PayPal™

Payflow SDK for Java Developer's Guide

For Professional Use Only
Currently only available in English.

A usage Professional Uniquement
Disponible en Anglais uniquement pour l'instant.

Last updated: December 2007

Payflow SDK for Java Developer's Guide

Document Number: 200031.en_US-200712

Contents

Preface

This Document

Payflow SDK for Java Developer's Guide describes the Payflow Software Development Kit (SDK) for Java.

The Payflow SDK for Java enables you to develop web and desktop applications using Java and to directly integrate them with Payflow services.

Intended Audience

This guide assumes that its readers:

- Are experienced web or application developers
- Use Java for creating applications
- Have a background in payments services

What's New

This version of the Payflow SDK contains the following new features:

- New Console samples:
  - DoSaleComplete - Fully commented Sale example
  - DoSwipe

  For more information, see Chapter 4, "Transaction Samples"
- Organizes transaction parameter/object mapping alphabetically by transaction parameter name

  For more information, see Appendix A, "Transaction Parameter Mapping"
- Removes certificate path from the SDK

Where to Go For More Information

NOTE: All the documentation described in this section is available from the PayPal Manager Documentation page.

'Must Have' Documentation

You must have the following documentation.

- SDK Class Descriptions

  The `docs/payflow_API` directory of the Payflow SDK contains descriptions of the SDK classes.

- *Developer's Guide*

  This Java developer's guide is not the complete source of all the information you need to develop applications using the Payflow SDK. You must use the appropriate *Developer's Guide* below along with this guide:

  – Use *Payflow Pro Developer's Guide* if you are developing applications that will send transactions to processors other than the PayPal processor

  – Use *Website Payments Pro Payflow Edition Developer's Guide* if you are developing applications that will send transactions to the PayPal processor

  The *Developer's Guide* provides detailed descriptions of all Payflow SDK name-value pair parameters. In addition, it contains testing data and error codes.

When Developing XML Applications

If you are developing applications in XML, you also will need the following documentation.

XMLPay Developer's Guide

- Use *Payflow Pro XMLPay Developer's Guide* if you are developing applications for Payflow Pro
- Use *Website Payments Pro Payflow Edition XMLPay Developer's Guide* if you are developing applications for Website Payments Pro Payflow Edition

The *XMLPay Developer's Guide* defines the XMLPay syntax, contains examples, and includes the XML schemas.

Organization of This Document

This guide is organized as follows:

Chapter 1, "Getting Started With the Payflow SDK for Java," provides information on additional software requirements and describes the SDK package.

Chapter 2, "Getting Started With Transactions," describes the transaction flow at a high level, helps you choose a mode for sending transaction data, and provides the key steps for implementing a simple sale transaction.

Chapter 3, "Object Descriptions," provides additional information about some of the more important objects.

Chapter 4, "Transaction Samples," provides information on the code samples included in the Payflow SDK.

Appendix A, "Transaction Parameter Mapping," maps name-value pair transaction parameters to Payflow SDK data object variable names.

Appendix B, "Header Parameter Mapping," maps header parameters to Payflow SDK objects.

Appendix C, "Logging, Error Codes, and Exceptions," provides logging information.

For each payments transaction type, this guide provides supporting samples. Detailed descriptions of all Payflow SDK name-value pair parameters are provided in the *Developer's Guide*.

Customer Service

For problems with transaction processing or connections, contact Customer Service by opening a ticket on the Contact Support tab at http://www.paypal.com/mts.

Before contacting Customer Service, ensure that you have done the following:

- If you are having problems compiling, make sure that you have installed Java 2 Platform, Standard Edition (J2SE) 1.4 or later.

  You can get the J2SE from:
  - Sun Developer Network at http://java.sun.com
  - IBM at http://www-128.ibm.com/developerworks/java.
- If you are having problems deploying the application, make sure that you have installed the J2SE Java Runtime Environment (JRE) version 1.4 or later.

