If SDMerchantName = 3 bytes – then Max = 18 bytes o If SDMerchantName = 7 bytes – then Max = 14 bytes o If SDMerchantName = 12 bytes – then Max = 9 bytes - ECP: o 10 bytes Max PNS: - This field will **not** show on Cardholder statements for PNS Merchants.	C	18	A
SDMerchantCity	**Soft Descriptor Merchant City** Tag conditionally required for Soft Descriptors. Merchant City for Retail. Field required but should be null filled if any Soft Descriptor data is submitted.	C	13	A
SDMerchantPhone	**Soft Descriptor Merchant Phone** Tag conditionally required for Soft Descriptors Only one of the location Soft Descriptor records should be sent [meaning Phone, URL, or Email]. This field will **not** show on Cardholder statements for PNS Merchants. Valid Formats: - NNN-NNN-NNNN - NNN-AAAAAAA NOTE: For MasterCard MOTO (Transaction Type 1) and Recurring (Transaction Type 2), if the City/Phone field at the division level is not a Customer Service Phone Number, then a Customer Service Phone Number must be populated in the Merchant city/Customer Phone Number field or the transaction will reject with Response Reason Code BP (Missing Customer Service Phone).	C	12	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
SDMerchantURL	**Soft Descriptor Merchant URL** Tag conditionally required for Soft Descriptors [can be null filled] Only one of the location Soft Descriptor records should include data [meaning Phone, URL, or Email]. This field will **not** show on Cardholder statements for PNS Merchants.	C	13	A
SDMerchantEmail	**Soft Descriptor Merchant Email** Tag conditionally required for Soft Descriptors [can be null filled] Only one of the location Soft Descriptor records should include data [meaning Phone, URL, or Email]. This field will **not** show on Cardholder statements for PNS Merchants.	C	13	A

11.14 Profile Response Elements

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
CustomerBin	**Transaction Routing Definition:** Value from the Request Returned	M	6	N
CustomerMerchantID	**Gateway merchant account number from Request**	M	15	N
CustomerName	**Customer Billing Name**	M	30	A
CustomerRefNum	**Customer Reference Number**	M	22	A
CustomerProfileAction	**Defines the Customer Profile action requested**	M	6	A
CustomerProfileMessage	**Verbose Text Description associated with ProfileProcStatus**	M	Var	A
CustomerAddress1	**Cardholder Billing Address line 1**	M	30	A
CustomerAddress2	**Cardholder Billing Address line 2**	M	30	A
CustomerCity	**Cardholder Billing City**	M	20	A
CustomerState	**Cardholder Billing State**	M	2	A
CustomerZIP	**Cardholder Billing Address Zip Code:**	M	10	A
CustomerEmail	**Cardholder Email**	M	50	A
CustomerPhone	**Cardholder Telephone Number:** Format = AAAEEENNNNXXXX, where AAA – Area Code EEE – Exchange NNNN – Number XXXX – EXTENSION	M	14	A
CustomerProfileOrderOverrideInd	**Defines if any Order Data can be pre-populated from the Customer Reference Number [CustomerRefNum]:** **NO** No mapping to order data **OI** Use <*CustomerRefNum*> for <*OrderID*> and <*ECOrderNum*> or <*MailOrderNum*> **OD** Use <*CustomerReferNum*> for <*Comments*> **OA** Use <*CustomerRefNum*> for <*OrderID*> and <*ECOrderNum*> [or <*MailOrderNum*>] and <*Comments*>	M	2	A
OrderDefaultDescription	**Order Description**	M	64	A
OrderDefaultAmount	**Defaulted Transaction Amount** Implied decimal	M	12	N

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
CustomerAccountType	**Defines the Customers Payment Type** C – Credit Card S – Switch / Solo E – Electronic Check	M	1	A
Status	**Defines the current status of a Profile** A – Active I – Inactive MS – Manual Suspend	C	Var	A
CCAccountNum	**Customer Credit Card Number**	M	19	N
CCExpireDate	**Customer Credit Card Expiration Date**	M	4	N
ECPAccountDDA	**ECP [DDA] Account Number**	M	17	A
ECPAccountType	**Defines the ECP deposit account type** C – Consumer Checking [US or Canadian] S – Consumer Savings [US Only] X – Commercial Checking [US Only]	M	1	A
ECPAccountRT	**ECP Bank routing and transit number for the customer**	M	9	N
ECPBankPmtDlv	**Defines the ECP payment delivery method** B – Best Possible Method [US Only] Chase Paymentech utilizes the method that best fits the situation. If the RDFI is not an ACH participant, a facsimile draft will be created. This should be the default value for this field. A – ACH [US or Canadian] **Deposit the transaction by ACH only. If the RDFI is not an ACH participant, the transaction is rejected.**	M	1	A
SwitchSoloStartDate	**Customer Switch / Solo cards activation date** Format: MMYY	M	4	N
SwitchSoloIssueNum	**Customer Switch / Solo cards Issue Number**	M	2	N
MBType	**Managed Billing Type** R – Recurring D – Deferred	C	1	A
MBOrderIdGenerationMethod	**Managed Billing Order ID generation method** IO – Use Customer Reference Number (Profile ID); this value is made up of the capital letters 'I' and 'O', no numerals. DI – Dynamically generate; this value is made up of the capital letters 'D' and 'I', no numerals.	C	2	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
MBRecurringStartDate	**Managed Billing Recurring Start Date** Defines the future date that Orbital will begin a recurring billing cycle to the associated Profile. Format is MMDDYYYY	C	8	N
MBRecurringEndDate	**Managed Billing Recurring End Date** Defines the future date that Orbital will end a recurring billing cycle to the associated Profile. Format is MMDDYYYY	C	8	N
MBRecurringNoEndDateFlag	**Managed Billing 'No End Date' Indicator** Y – This value will schedule recurring transactions for an infinite amount of time. If "Y", this value will override element 65 (above), even if populated with an end date. N (or blank) – Orbital will use element 65 (above) to define the recurring end date	C	1	A
MBRecurringMaxBillings	**Managed Billing Max Number of Billings** This value will define the maximum number of billings that will be allowed for a recurring billing cycle. Valid values: 1 - 999999	C	6	N
MBRecurringFrequency	**Managed Billing Recurring Frequency Pattern** This pattern is a subset of a standard CRON expression, comprised of 3 fields separated by white space. For a full discussion of these three fields, the usage of the special characters, and multiple example values, see Appendix B.	C	Var	A
MBDeferredBillDate	**Managed Billing Deferred billing date** Format is MMDDYYYY	C	8	N
MBCustomerStatus	**Managed Billing Customer Status** Text message indicating the status of a Managed Billing request	C	Var	A
EUDDCountryCode	**European Direct Debit Country Code** AT – Austria BE – Belgium DE – Germany FR – France GB – United Kingdom NL – Netherlands	C	2	A

Field Name	Description	Req. M/C/O	Max. Char	Field Type A/N
EUDDBankSortCode	**European Direct Debit Bank Sort Code** AT – Austria DE – Germany FR – France GB – United Kingdom	C	10	A
EUDDRibCode	**European Direct Debit RIB** Bank Account checksum – used in France only	C	2	A
SDMerchantName	**Soft Descriptor Merchant Name**	C	25	A
SDProductDescription	**Soft Descriptor Product Description**	C	18	A
SDMerchantCity	**Soft Descriptor Merchant City**	C	13	A
RespTime	**Response Time**	C	Var	N
SDMerchantPhone	**Soft Descriptor Merchant Phone**	C	12	A
SDMerchantURL	**Soft Descriptor Merchant URL**	C	13	A
SDMerchantEmail	**Soft Descriptor Merchant Email**	C	13	A
RespTime	**Time the transaction was processed by the Gateway:** Format – 'hh24miss'	M	6	N

11.15 Quick Response Elements

When a transaction has an error condition, such as a time out condition or a poorly formed message request, the gateway will generate a quick error message back to the requestor. This error response takes the form of a "Quick Response".