  You can get the JRE from:
  - Sun Developer Network at http://java.sun.com
  - IBM at http://www-128.ibm.com/developerworks/java.
- If you are having problems running an application, make sure that you have installed the Xerces XML parser.
  - Download the Xerces Java Parser version 2.7.1 from Apache's archive website at http://archive.apache.org/dist/xml/xerces-j.
  - Extract the Xerces parser and copy `xercesImpl.jar` and `xml-apis.jar` to the `lib` folder of the SDK.
- If you are having problems with a specific type of transaction, refer to the samples provided for that transaction type.

- If you are having problems while running an application, enable logging. For instructions on how to enable logging, see "Enabling Logging in the Payflow SDK for Java" on page 56. When reporting your problem, send your log files to us, along with your questions.

Revision History

Revision history for *Payflow SDK for Java Developer's Guide.*

TABLE P.1 Revision History

Date	Description
December 2007	Remove certificate information. Emphasize use of this guide with Developer's Guide. Add DoSaleComplete samples. Add DoSwipe samples. Update objects information.
July 2006	Change VeriSign to PayPal. Update product name to Payflow SDK. Replace PFPro with Payflow. Change PFProUtility to PayflowUtility. Change PFProConstants to PayflowConstants. Change PFProConnectionData to PayflowConnectionData.
June 2006	Removed all references to SubmitCommit and X-VPS-CLIENT-DURATION.
May 2006	Revision history started. Beta version of documentation.

1 Getting Started With the Payflow SDK for Java

In This Chapter

This chapter helps you get started with the Payflow SDK and consists of the following sections:

Prerequisites

Before installing the Payflow SDK, be sure that you:

- Use this guide along with the appropriate *Developer's Guide*. See "Where to Go For More Information" on page 8," for details on the documentation you must have.
- Install Java 2 Platform, Standard Edition (J2SE) 1.4 or later on the system that will be used for developing the application.

  You can get the J2SE from:
  - Sun Developer Network at: http://java.sun.com
  - IBM at http://www-128.ibm.com/developerworks/java.
- Install JRE 1.4 or later on the system that will be used for deploying the application.

  You can get the JRE from:
  - Sun Developer Network at http://java.sun.com
  - IBM at http://www-128.ibm.com/developerworks/java.
- Download the Xerces Java Parser version 2.7.1 from Apache's archive website at http://archive.apache.org/dist/xml/xerces-j.

  Extract the Xerces parser and copy `xercesImpl.jar` and `xml-apis.jar` to the `lib` folder of the Payflow SDK.

Payflow SDK for Java Packages

The Payflow SDK package contains libraries for building Payflow SDK applications, documentation, and sample code.

Figure 1.1 shows the directory structure of this package.