Complex Type Name:
Quick Response = QuickResp

XML Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
MerchantID	**Gateway merchant account number assigned by Chase Paymentech Solutions:** Echoes the account number passed in the request	M	12	N
TerminalID	**Merchant Terminal ID assigned by Chase Paymentech Solutions:** Echoes the Terminal ID passed in the request	M	3	N
OrderID	**Merchant Defined Order Number:** Echoes the Order Number passed in the request	C	22	A
AccountNum	**Card Number identifying the customer.** Echoes the Account Number passed in the request	C	19	N
StartAccountNum	**Defines the first Card Number in a Block Activation Sequence**	C	19	N
TxRefNum	**Gateway transaction reference number:** Conditionally returned dependant on the error	C	40	A
TxRefIdx	**Gateway transaction index:**	C	4	A
ProcStatus	**Process Status:** The first element that should be checked to determine the result of a request. It is the only element that is returned in all response scenarios. It identifies whether transactions have successfully passed all of the Gateway edit checks. See appendix for definition of those error values.	M	6	A
StatusMsg	**Text message associated with ProcStatus value.**	C	Var	A
ApprovalStatus	**Approval Status** Conditional on Process Status returning a '0' or successful response. If so, approval status identifies the result of the authorization request to the host system. 0 - Decline 1 - Approved 2 - Message/System Error	C	1	N
CustomerBin	**Transaction Routing Definition** Value from the Request Returned	C	6	N
CustomerMerchantID	**Gateway merchant account number from Request**	C	15	N

XML Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
CustomerName	**Customer Billing Name** Value from the Request Returned	C	30	A
CustomerRefNum	**The Customer Reference Number**	C	22	A
CustomerProfileAction	**Defines the Customer Profile action requested**	C	6	A
ProfileProcStatus	**Result Status of Profile Management** Communicates the success or failure of a Profile Management request 0 - Success Greater than 0 - An error condition, see Appendix A for definition of the specific Profile Management error values	C	6	A
CustomerProfileMessage	**Verbose Text Description associated with ProfileProcStatus**	C	Var	A
CustomerAddress1	**Cardholder Billing Address line 1**	C	30	A
CustomerAddress2	**Cardholder Billing Address line 2**	C	30	A
CustomerCity	**Cardholder Billing City**	C	20	A
CustomerState	**Cardholder Billing State**	C	2	A
CustomerZIP	**Cardholder Billing Address ZIP Code**	C	10	A
CustomerEmail	**Cardholder Email**	C	50	A

CustomerPhone	**Cardholder Telephone Number**
	
	Format: AAAEEENNNNXXXX, where:
	AAA = Area Code
	EEE = Exchange
	NNNN = Number
	XXXX = Extension
		C	14	A

CustomerProfileOrderOverrideInd	**Defines if any Order Data can be pre-populated from the Customer Reference Number [CustomerRefNum]** NO - No mapping to order data OI - Use <CustomerRefNum> for <OrderID> and <ECOrderNum> or <MailOrderNum> OD - Use <CustomerReferNum> for <Comments> OA - Use <CustomerRefNum> for <OrderID> and <ECOrderNum> [or <MailOrderNum>] and <Comments>	C	2	A
OrderDefaultDescription	**Order Description**	C	64	A
OrderDefaultAmount	**Defaulted Transaction Amount** Implied decimal	C	12	N

XML Name	Description	Req. M/C/O	Max. Char.	Field Type A/N
CustomerAccountType	**Defines the Card Type/Brand for the Transaction** Echoes the Card Type/Brand passed in the request, except: - If no CardBrand was sent in the request [when optional], such as Visa / MasterCard, the specific Card Brand mnemonic is returned. - For PINLess Debit transactions, the request Card Brand is DP [which is a generic PINLess mnemonic. However the response Card Brand will be one of the three supported PINLess Debit Card Brands: NP - NYCE PINless Debit PP - Pulse PINless Debit SP - Star PINless Debit	C	2	A
CCAccountNum	**Customer Credit Card Number**	C	19	N
CCExpireDate	**Customer Credit Card Expiration Date**	C	4	N
ECPAccountDDA	**ECP [DDA] Account Number**	C	17	A
ECPAccountType	**Defines the ECP deposit account type** C - Consumer Checking [US or Canadian] S - Consumer Savings [US Only] X - Commercial Checking [US Only]	C	1	A
ECPAccountRT	**ECP Bank routing and transit number for the customer**	C	9	N
ECPBankPmtDlv	**Defines the ECP payment delivery method** B – Best Possible Method [US Only] Chase Paymentech utilizes the method that best fits the situation. If the RDFI is not an ACH participant, a facsimile draft will be created. This should be the default value for this field. A – ACH [US or Canadian] Deposit the transaction by ACH only. If the RDFI is not an ACH participant, the transaction is rejected.	C	1	A
SwitchSoloStartDate	**Customer Switch/Solo card activation date** Format: MMYY	C	4	N
SwitchSoloIssueNum	**Customer Switch/Solo card issue number**	C	2	N
RespTime	**Time the transaction was processed by gateway:** Format – 'hh24miss'	M	6	N

12 Appendix Values

12.1 CurrencyCode and CurrencyExponent Tag Values

Presentment Currencies by Country	ISO Currency Codes	Currency Decimals
Algerian Dinar	012	2
Argentine Peso	032	2
Armenian Dram	051	2
Aruban Guilder	533	2
Australian Dollar	036	2
Azerbaijanian Manat	944	2
Bahamian Dollar	044	2
Bangladeshi Taka	050	2
Barbados Dollar	052	2
Belarussian Ruble	974	0
Belize Dollar	084	2
Bermudian Dollar	060	2
Bolivian Boliviano	068	2
Botswana Pula	072	2
Brazilian Real	986	2
British Pound	826	2
Brunei Dollar	096	2
Bulgarian Lev	975	2
Burundi Franc	108	0
CFA Franc BCEAO	952	0
CFA Franc BEAC	950	0
CFP Franc	953	0
Cambodian Riel	116	2
Canadian Dollar	124	2
Cape Verdi Escudo	132	2
Cayman Islands Dollar	136	2
Chilean Peso	152	2
Chinese Yuan Renminbi	156	2
Colombian Peso	170	2
Comoro Franc	174	0
Costa Rican Colon	188	2
Cyprus Pound	196	2
Czech Koruna	203	2
Danish Krone	208	2
Djibouti Franc	262	0
Dominican Peso	214	2
East Caribbean Dollar	951	2
Egyptian Pound	818	2
El Salvador Colon	222	2
Estonian Kroon	233	2

Ethiopian Birr	230	2
Euro	978	2
Falkland Islands Pound	238	2
Fiji Dollar	242	2
Gambian Dalasi	270	2
Georgian Lari	981	2
Ghanaian Cedi	288	2
Gibraltar Pound	292	2
Guatemala Quetzal	320	2
Guinea Franc	324	2
Guinea-Bissau Peso	624	2
Guyanese Dollar	328	2
Haitian Gourde	332	2
Honduras Limpera	340	2
Hong Kong Dollar	344	2
Hungarian Forint	348	2
Iceland Krona	352	2
Indian Rupee	356	2
Indonesian Rupiah	360	2
Israeli New Shekel	376	2
Jamaican Dollar	388	2
Japanese Yen	392	0
Kazakhstan Tenge	398	2
Kenyan Shilling	404	2
Kyrgyzstan Som	417	2
Laos Kip	418	0
Latvian Lats	428	2
Lebanese Pound	422	2
Lithuanian Litas	440	2
Macau Pataca	446	2
Malagasy Franc	450	0
Malawi Kwacha	454	2
Malaysian Ringgit	458	2
Maldive Rufiyaa	462	2
Maltese Lira	470	2
Mauritania Ouguiya	478	2
Mauritius Rupee	480	2
Mexican Peso	484	2
Moldovan Leu	498	2
Mongolia Tugrik	496	2
Moroccan Dirham	504	2
Mozambique Metical	943	2
Namibia Dollar	516	2
Nepalese Rupee	524	2
Netherlands Antillean Guilder	532	2
New Guinea Kina	598	2
New Zealand Dollar	554	2

Nicaraguan Cordoba Oro	558	2
Nigerian Naira	566	2
Norwegian Krone	578	2
Pakistan Rupee	586	2
Panamanian Balboa	590	2
Paraguay Guarani	600	0
Peruvian Nuevo Sol	604	2
Philippines Peso	608	2
Polish Zloty	985	2
Qatari Rial	634	2
Romania Leu	946	2
Russian Ruble	643	2
Rwanda Franc	646	0
St. Helena Pound	654	2
Samoan Tala	882	2
Sao Tome & Principe Dobra	678	2
Saudi Riyal	682	2
Seychelles Rupee	690	2
Sierra Leonean Leone	694	2
Singapore Dollar	702	2
Slovak Koruna	703	2
Solomon Islands Dollar	090	2
Somali Shilling	706	2
South African Rand	710	2
South Korean Won	410	0
Sri Lanka Rupee	144	2
Swaziland Lilangeni	748	2
Swedish Krona	752	2
Swiss Franc	756	2
Taiwan Dollar (New)	901	2
Tanzanian Shilling	834	2
Thai Baht	764	2
Tonga Pa’anga	776	2
Trinidad & Tobago Dollar	780	2
Turkish Lira (New)	792	0
Uganda Shilling	800	2
Ukrainian Hryvnia	980	2
United Arab Emirates Dirham	784	2
Uruguayan Peso	858	2
US Dollar	840	2
Uzbekistan Sum	860	2
Vanuatu Vatu	548	0
Venezuelan Bolivar	862	2
Vietnamese Dong	704	2
Yemeni Rial	886	2
Zambia Kwacha	894	2
Zimbabwe Dollar	716	2

12.2 Purchasing Card 3 Tables

12.2.1 ISO Country Codes

ISO Code	Country
GBR	UNITED KINGDOM
DEU	GERMANY
FRA	FRANCE
AUT	AUSTRIA
BEL	BELGIUM
CAN	CANADA
CZE	CZECH REPUBLIC
DNK	DENMARK
FIN	FINLAND
GRC	GREECE
HUN	HUNGARY
ISL	ICELAND
IRL	IRELAND
ISR	ISRAEL
ITA	ITALY
LIE	LIECHTENSTEIN
LUX	LUXEMBOURG
MEX	MEXICO
NLD	NETHERLANDS
NOR	NORWAY
POL	POLAND
SVK	SLOVAKIA
ESP	SPAIN
SWE	SWEDEN
CHE	SWITZERLAND
USA	UNITED STATES
ARE	UNITED ARAB EMIRATES
AUD	AUSTRALIA
HKD	HONGKONG
JPY	JAPAN
MYR	MALAYSIA
NZD	NEW ZEALAND

ISO Code	Country
SGD	SINGAPORE
ZAD	SOUTH AFRICA

12.2.2 Unit of Measure

UOM Code	Unit Name
ACR	Acre
ASM	Alcoholic strength by mass
ASV	Alcoholic strength by volume
AMP	Ampere
AMH	Ampere-hour (3,6 kC)
ARE	Are (100 m2)
BAR	Bar
BLL	Barrel (petroleum) (158,987 dm3)
BFT	Board foot
BQL	Becquerel
BIL	Billion EUR
MLD	Billion US
BHP	Brake horse power (245,7 watts)
BTU	British thermal unit (1,055 kilojoules)
BUA	Bushel (35,2391 dm3)
BUI	Bushel (36,36874 dm3)
CDL	Candela
CCT	Carrying capacity in metric tonnes
CNT	Cental GB (45,359237 kg)
CGM	Centigram
CLT	Centilitre
CMT	Centimetre
DTN	Centner, metric (100 kg)
WCD	Cord (3,63 m3)
COU	Coulomb
CKG	Coulomb per kilogram
CMQ	Cubic centimeter
DMQ	Cubic decimeter
FTQ	Cubit foot
INQ	Cubic inch
MTQ	Cubic metre
MQH	Cubic metre per hour
MQS	Cubic metre per second
MMQ	Cubic millimetre
YDQ	Cubic yard
CUR	Curie
DAY	Day
DAA	Decare
DLT	Decilitre
DMT	Decimetre
DTN	Decitonne

UOM Code	Unit Name
CEL	Degree Celsius
FAH	Degree Fahrenheit
	Degree Kelvin: see Kelvin
DPT	Displacement tonnage
DZN	Dozen
DZP	Dozen packs
DZR	Dozen pairs
DCP	Dozen pieces
DRL	Dozen rolls
DRM	Drachm GB (3,887935 g)
DRI	Dram GB (1,771745 g)
DRA	Dram US (3,887935 g)
BLD	Dry barrel (115,627 dm3)
GLD	Dry gallon (4,404884 dm3)
PTD	Dry pint (0,55061 dm3)
QTD	Dry quart (1,101221 dm3)
FAR	Farad
OZI	Fluid ounce (28,413 cm3)
OZA	Fluid ounce (29,5735 cm3)
FOT	Foot (0,3048 m)
GLI	Gallon (4,546092 dm3)
GBQ	Gigabecquerel
GWH	Gigawatt-hour (1 million kW/h)
GII	Gill (0,142065 dm3)
GIA	Gill (11,8294 cm3)
GRN	Grain GB, US (64,798910 mg)
GRM	Gram
GFI	Gram of fissile isotopes
GGR	Great gross (12 gross)
GRO	Gross
GRT	Gross (register) ton
SAN	Half year (six months)
HAR	Hectare
HBA	Hectobar
HGM	Hectogram
DTH	Hectokilogram
HLT	Hectolitre
HPA	Hectolitre of pure alcohol
HMT	Hectometre
HTZ	Hertz
HUR	Hour
CEN	Hundred
BHX	Hundred boxes
HIU	Hundred international units
CLF	Hundred leaves
CNP	Hundred packs
CWA	Hundredweight US (45,3592 kg)
INH	Inch (25,4 mm)
JOU	Joule

UOM Code	Unit Name
KEL	Kelvin
KBA	Kilobar
KGM	Kilogram
KPH	Kilogram of caustic potash
KSH	Kilogram of caustic soda
KNS	Kilogram of named substance
KNI	Kilogram of nitrogen
KPP	Kilogram of phosphonic anhydride
KPP	Kilogram of phosphorus pentoxide
KPH	Kilogram of potassium hydroxide
KPO	Kilogram of potassium oxide
KSH	Kilogram of sodium hydroxide
KSD	Kilogram of substance 90% dry
KUR	Kilogram of uranium
KMQ	Kilogram per cubic meter
KGS	Kilogram per second
KHZ	Kilohertz
KJO	Kilojoule
KMT	Kilometre
KMH	Kilometre per hour
KPA	Kilopascal
KTN	Kilotonne
KVR	Kilovar
KVT	Kilovolt
KVA	Kilovolt-ampere
KWT	Kilowatt
KWH	Kilowatt-hour
KNT	Knot (1 nautical mile per hour)
LEF	Leaf
GLL	Liquid gallon (3,78541 dm3)
PTL	Liquid pint (0,473176 dm3)
QTL	Liquid quart (0,946353 dm3)
LTR	Litre (1dm3)
LPA	Litre of pure alcohol
CWI	(Long) hundredweight GB (50,802345 kg)
LTN	Long ton GB, US (1,0160469 t)
LUM	Lumen
LUX	Lux
MHZ	Megahertz
MAL	Megalitre
MAM	Megametre
MPA	Megapascal
MVA	Megavolt-ampere (1000 KVA)
MAW	Megawatt
MWH	Megawatt-hour (100 kW/h)
MTR	Metre
MTS	Metre per second
MSK	Metre per second squared
CTM	Metric carat (200 mg = 2.10-4 kg)