FIGURE 1.1 Directory structure of the installed package

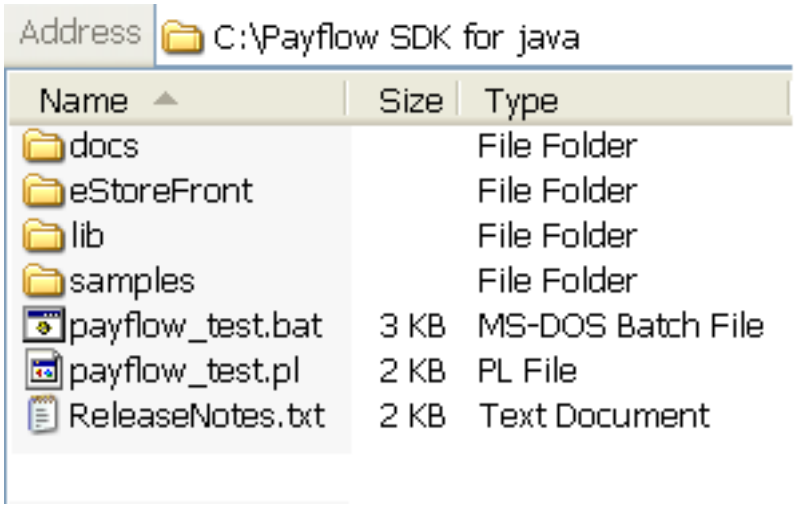


TABLE 1.1 Payflow SDK for Java directory contents

Directory/File	Description
`docs`	`readme.txt` Contains configuration, set-up, and last minute information. `payflow_API` Contains SDK class descriptions.
`eStoreFront`	Web-based sample store
`lib`	Contains `payflow.jar`, the Payflow SDK for Java API library.
`samples`	Contains samples that can be run from the console. These samples may be used as a starting point for learning about the Payflow SDK. The Payflow SDK provides Windows batch files (.bat) and Perl scripts (.pl) that allow you to run transactions from the console. Each sample contains a description and instructions on how to run it. For more information about individual samples, see Chapter 4, "Transaction Samples."
`payflow_test.bat`	Batch script to compile and execute the samples on Windows.
`payflow_test.pl`	Perl script to compile and execute the samples.
`ReleaseNotes.txt`	Includes information specific to the release.

Verifying the Installation

To verify the installation, run one of the sample programs in the `samples` directory.

2 Getting Started With Transactions

In This Chapter

This chapter describes the transaction flow at a high level, helps you choose a mode for sending transaction data, and provides the key steps for implementing a simple sale transaction.

This chapter consists of the following sections:

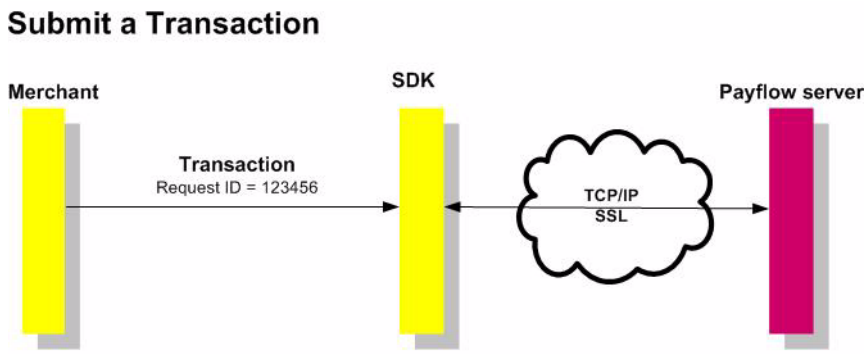
NOTE: Do not copy code samples from this guide. Code formatting may prevent the samples from compiling.

Transaction Flow

The flow of a Payflow SDK transaction consists of two key elements:

- Submitting a transaction
- Request ID

FIGURE 2.1 Transaction flow



Request ID

A request ID is a unique value you send to the Payflow server during a submit transaction. The request ID helps you and the Payflow server keep track of a particular transaction and allows you to resubmit it, if necessary, without creating a duplicate transaction. If you do not send a request ID with your submit transaction, your transaction fails.

For a failed submit transaction, you can resubmit the transaction as long as you use the original request ID. If you do not send the original request ID, the Payflow server thinks you are sending a new transaction. As a result, the buyer may be billed twice.

Submitting a Transaction

You submit a transaction to pass the data parameters to the Payflow server. If the Payflow server does not respond, you may resubmit the transaction as long as you use the original request ID.

Choosing Your Transaction Mode

The Payflow SDK for Java supports the following modes for sending a transaction:

- Object-based
- Parameter-based
  - Using name-value pairs
  - Using XMLPay 2.0

All modes offer the same functionality. This section describes each mode and helps you decide which is right for you.

Object-Based

With the object-based transaction mode, you use data objects to send and receive transaction data from the Payflow server. You set the transaction parameters in data objects, and then the SDK constructs the request string from the data objects and sends it to the Payflow server. Because the SDK constructs the request string, there is less chance of an error in the request string.