UOM Code	Unit Name
TNE	Metric ton (1000 kg)
MLD	Milliard
MBR	Millibar
MCU	Millicurie
MGM	Milligram
MLT	Millilitre
MMT	Millimetre
MIO	Million
HMQ	Million cubic metres
MIU	Million international units
MIN	Minute
MON	Month
NMI	Nautical mile (1852 m)
NTT	Net (register) ton
NEW	Newton
NMB	Number
NAR	Number of articles
NBB	Number of bobbins
NCL	Number of cells
NIU	Number of international units
NMP	Number of packs
NMR	Number of pairs
NPL	Number of parcels
NPT	Number of parts
NRL	Number of rolls
OHM	Ohm
ONZ	Ounce GB, US (28,349523 g)
APZ	Ounce GB, US (31,10348 g)
PAL	Pascal
DWT	Pennyweight GB, US (1,555174 g)
PCE	Piece
PTI	Pint (0,568262 dm3)
LBR	Pound GB, US (0,45359237 kg)
PGL	Proof gallon
QTI	Quart
QAN	Quarter (of a year)
QTR	Quarter, GB (12,700586 kg)
DTN	Quintal, metric (100 kg)
RPM	Revolution per minute
RPS	Revolution per second
SCO	Score
SCR	Scruple GB, US (1,295982 g)
SEC	Second
SET	Set
SHT	Shipping ton
SST	Short standard
STN	Short ton GB, US (0,90718474 t)
SIE	Siemans
CMK	Square centimeter

UOM Code	Unit Name
DMK	Square decimeter
FTK	Square foot
INK	Square inch
KMK	Square kilometer
MTK	Square metre
MIK	Square mile
MMK	Square millimeter
TDK	Square yard
WSD	Standard
ATM	Standard atmosphere (101325 Pa)
SMI	(Statute) mile (1609,344 m)
STI	Stone GB (6,350293 kg)
ATT	Technical atmosphere (98066,5 Pa)
DAD	Ten days
TPR	Ten pairs
MIL	Thousand
TAH	Thousand ampere-hour
MBF	Thousand board feet (2,36 m3)
TQD	Thousand cubic metres per day
MBE	Thousand standard brick equivalent
TSH	Ton of steam per hour
TNE	Tonne (1000 kg)
TSD	Tonne of substance 90% dry
TRL	Trillion EUR
BIL	Trillion US
APZ	Troy Ounce
LBT	Troy pound, US (373,242 g)
VLT	Volt
WTT	Watt
WHR	Watt-hour
WEB	Weber
WEE	Week
YRD	Yard
ANN	Year

12.3 General Card Validation

There are three common edits that catch the greatest majority of bad card numbers:

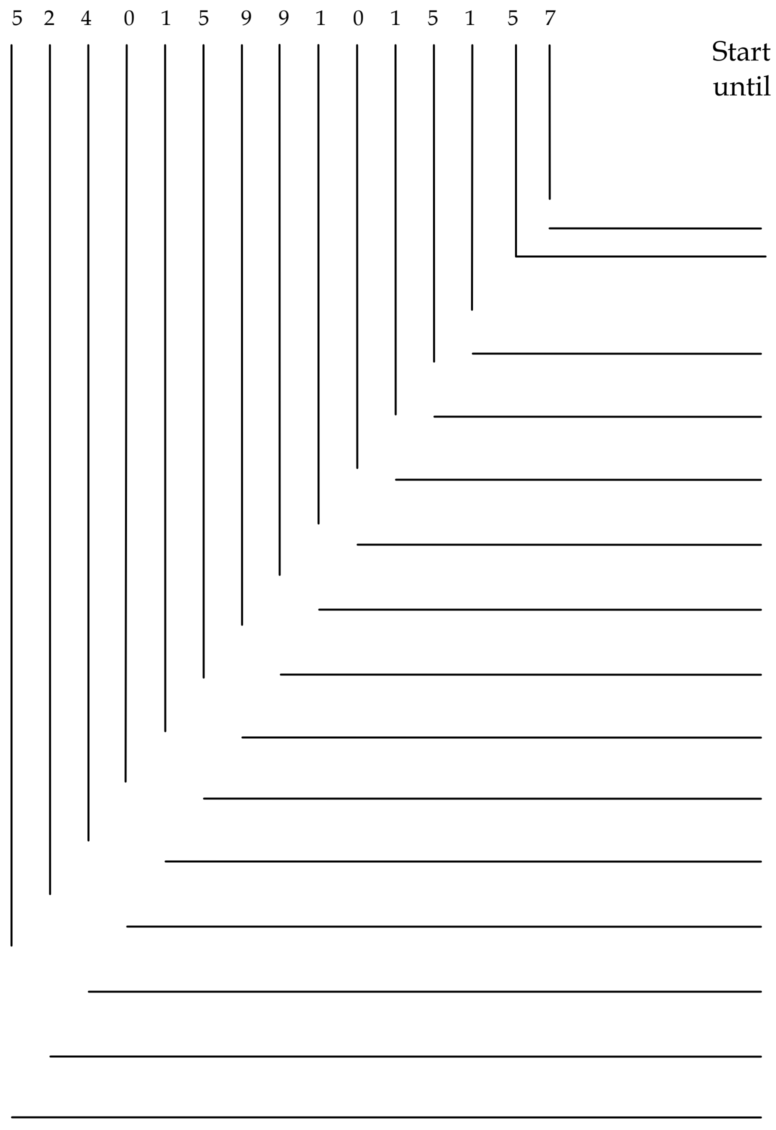
- MOD 10 check digit
- Credit card prefix check
- Credit card length validation

12.3.1 MOD 10 Check Digit

The MOD 10 check digit calculation validates the credit card by calculating the last digit of the card number from all the other numbers in the card.

The last digit of a credit card can be calculated based on a calculation performed upon all the digits preceding it. This operation is called a **MOD 10 check-digit routine.**

Use the following card for Example: 524015991015157**3**



Start from the right and proceed to the left until all digits are multiplied by weight

7 * 2 = 14	sum = 1 + 4		= 5
5 * 1 = 5	sum = sum (5)	+5	=10
1 * 2 = 2	sum = sum(10)	+2	=12
5 * 1 = 5	sum = sum(12)	+5	=17
1 * 2 = 2	sum = sum(17)	+2	=19
0 * 1 = 0	sum = sum(19)	+0	=19
1 * 2 = 2	sum = sum(19)	+2	=21
9 * 1 = 9	sum = sum(21)	+9	=30
9 * 2 = 18	sum = sum(30)	+1 + 8	=39
5 * 1 = 5	sum = sum(39)	+5	=44
1 * 2 = 2	sum = sum(44)	+2	=46
0 * 1 = 0	sum = sum(46)	+0	=46
4 * 2 = 8	sum = sum(46)	+8	=54
2 * 1 =2	sum = sum(54)	+2	=56
5 * 2 =10	sum = sum(56)	+1+0	=57

Remove the **check digit**, 3, which is present in this example card.
sum = 57
sum MOD 10 — — —> 57 MOD 10 = 7
10-7 = **3**
check digit of 524015991015157**3** is **3**.

12.3.2 Card Prefix Check

The prefix check is the comparison of the first few digits of each card number to a list of known prefixes.

Card Type	Prefix
American Express / Optima	37, 34
Carte Blanche	389
Diners Club	30, 36, 381 - 388
Discover (Novus)	60110, 60112, 60113, 60114, 60119, 601174, 601177, 601178, 601179, 650
JCB	3528 – 3589
MasterCard	51 – 55, 36
Switch/Solo [BIN 000001 ONLY]	49, 56, 6* where * is any single digit
Visa/Delta	4
Bill Me Later	504990, 621993
Flex Cache	603571

12.3.3 Card Length Check

The number of digits for each card is constant, allowing a validation to be performed by verifying the number of digits for each card number.

Card Type	Length
American Express / Optima	15
Carte Blanche	14
Diners Club	14
Discover (Novus)	16
JCB	16
MasterCard	16
Switch/Solo [BIN 000001 ONLY]	16, 18, or 19
Visa/Delta	13 or 16
Bill Me Later [BIN 000001 ONLY]	16
Flex Cache	19

13 Response Data

13.1 RespCode Values

Response codes received that are not in this table should be treated as a general error not approved.

Key

Action	Description
Call	Call your Chase Paymentech Customer Service for assistance
Cust.	Try to resolve with customer or obtain alternate payment method
Fix	There is an invalid value being sent. Fix and resend
Resend	Send this transaction back at any time
Voice	Perform a voice authorization per instructions provided by Chase Paymentech
Wait	Wait 2-3 days before resending or try to resolve with the customer

Code	Definition	Status	Action
00	Approved	Approved	None
01	Call/Refer to Card Issuer	Decline	Voice
02	Refer to Card issuer's special conditions	Decline	Voice
03	Invalid Merchant Number	Error	Fix
04	Pickup	Decline	Cust.
05	Do Not Honor	Decline	Cust.
06	Other Error	Decline	Cust.
07	Stop Deposit Order	Decline	Cust.
08	Approved authorization, honor with identification	Approved	None
09	Revocation of Auth	Decline	Cust.
10	Default Call	Decline	Voice
11	Approved authorization, VIP Approval	Approved	None
12	Invalid Transaction Type	Decline	Cust.
13	Bad Amount	Decline	Fix
14	Invalid Credit Card Number	Decline	Fix
15	Default Call Low Fraud	Decline	Voice
16	Default Call Medium Fraud	Decline	Voice
17	Default Call High fraud	Decline	Voice
18	Default Call Unavailable Fraud	Decline	Voice
19	Re-enter Transaction	Error	Resend
20	Floor Low Fraud	Decline	Cust.

Code	Definition	Status	Action
21	Floor Medium Fraud	Decline	Cust.
22	Floor High fraud	Decline	Cust.
23	Floor Unavailable Fraud	Decline	Cust.
24	Validated	Approved	None
25	Verified	Approved	None
26	Pre-noted	Approved	None
27	No reason to decline	Decline	Cust.
28	Received and stored	Approved	None
29	Provided Auth	Approved	None
30	Invalid value in message	Error	Fix
31	Request received	Approved	None
32	BIN Alert	Approved	None
33	Card is expired	Decline	Cust.
34	Approved for partial	Approved	None
35	Zero Amount	Error	Fix
36	Bad Total Auth amount	Error	Fix
37	Invalid Secure Payment Data	Error	Fix
38	Merchant not MC SecureCode Enabled	Decline	Call
40	Requested Function not supported	Error	Call or Fix
41	Lost / Stolen	Decline	Cust.
43	Lost / Stolen Card	Decline	Cust.
50	Positive ID	Decline	Cust.
52	Processor Decline	Decline	Cust.
56	Restraint	Decline	Cust.
58	Transaction not permitted to terminal	Error	Call
59	Soft AVS	Decline	Cust.
60	Do Not Honor Low Fraud	Decline	Cust.
61	Do Not Honor Medium Fraud	Decline	Cust.
62	Do Not Honor High fraud	Decline	Cust.
63	Do Not Honor Unavailable Fraud	Decline	Cust.
64	CVV2/CVC2 Failure	Decline	Cust.
65	Invalid Amex CID	Decline	Cust.
66	Other Error	Error	Fix
68	Invalid CC Number	Error	Fix
69	Does not match MOP	Error	Fix

Code	Definition	Status	Action
71	No Account	Decline	Fix
72	Invalid Institution Code	Decline	Fix
73	Method of Payment is Invalid for Merchant	Error	Fix
74	Invalid Expiration Date	Decline	Cust.
75	Bad Amount	Error	Fix
77	Invalid Amount	Decline	Fix
78	Missing Companion Data	Error	Fix
79	Invalid Merchant	Error	Fix
80	Invalid MOP for division	Error	Fix
81	Call Low Fraud	Decline	Voice
82	Call Medium Fraud	Decline	Voice
83	Call High Fraud	Decline	Voice
84	Call Unavailable Fraud	Decline	Voice
85	Duplicated Order #	Error	Fix
86	Auth Recycle Host down	Error	Wait
87	Invalid Currency	Error	Fix
88	Invalid Purch. Level III	Error	Fix
89	Credit Floor	Decline	Cust.
91	Approved Low Fraud	Approved	None
92	Approved Medium Fraud	Approved	None
93	Approved High Fraud	Approved	None
94	Approved Fraud Service Unavailable	Approved	None
95	Invalid Data Type	Error	Fix
96	Invalid Record Sequence	Error	Fix
97	Percents Not Total 100	Error	Fix
98	Issuer Unavailable	Decline	Resend
99	No Answer / Unable to send	Error	Resend
A1	Payments Not Total Order	Error	Fix
A2	Bad Order Number	Error	Fix
A3	FPO Locked	Error	Wait
A4	FPO Not Allowed	Error	Call
A5	Auth Amount Wrong	Error	Fix
A6	Illegal Action	Error	Fix
A8	Invalid Start Date	Error	Fix
A9	Invalid Issue Number	Error	Fix

Code	Definition	Status	Action
B1	Invalid Transaction Type	Error	Fix
B2	Account Previously Activated	Decline	Cust
B3	Unable to void transaction	Error	Fix
B5	Not on file	Decline	Fix
B7	Fraud	Decline	Cust.
B8	Bad Debt	Decline	Cust.
B9	On Negative File	Decline	Cust.
BA	Under 18 Years Old	Decline	Cust.
BB	Possible Compromise	Decline	Cust.
BC	Bill To Not Equal To Ship To	Decline	Cust.
BD	Invalid Pre-approval Number	Decline	Cust.
BE	Invalid Email Address	Decline	Cust.
BF	PA ITA Number Inactive	Decline	Cust.
BG	Blocked Account	Decline	Cust.
BH	Address Verification Failed	Decline	Cust.
BI	Not on Credit Bureau	Decline	Cust.
BJ	Previously Declined	Decline	Cust.
BK	Closed Account, New Account Closed	Decline	Cust.
BL	Re-Authorization	Decline	Cust.
BM	Re-Authorization – No Match	Decline	Cust.
BN	Re-Authorization – Timeframes Exceeded	Decline	Cust.
BO	Stand In Rules	Decline	Cust.
BP	'Customer Service Phone Number required on Transaction Types 1 (MO/TO) and 2 (Recurring). MC Only	Error	Fix
BQ	'Issuer has flagged account as suspected fraud. (Discover Only)	Decline	Cust.
BR	Invalid MCC Sent	Error	Fix
BS	New Card Issued	Decline	Cust.
C1	Invalid Issuer	Decline	Cust.
C2	Invalid Response Code	Decline	Fix
C3	Excessive Pin Try	Decline	Cust.
C4	Over Limit	Decline	Cust.
C5	Over Freq Limit	Decline	Cust.
C6	Over Sav limit	Decline	Cust.
C7	Over Sav Freq	Decline	Cust.