For example, using this mode to send a sale transaction, you set the data objects as follows:

```
...
BillTo bill = new BillTo();
bill.setStreet("123 Main St.");
bill.setZip("12345");
inv.setBillTo(bill);
...
SaleTransaction trans = new SaleTransaction(user, connection, inv,
                                card, PayflowUtility.getRequestId());
```

Parameter-Based

With the parameter-based transaction mode, you send and receive the transaction data in a request string from the Payflow server. The parameter-based mode accepts the data as name-value pairs or in XMLPay 2.0 format.

Name-value pairs

Submit transaction data as name-value pairs separated by an ampersand (&).

For example, using name-value pairs to send a sale transaction, you set the transaction data string as follows:

```
PayflowAPI pa = new PayflowAPI();
...
String request = "...&STREET=123 Main St.&ZIP=12345...";
String requestId = pa.generateRequestId();
pa.submitTransaction(request, requestId);
```

For more information on the values that can be included in the request string, see the *Developer's Guide*.

XMLPay

XMLPay allows you to submit transaction data in a XML format conforming to the XMLPay 2.0 schema.

For example, using XMLPay to send a sale transaction, you set the transaction data as follows:

```
PayflowAPI pa = new PayflowAPI();
...
String request = "<?xml version=\"1.0\"?> \
     ...
     <BillTo> \
        <PONum>12345</PONum> \
        <Address> \
           <Street>123 Main St.</Street> \
           <Zip>12345</Zip> \
        </Address> \
        ...
     </BillTo>";
String requestId = pa.generateRequestId();
pa.submitTransaction(request, requestId);
```

For more information about XMLPay, see the *XMLPay Developer's Guide*.

Code Sample

Complete code for a simple sale transaction is provided for each transaction mode.

- Object-based: See `DOSaleComplete.java` in `samples/com/paypal/samples/dataobjects/basictransactions` directory.
- Parameter-based
  - Name-value pairs: see `NVPSale.java` in `samples/com/paypal/samples/namevaluepairs`.
  - XMLPay: see `XMLPaySale.java` in `samples/com/paypal/samples/xmlpay`.

For a list of code samples included with the SDK, see Chapter 4, "Transaction Samples."

Steps for a Simple Sale Transaction

This section describes the steps necessary to perform a simple sale transaction using the object-based transaction mode.

The following are the key steps for a basic sale transaction found in `DOSaleComplete.java`:

1. Set the properties of the `SDKProperties` object.

   This object refers to connection related information such as Payflow server host address, Payflow server port number, and so forth. For more information about the `SDKProperties` object, see "SDKProperties Object" on page 23.

   For example, the following code specifies the host address used for test transactions, and other properties.

   ```
   SDKProperties.setHostAddress("pilot-payflowpro.verisign.com");
   SDKProperties.setHostPort(443);
   SDKProperties.setTimeOut(20);
   ```

2. Create the required data objects for the `SaleTransaction` object and populate them with the appropriate data.

   Provide information such as connection data, customer billing address, and credit card number. For the list of data objects typically used for a sale transaction, see "DOSale.java and DoSaleComplete.java" on page 35.

3. Create a `SaleTransaction` object with a unique request ID.

   You create the request ID using `PayflowUtility.getRequestId()`, or you can write your own implementation for generating a unique string.

   ```
   SaleTransaction trans = new SaleTransaction(user, connection, inv, card,
                                       PayflowUtility.getRequestId());
   ```

4. Submit the transaction, and receive the response from the Payflow server.

   ```
   Response resp = trans.submitTransaction();
   ```

   NOTE: The original request ID and response ID are set by the SDK in the `SaleTransaction` object.

5. Check the `Response.TransactionResponse` object to verify that the submit completed successfully.

   If `RESULT` is 0, then the submit completed successfully. For more information on the `Response` object, see "UserInfo Object" on page 24.

If the submit completed successfully, you have a valid transaction.

3 Object Descriptions

In This Chapter

This chapter provides additional information about some of the more important objects. For complete information, see the class descriptions in the `samples/docs/payflow_API` directory.

This chapter consists of the following sections:

SDKProperties Object

The `SDKProperties` object allows you to set properties of the SDK such as the host address and timeout.

Modifications to the `SDKProperties` object must occur at the beginning of your code. If you modify the `SDKProperties` object later, it can lead to unpredictable SDK behavior.

The constructors of certain classes can override properties of the `SDKProperties` object. The constructors for the `PayflowAPI` or `PayflowConnectionData` objects can override `hostAddress`, `timeout`, and other properties.

Connection Parameters

The following table lists some of the important parameters of the `SDKProperties` object.

TABLE 3.1 Connection parameters

Parameter	Description
`setHostAddress()`	Test: `pilot-payflowpro.verisign.com` (Default) Live: `payflowpro.verisign.com`
`setTimeOut()`	45 (seconds) (Default)
`setProxyAddress()`	`myproxy.mydomain.com` OR empty string/null (Optional)

***TABLE 3.1 Connection parameters** (Continued)*

Parameter	Description
`setProxyPort()`	HTTP proxy port OR 0 (Optional)
`setProxyLogin()`	proxy login name OR empty string/null (Optional)
`setProxyPassword()`	proxy password OR empty string/null (Optional)

NOTE: If these properties are not set in the code, default values are used.

Response Object

The `Response` object contains the following:

- `TransactionResponse` – Contains normal transaction parameters such as `RESULT`, `PNREF`, and so forth.
- `FraudResponse` – Contains fraud parameters such as `PREFPSMSG`, `POSTFPSMSG`, and so forth.
- `RecurringResponse` – Contains recurring parameters such as `PROFILEID`, `RPREF`, and so forth.

NOTE: If you have signed up for Fraud Protection and Recurring Billing Services, the SDK will return parameters in the Fraud Protection and Recurring Billing objects. Otherwise, these objects contain no data.

UserInfo Object

The `UserInfo` object contains the user account information needed to authenticate a user before performing a transaction. This object is required when using the object-based transaction mode. Table 3.2, "UserInfo information" describes the required parameters.

TABLE 3.2 UserInfo information

Parameter	Meaning
`User`	If you set up one or more additional users on the account, this value is the ID of the user authorized to process transactions. If, however, you have not set up additional users on the account, `User` has the same value as `Vendor`.
`Vendor`	Your merchant login name that you created when you registered for the Payflow Pro or Website Payments Pro Payflow Edition account.

TABLE 3.2 UserInfo information *(Continued)*

Parameter	Meaning
`Partner`	The ID provided to you by the authorized PayPal reseller who registered you for Payflow Pro or Website Payments Pro Payflow Edition. If you purchased your account directly from PayPal, use `PayPal`.
`Password`	The 6- to 32-character case-sensitive password you created while registering for the account.

Transaction Objects

Table 3.3 shows the transaction objects in the `paypal.payflow.*` namespace that are supported by the object-based transaction mode of the Payflow SDK for Java. For example, the sale transaction type is in the following namespace:

```
paypal.payflow.SaleTransaction
```

TABLE 3.3 Transaction objects

Transaction Type	Transaction Object
Sale	`SaleTransaction`
Authorize	`AuthorizationTransaction`
Capture	`CaptureTransaction`
Credit	`CreditTransaction`
Void	`VoidTransaction`
Voice Authorization	`VoiceAuthTransaction`
Inquiry	`InquiryTransaction`
Fraud Protection	
FraudReview	`FraudReviewTransaction`
Recurring Billing	
Add	`RecurringAddTransaction`
Modify	`RecurringModifyTransaction`
Cancel	`RecurringCancelTransaction`

TABLE 3.3 Transaction objects *(Continued)*

Transaction Type	Transaction Object
Inquiry	`RecurringInquiryTransaction`
ReActivate	`RecurringReActivateTransaction`
Payment	`RecurringPaymentTransaction`
Buyer Authentication	
Buyer Authentication	`BuyerAuthTransaction`
Validate Authentication	`BuyerAuthVATransaction`
Verify Enrollment	`BuyerAuthVETransaction`

NOTE: You must sign up for Fraud Protection and Recurring Billing Services to use their respective functionality in the Payflow SDK for Java.