Code	Definition	Status	Action
C8	Over Credit limit	Decline	Cust.
C9	Over Credit Freq	Decline	Cust.
D1	Invalid For Credit	Decline	Fix
D2	Invalid For Debit	Decline	Fix
D3	Rev Exceed Withdrawal	Decline	Cust.
D4	One Purchasing Limit	Decline	Cust.
D5	On Negative File	Decline	Cust.
D6	Changed Field	Decline	Fix
D7	Insufficient Funds	Decline	Cust.
D8	Encrypted Data Bad	Decline	Fix
D9	Altered Data	Decline	Fix
E3	Invalid Prefix	Decline	Fix
E4	Invalid Institution	Decline	Fix
E5	Invalid Cardholder	Decline	Fix
E6	BIN Block	Decline	Fix
E7	Stored	Approved	None
E8	Invalid Transit Routing Number	Error	Fix
E9	Unknown Transit Routing Number	Error	Fix
F1	Missing Name	Error	Fix
F2	Invalid Account Type	Error	Fix
F3	Account Closed	Error	Cust.
F4	No Account / Unable To Locate	Error	Fix
F5	Account holder Deceased	Error	Cust.
F6	Beneficiary Deceased	Error	Cust.
F7	Account Frozen	Error	Cust.
F8	Customer Opt Out	Error	Cust.
F9	ACH Non-Participant	Error	Cust.
G1	No Pre-note	Error	Fix
G2	No Address	Error	Fix
G3	Invalid Account Number	Error	Fix
G4	Authorization Revoked by Consumer	Error	Cust.
G5	Customer Advises Not Authorized	Error	Cust.
G6	Invalid CECP Action Code	Error	Fix
G7	Invalid Account Format	Error	Fix
G8	Bad Account Number Data	Error	Fix

Code	Definition	Status	Action
G9	No Capture	Decline	N/A
H1	No Credit Function	Decline	N/A
H2	No Debit Function	Decline	N/A
H3	Rev Exceed Withdrawal	Decline	Cust.
H4	Changed Field	Decline	N/A
H5	Terminal Not Owned	Decline	N/A
H6	Invalid Time	Decline	Fix
H7	Invalid Date	Decline	Fix
H8	Invalid Terminal Number	Decline	Fix
H9	Invalid PIN	Decline	Cust.
I1	Block activation failed – Card Range not set up for MOD 10	Error	Fix
I2	Block activation failed – email or fulfillment flags were set to Y	Error	Fix
I3	Declined – Issuance doesn't meet minimum amount	Declined	Cust
I4	Declined – no original auth found	Decline	Cust
I5	Declined – outstanding auth, funds on hold	Decline	Cust
I6	Activation amount incorrect	Decline	Fix
I7	Block activation failed- account not correct or block size not correct	Decline	Fix
I8	Mag stripe CVD value failed	Decline	Fix
I9	Max Redemption limit met	Decline	Fix
J1	No Manual Key	Decline	Fix
J2	Not Signed In	Decline	Fix
J3	Excessive Pin Try	Decline	Cust.
J4	No DDA	Decline	Fix
J5	No SAV	Decline	Fix
J6	Excess DDA	Decline	Cust.
J7	Excess DDA FREQ	Decline	Cust.
J8	Excess SAV	Decline	Cust.
J9	Excess SAV FREQ	Decline	Cust.
K1	Excess Card	Decline	Cust.
K2	Excess Card Freq	Decline	Cust.
K3	Reserved Future	Decline	N/A

Code	Definition	Status	Action
K4	Reserved Closing	Decline	N/A
K5	Dormant	Decline	Cust.
K6	NSF	Decline	Cust.
K7	Future RD Six	Decline	N/A
K8	Future RD Seven	Decline	N/A
K9	Transaction Code Conflict	Decline	Fix
L1	In Progress	Decline	Wait
L2	Process Unavailable	Error	Resend
L3	Invalid Expiration	Error	Fix
L4	Invalid Effective	Error	Fix
L5	Invalid Issuer	Decline	Fix
L6	Tran not allowed for cardholder	Decline	Cust.
L7	Unable to Determine Network Routing	Error	Call
L8	System Error	Error	Call
L9	Database Error	Error	Call
M1	Merchant Override Decline	Decline	Cust.
PA	Partial Approval	Approved	N/A
PB	Revocation of all Authorization	Decline	Cust.

13.2 AVS Response Code Values

Code	AVS Message
1	No address supplied
2	Bill-to address did not pass Auth Host edit checks
3	AVS not performed
4 or R	Issuer does not participate in AVS
5	Edit-error - AVS data is invalid
6	System unavailable or time-out
7	Address information unavailable
8	Transaction Ineligible for AVS
9	Zip Match / Zip4 Match / Locale match
A	Zip Match / Zip 4 Match / Locale no match
B	Zip Match / Zip 4 no Match / Locale match
C	Zip Match / Zip 4 no Match / Locale no match
D	Zip No Match / Zip 4 Match / Locale match

E	Zip No Match / Zip 4 Match / Locale no match
F	Zip No Match / Zip 4 No Match / Locale match
G	No match at all
H	Zip Match / Locale match
J	Issuer does not participate in Global AVS
JA	International street address and postal match
JB	International street address match. Postal code not verified.
JC	International street address and postal code not verified.
JD	International postal code match. Street address not verified.
N3	Address matches, ZIP not verified
N4	Address and ZIP code match (International only)
N5	Address not verified (International only)
N6	Address and ZIP code match (International only)
N7	ZIP matches, address not verified
UK	Unknown
X	Zip Match / Zip 4 Match / Address Match
Z	Zip Match / Locale no match
'Blank'	Not applicable (non-Visa)

13.3 CVV2 Response Code Values

Code	CVV2 (CVC2/ Discover CID) Description
M	CVV Match
N	CVV No match
P	Not processed
S	Should have been present
U	Unsupported by issuer
I	Invalid
Y	Invalid
'Blank'	Not applicable (non-Visa)

13.4 CAVV Response Code Values

Code	CAVV Description
'Blank'	CAVV Not Present
0	CAVV Not Validated due to erroneous data submitted

1	CAVV Failed Validation – Authentication Transaction
2	CAVV Passed Validation – Authentication Transaction
3	CAVV Attempt a 3-D Secure authentication value of 7 from the Issuer ACS indicates authentication was attempted. (Determined that the issuer ACS generated this value from the use of Visa CAVV key[s]).
4	CAVV Failed Validation – Attempt a 3-D Secure authentication value of 7 from Visa's ACS indicates that an authentication attempt was performed. (Determined that Visa generated this value from the use of CAVV key[s]).
5	Reserved for Future Use – NOT USED.
6	CAVV Not Validated – Issuer not participating in CAVV validation.
7	CAVV Failed Validation - Attempt (CAVV generated with Visa Key)
8	CAVV Passed Validation – Attempt (CAVV generated with Visa Key)
9	CAVV Failed Validation - Attempt (CAVV generated with Visa Key – Issuer ACS unavailable)
A	CAVV Passed Validation – Attempt (CAVV generated with Visa Key – Issuer ACS unavailable)
B	CAVV Passed Validation – Information only, no liability shift (CAVV with ECI = 7)
C	CAVV Not Validated – Attempt Issuer did not return a CAVV results code in the Authorization response.
D	CAVV Not Validated – Authentication – Issuer did not return a CAVV results code in the authorization response.
I	Invalid Security Data
U	Issuer does not participate or 3-D Secure data not utilized.