Buyer Authentication requires Fraud Protection Services.

Data Objects

Table 3.4 lists and describes the important Payflow SDK for Java data objects in the `paypal.payflow.*` namespace. For example, the `CreditCard` data object is in the following namespace:

```
paypal.payflow.CreditCard
```

NOTE: For complete information on data objects, see the class descriptions in `samples/docs/payflow_API` directory

TABLE 3.4 Important data objects

Data object	Description
`ACHTender`	Contains the parameters for ACH tender
`BankAcct`	Contains the parameters for bank account information
`BillTo`	Contains the parameters for billing address information
`BrowserInfo`	Contains the parameters for browser information
`CardTender`	Contains the parameters for credit card tender
`CheckPayment`	Contains the parameters for a check payment
`CheckTender`	Contains the parameters for check tender
`ClientInfo`	Contains the parameters for client information

TABLE 3.4 Important data objects *(Continued)*

Data object	Description
CommitResponse	Contains the parameters for the Commit response from the Payflow server
CreditCard	Contains the parameters for a credit card
CustomerInfo	Contains the parameters for customer information
FraudResponse	Contains the parameters for a fraud response
Invoice	Contains the parameters for a payment invoice
LineItem	Contains the parameters for line items such as those for purchase cards
PayflowConnectionData	Contains the parameters for establishing a connection with the Payflow server
PurchaseCard	Contains the parameters for a purchase card
RecurringInfo	Contains the parameters for a recurring transaction
RecurringResponse	Contains the parameters for the response to a recurring transaction
Response	Contains the parameters for the response from the Payflow server
ShipTo	Contains the parameters for shipping address information
SwipeCard	Contains the parameters for a swipe card
TransactionResponse	Contains the parameters for a normal transaction response
UserInfo	Contains the parameters for user information
ExpressCheckoutResponse	Base class of all express checkout response classes
ExpressCheckoutRequest	Base class of all express checkout request classes
ECDoResponse	Contains the parameters for an express checkout do response
ECDoRequest	Used for express checkout do operation
ECGetRequest	Used for express checkout get operation
ECSetRequest	Used for express checkout set operation
ECGetResponse	Contains the parameters for an express checkout get response
BuyerAuthResponse	Contains the parameters for a buyer authorization response
BuyerAuthStatus	Contains the parameters for buyer authorization status

4 Transaction Samples

In This Chapter

This chapter provides information on the code samples included in the Payflow SDK for Java. All source files are located in the `samples` directory of the Payflow SDK.

This chapter consists of the following sections:

Sample Naming Conventions

The names of all object-based samples start with "DO" (for Data Object), for example: `DOSaleComplete.java`.

All the parameter-based name-value pair samples start with the prefix "NVP" (for name-value pair), for example `NVPSale.java`.

All the parameter-based XMLPay-based samples start with the prefix "XMLPay," for example `XMLPaySale.java`.

Web Samples

The eStoreFront contains samples demonstrating Buyer Authentication and Express Checkout. For more information, see `readme_eStoreFront.txt` in the `docs` directory.

See the "Complete Listing of Console Samples" on page 30 for code snippets of the different data objects you can integrate into your web application.

Complete Listing of Console Samples

The console samples are contained in the Samples directory of the Payflow SDK.

Table 4.1 lists samples by transaction type and provides a brief description of the sample.

For example, DOAuth.java is a sample that illustrates a simple authorization transaction. It is in the following namespace:

```
paypal.samples.dataobjects.basictransactions
```

NOTE: All samples described in this chapter are for simple transactions only. For complex transactions, you may have to send additional parameters and you may receive additional parameters in the response.

TABLE 4.1 Samples listed by transaction type

Transaction Type	Source Code File Name	When to Use
Basic Transactions **paypal.samples.dataobjects.basictransactions**		
Authorize	`DOAuth.java`	Doing a simple authorization transaction
Delayed Capture	`DOCapture.java`	Doing a simple capture transaction
Credit	`DOCredit.java`	Doing a simple credit transaction
Inquiry	`DOInquiry.java`	Doing a simple inquiry transaction
Reference Credit	`DOReferenceCredit.java`	Doing a reference credit transaction
Sale	`DOSaleComplete.java`	Doing a sale transaction **NOTE:** This is a fully commented sale transaction, complete with business logic, that you should examine first
Sale	`DOSale.java`	Doing a simple sale transaction
Swipe	`DOSwipe.java`	Doing a simple sale transaction using SWIPE
Force/Voice Authorization	`DOVoiceAuth.java`	Doing a simple voice authorization transaction
Void	`DOVoid.java`	Doing a simple void transaction
Fraud Protection **paypal.samples.dataobjects.fraud**		
NOTE: You must sign up for the Fraud Protection Service to be able to use the Fraud objects in the Payflow SDK		
Fraud Filters	`DOFraudFilters.java`	Using a Total Purchase Price Ceiling filter
Fraud Review	`DOFraudReview.java`	Approving a fraudulent transaction

TABLE 4.1 Samples listed by transaction type *(Continued)*

Transaction Type	Source Code File Name	When to Use
Recurring Billing **paypal.samples.dataobjects.recurring** **NOTE:** You must sign up for the Recurring Billing Service to be able to use the Recurring objects in the Payflow SDK.		
Recurring Add	`DORecurringAdd.java`	Adding a new Recurring Billing profile
Recurring Cancel	`DORecurringCancel.java`	Deactivating/canceling a profile
Recurring Inquiry	`DORecurringInquiry.java`	Viewing the status of a profile
Recurring Modify	`DORecurringModify.java`	Modifying a Recurring Billing profile
Recurring Payment	`DORecurringPayment.java`	Performing a real-time retry on a Recurring transaction that failed
Recurring ReActivate	`DORecurringReActivate.java`	Reactivating a profile with inactive status
General **paypal.samples.dataobjects.misc**		
Reference	`DOReference.java`	Using the base Reference transaction to perform any type of Reference transaction
ACH	`DOSale_ACH.java`	Doing a simple Sale – ACH transaction
Telecheck	`DOSale_Telecheck.java`	Doing a simple Sale Telecheck transaction
Sale with Verbosity	`DOVerbosity.java`	Doing a simple Sale transaction with Verbosity set to “HIGH”
Sale with additional headers	`DOAdditionalHeaders.java`	Passing additional headers currently not supported by the SDK to the Payflow server
Name-Value Pairs (NVP) **paypal.samples.namevaluepairs**		
NVP from Command Prompt	`NVPCommandLine.java`	Running a transaction from a batch file or command line by passing in the required parameters The parameters and their order are: `<Host Address> <Host Port> <NVP Parameter String> <Time Out> <Proxy Address> <Proxy Port> <Proxy Logon> <Proxy Password>`
NVP Sale	`NVPSale.java`	Running a Sale transaction using the name-value pair parameter list string
XMLPay **paypal.samples.xmlpay**		

TABLE 4.1 Samples listed by transaction type *(Continued)*

Transaction Type	Source Code File Name	When to Use
XML Pay from Command Prompt	`XMLPayCommandLine.java`	Running a transaction from a batch file or command line by passing in the required parameters The parameters and their order are: `<Host Address> <Host Port> <XML Parameter String> <Time Out> <Proxy Address> <Proxy Port> <Proxy Logon> <Proxy Password>`
XMLPay Sale	`XMLPaySale.java`	Using XMLPay 2.0 to do a Sale transaction

Object-Based Samples and Their Objects

The object-based samples use data objects and transaction objects to do the transactions in a strongly typed fashion.