13.5 ProcStatus Values

Code	Description	Action
1	PWS_UNKNOWN_ERROR	Resend
2	PWS_NETWORK_ERROR	Resend
3	PWS_DB_ERROR Unknown Database Issues	Resend
40	Cannot Get to Authorizer Service	Resend
54	Industry type is currently not supported for merchant and bin	Fix
205	PWS_DB_EXCEPTION_ERROR	Resend
208	PWS_ERROR_FAILED_TO_CONNECT	Resend

Code	Description	Action
301	PWS_NW_OPEN_ERROR	Resend
303	PWS_NW_READ_ERROR	Resend
328	PWS_ERROR_BAD_REVERSAL_AMOUNT (An invalid amount submitted on a Partial Void Request)	Fix
329	PWS_ERROR_BAD_REQUEST_AMOUNT	Fix
330	PWS_ERROR_ALREADY_CAPTURED	Fix
331	PWS_ERROR_INVALID_ACTION	Fix
333	PWS_ERROR_MISSING_TRANSACTION_REFERENCE_INDEX	Fix
335	PWS_ERROR_SPLIT_AUTH_NOT_ALLOWED_ALREADY_MARKE D	Fix
348	PWS_DID_NOT_ALLOW_A_CAPTURE_REQUEST_BECAUSE_TH E_ORIGINAL_AUTH_WAS_NOT_SUCCESSFUL (Cannot Void a Transaction in which the Mark for Capture Failed)	Fix
350	The amount requested cannot be zero	Fix
351	This industry type does not allow a capture greater than the value of the auth	Fix
355	There is nothing to capture [This error is returned when a Capture attempt is made on prior authorization, but there is no amount left to capture].	Fix
400	PWS_MANDATORY_FIELDS_ERROR	Fix
410	FE_NETWORK_ERROR (cannot connect to eHost)	Resend
411	FE_INTERRUPTED_SESSION (i/o problem while connecting to eHost)	Resend
516	The Merchant ID / Acquiring BIN ID is invalid or missing. Message rejected	Fix
518	This merchant is not active until … [This error is returned when a Merchant Account has been setup, but with an Activation date in the future of the present date].	Call Customer Service
519	This merchant is inactive	Call Customer Service
521	eHost has received a badly formatted message [This error is returned when required fields are missing]	Fix
523	An invalid TID was received {Terminal ID]	Fix
801	PWS_ERR_VALIDATION_AMOUNT	Fix
803	PWS_ERR_VALIDATION_AVSADDRESS	Fix
804	PWS_ERR_VALIDATION_AVSZIPCODE	Fix
806	PWS_ERR_VALIDATION_BIN	Fix
811	PWS_ERR_VALIDATION_CUSTOMERADDR	Fix
812	PWS_ERR_VALIDATION_CUSTOMEREMAIL	Fix
814	PWS_ERR_VALIDATION_CUSTOMERNAME	Fix
817	PWS_ERR_VALIDATION_CUSTOMERPHONE	Fix
818	PWS_ERR_VALIDATION_CVV2	Fix

Code	Description	Action
822	PWS_ERR_VALIDATION_ISSUENUM	Fix
823	PWS_ERR_VALIDATION_LANGUAGE	Fix
825	PWS_ERR_VALIDATION_MERCHANTID	Fix
826	PWS_ERR_VALIDATION_ORDERDESCRIPTION	Fix
827	PWS_ERR_VALIDATION_ORDERID	Fix
831	PWS_ERR_VALIDATION_TAXAMT	Fix
832	PWS_ERR_VALIDATION_TAXINCLUDED	Fix
833	PWS_ERR_VALIDATION_TERMINALID	Fix
834	PWS_ERR_VALIDATION_TRANSDATE	Fix
835	PWS_ERR_VALIDATION_TRANSTIME	Fix
836	PWS_ERR_VALIDATION_ECOM	Fix
838	PWS_ERR_VALIDATION_ACNUMBER	Fix
839	PWS_ERR_VALIDATION_PAN_LUHN	Fix
840	PWS_ERR_VALIDATION_PAN_LENGTH	Fix
841	PWS_ERR_VALIDATION_PAN_RANGE	Fix
842	PWS_ERR_VALIDATION_EXP_DATE_FORMAT	Fix
844	PWS_ERR_VALIDATION_EXP_DATE_TOO_NEW	Fix
845	PWS_ERR_VALIDATION_START_DATE_FORMAT	Fix
846	PWS_ERR_VALIDATION_START_DATE_TOO_NEW	Fix
847	PWS_ERR_VALIDATION_PAN_FORMAT	Fix
848	PWS_ERR_VALIDATION_CURRENCY_FORMAT	Fix
849	PWS_ERR_VALIDATION_CURRENCY_UNSUPPORTED	Fix
850	PWS_ERR_VALIDATION_CURRENCY_BAD_EXPONENT	Fix
851	PWS_ERR_VALIDATION_MERCHANT_UNSUPPORTED	Fix
852	PWS_ERR_VALIDATION_BRAND_UNSUPPORTED	Fix
853	PWS_ERR_VALIDATION_BRAND_PAN_MISMATCH	Fix
881	The LIDM you supplied # does not match with any existing transaction (Cannot void or Mark a Transaction because the TxRefNum does match a transaction)	Fix
882	LOCKED_DOWN (Cannot mark or unmark transaction)	Fix
9720	Soft Desc: Merchant not activated for soft descriptors	Fix
9721	Soft Desc: Merchant Name is required if soft descriptor data is sent	Fix
9722	Soft Desc: Merchant Name exceeds max length of [%s] for %s transactions	Fix
9723	Soft Desc: [%s] cannot contain leading spaces	Fix
9724	Soft Desc: [%s] exceeds max length of [%s]	Fix

Code	Description	Action
9725	Soft Desc: Product Description cannot be present if Merchant Name is > %s	Fix
9726	Soft Desc: Product Description length cannot exceed [%s] if Merchant Name length is between %s and %s	Fix
9727	Soft Desc: Too many Merchant descriptors. Never send more than one of the following: City, phone, url OR email	Fix
9728	Soft Desc: [%s] is not allowed for ECP transactions	Fix
9729	Soft Desc: Invalid format for Merchant Phone. Must be nnn-nnn-aaaa or nnn-aaaaaaaa	Fix
9737	Gateway is Down	Resend
9738	Database Connection Problem: Cannot acquire Database Connection	Resend
9743	Pcard 3 data was sent in parent split, but is missing in current request	Fix
9744	If Alt Tax is sent Alt Tax ID is required	Fix
9745	Three reasons could result in this error: - Pcard 3 data can only be sent with MC and VI cards. - Pcard 3 data cannot be sent on this request type. - Pcard 3 data can only be sent with US or Canadian currency.	Fix
9746	Line item count must be between 1 and 98 inclusive	Fix
9747	Line item detail number [%s] is missing	Fix
9748	Cannot send Pcard 3 data without sending Pcard 2 field	Fix
9749	Minimal Pcard 3 base data missing or invalid	Fix
9750	Minimal Pcard 3 line item data missing or invalid on index	Fix
9751	Line Item Count does not match the number of line items sent	Fix
9752	Invalid debit indicator for Bin 000002 in index. Must be 'D' or 'C'	Fix
9753	Invalid Gross / Net for Bin 000002 in index. Must be 'Y' or 'N'	Fix
9754	Amount hash error, negative total on line item data index	Fix
9755	Amount hash error on line item data index. Total = [%s] Hash = [%s]	Fix
9756	Detail totals do not match requested amount	Fix
9757	Invalid Country Code	Fix
9758	Invalid Unit of Measure in index	Fix
9759	Gateway Operation In-Progress For This Order	Action Complete do not resend
10005	Error communicating with the host	Fix
10011	Response timed out waiting for Authorization Host	Resend
10096	Invalid Card Number	Fix
10204	Invalid AVS ZIP Code	Fix

Code	Description	Action
10332	Invalid Message Format. Transaction was flagged as an eCommerce Industry Type but No ECOrderNum was sent	Fix
10333	The ECOrderNum or MailOrderNum was all Zero's or All Spaces. These are not valid	Fix
10334	Invalid Card Number	Fix
10336	Transaction Amount to Large	Fix
10337	Transaction Amount to Small	Fix
10349	Host eFalcon check requested from Tampa [BIN 000002] – This functionality is not supported on this platform	Fix
11001	Locked Down: Unable to Perform a Partial Void on Industry Type: [RE].	Fix
All other 10000 – 11000	GATEWAY SYSTEM ERROR CONDITIONS This encompasses various processing errors.	Resend
	Profile Errors	
9550	Invalid Customer Reference Number From Order Indicator	Fix
9551	Invalid Customer Reference Number	Fix
9552	System Failure. Unable To Perform Customer Profile Request at This Time.	Call
9553	Invalid Action Indicator	Fix
9555	Invalid BIN	Fix
9556	Invalid Merchant ID	Fix
9557	Invalid Name	Fix
9558	Invalid Address	Fix
9559	Invalid Address 2	Fix
9560	Invalid City	Fix
9561	Invalid State	Fix
9562	Invalid ZIP	Fix
9563	Invalid Email	Fix
9564	Invalid Phone	Fix
9565	Invalid Order Description	Fix
9566	Invalid Amount	Fix
9567	Invalid Account Type Indicator	Fix
9568	Invalid Account Number	Fix
9569	Invalid Account Expire Date	Fix
9570	Invalid ECP Account DDA	Fix
9571	Invalid ECP Account Type Indicator	Fix
9572	Invalid ECP Account Route	Fix
9573	Invalid ECP Bank Payment Delivery Method	Fix
9574	Invalid Switch Solo Start Date	Fix
9575	Invalid Switch Solo Issue Number	Fix
9576	Unable to Perform Profile Transaction. The Associated Transaction Failed.	Call
9577	Invalid Order Override Indicator	Fix

Code	Description	Action
9578	Merchant-Bin combination is not allowed to perform profile transactions.	Call
9579	Merchant-Bin is not active.	Call
9580	Cannot process profile for Cust Ref Num and MID combination. A database error has occurred	Call
9581	Cannot process profile. Profile does not exist for Cust Ref Num and MID.	Fix
9582	Cannot process profile. Profile already exists for Cust Ref Num and MID.	Fix
9583	Missing Switch Solo Account Information. Either start date or issue number is required.	Fix
9584	Missing Electronic Check Account Information.	Fix
9585	Missing Credit Card Account Information.	Fix
9587	Auto-Gen Cust Ref Num Error.	Call
9588	Unable to Determine Profile Action from Auth Request	Fix
9589	Cannot Create Profile: A Customer Profile Name is Required	Fix
9711	Too many transactions to process	Wait & Resend
9712	Request timeout - Please try again	Resend
9713	Invalid MIME header - Merchant ID in MIME does not match XML message	Fix
9714	Invalid MIME header- Trace number must be between 1 and 9999999999999999	Fix
9715	The retry request did not match the original request for this trace number	Fix
	IP Authentication Errors	
9716	Security Information is Missing	Call Customer Service
9717	Security Information - agent/chain/merchant is missing	Call Customer Service
	Web Server Errors	
20400	Invalid Request	Fix
20403	Forbidden: SSL Connection Required	Fix
20408	Request Timed Out	Resend
20412	Precondition Failed: Security Information is missing	Call Customer Service
20500	Internal Server Error	Resend
20502	Connection Error	Resend
20503	Server Unavailable: Please Try Again Later	Resend

13.6 ProfileProcStatus Response Values

Code	Definition	Action
0	Profile Action Successful	None
9550	Invalid Customer Reference Number From Order Indicator	Fix
9551	Invalid Customer Reference Number	Fix
9552	System Failure. Unable To Perform Customer Profile Request at This Time.	Call
9553	Invalid Action Indicator	Fix
9555	Invalid BIN	Fix
9556	Invalid Merchant ID	Fix
9557	Invalid Name	Fix
9558	Invalid Address	Fix
9559	Invalid Address 2	Fix
9560	Invalid City	Fix
9561	Invalid State	Fix
9562	Invalid ZIP	Fix
9563	Invalid Email	Fix
9564	Invalid Phone	Fix
9565	Invalid Order Description	Fix
9566	Invalid Amount	Fix
9567	Invalid Account Type Indicator	Fix
9568	Invalid Account Number	Fix
9569	Invalid Account Expire Date	Fix
9570	Invalid ECP Account DDA	Fix
9571	Invalid ECP Account Type Indicator	Fix
9572	Invalid ECP Account Route	Fix
9573	Invalid ECP Bank Payment Delivery Method	Fix
9574	Invalid Switch Solo Start Date	Fix
9575	Invalid Switch Solo Issue Number	Fix
9576	Unable to Perform Profile Transaction. The Associated Transaction Failed.	Call
9577	Invalid Order Override Indicator	Fix
9578	Merchant-Bin combination is not allowed to perform profile transactions.	Call
9579	Merchant-Bin is not active.	Call
9580	Cannot process profile for Cust Ref Num and MID combination. A database error has occurred	Call
9581	Cannot process profile. Profile does not exist for Cust Ref Num and MID.	Fix
9582	Cannot process profile. Profile already exists for Cust Ref Num and MID.	Fix
9583	Missing Switch Solo Account Information. Either start date or issue number is required.	Fix
9584	Missing Electronic Check Account Information.	Fix
9585	Missing Credit Card Account Information.	Fix
9587	Auto-Gen Cust Ref Num Error.	Call
9588	Unable to Determine Profile Action from Auth Request	Fix

Code	Definition	Action
9589	Cannot Create Profile: A Customer Profile Name is Required	Fix

14 Frequently Asked Questions

14.1 General

14.1.1 I am not permitted to modify my trusted server (cacerts) file… how can I direct the SDK to use a different cacerts file?

Uncomment the following lines in your linehandler.properties file, and edit them to include the location and passphrase for your cacerts file.

engine.ssl.trustore.filename=C:/jdk1.3.1_03/jre/lib/security/cacerts ← substitute your file path

engine.ssl.trustore.passphrase=changeit ← substitute your passphrase

14.1.2 I get exceptions when I try to build the sample code… how do I fix?

Use the provided build scripts (build.bat for Windows, build.sh for UNIX and Linux). If the problem persists, it is most likely an environmental issue. Confirm the required environment variables have been set (PAYMENTECH_HOME, JAVA_HOME).

14.1.3 How do I enable automatic retries for transactions?

The Java SDK will automatically retry transactions when an error occurs, if the application has set a Trace Number in the request object using the setTraceNumber() method.

14.1.4 I see an exception "Algorithm TLS not available" when I try to execute a transaction. What is causing this error?

The Java Security Provider must be specified in the java.security file of your Java JDK installation (see Section 3.1.2 "Java Security Provider Settings").