NOTE: All samples are for simple transactions only. For complex transactions, you may have to send additional parameters, and you may receive additional parameters in the response.

DOAdditionalHeaders.java

Typically, the following data objects need to be created:

- `UserInfo`
- `PayflowConnectionData`
- `Invoice`
- `BillTo`
- `CreditCard`
- `CardTender`
- `SaleTransaction`
- `ClientInfo`

Use `AddTransHeader` method of the transaction object to add additional headers to the transaction request.

Use `AddCommitHeader` method of the transaction object to add additional headers to the commit request.

DOAuth.java

Typically, the following data objects need to be created to use the `AuthorizationTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `Invoice`
- `BillTo`
- `CreditCard`
- `CardTender`
- `ClientInfo`

`TransactionResponse` and `FraudResponse` contain the response parameters.

DOCapture.java

Typically, the following data objects need to be created to use the `CaptureTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `ClientInfo`

Send the `ORIGID` for the reference transaction in the transaction.

`TransactionResponse` contains the response parameters.

DOCredit.java

Typically, the following data objects need to be created for an independent credit to use the `CreditTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `Invoice`
- `CreditCard`
- `CardTender`
- `ClientInfo`

`TransactionResponse` contains the response parameters.

DOFraudFilters.java

Typically, the following data objects need to be created to use the `SaleTransaction` transaction object for doing the transaction.

This sample uses `SaleTransaction`, but any other transaction object can also be used.

- `UserInfo`
- `PayflowConnectionData`
- `Invoice`
- `BillTo`
- `CreditCard`
- `CardTender`
- `ClientInfo`

Set the Fraud `Verbosity` to HIGH to get maximum information in the response.

`TransactionResponse` and `FraudResponse` contain the response parameters.

`FraudResponse` has Pre- and Post-XML Data objects as `FpsXmlData`.

`FpsXmlData` contains a list of `Rule` objects, which in turn has a list of `RuleParameter` objects. The XML response from the server for the `FPS_PREXMLDATA` and `FPS_POSTXMLDATA` parameters gets converted into this list of objects.

DOFraudReview.java

Typically, the following data objects need to be created to use the `FraudReviewTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `ClientInfo`

Send the `ORIGID` for the fraudulent transaction and the `UPDATEACTION` to be performed in the transaction.

`TransactionResponse` contains the response parameters.

DOInquiry.java

Typically, the following data objects need to be created to use the `InquiryTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `ClientInfo`

Send the `ORIGID` for the reference transaction in the transaction.

`TransactionResponse` and `FraudResponse` contain the response parameters.

DOReferenceCredit.java

Typically, the following data objects need to be created to use the `CreditTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `ClientInfo`

Send the `ORIGID` for the reference transaction in the transaction.

`TransactionResponse` contains the response parameters.

DOSale.java and DoSaleComplete.java

Typically, the following data objects need to be created to use the `SaleTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `Invoice`
- `BillTo`
- `CreditCard`
- `CardTender`
- `ClientInfo`

`TransactionResponse` and `FraudResponse` contain the response parameters.

DOSwipe.java

Typically, the following data objects need to be created to use the `SaleTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `Invoice`
- `BillTo`
- `Swipe`
- `CardTender`
- `ClientInfo`

`TransactionResponse` and `FraudResponse` contain the response parameters.

DOVoiceAuth.java

Typically, the following data objects need to be created to use the `VoiceAuthTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `Invoice`
- `CreditCard`
- `CardTender`
- `ClientInfo`

Send the `AUTHCODE` in the transaction.

`TransactionResponse` contains the response parameters.

DOVoid.java

Typically, the following data objects need to be created to use the `VoidTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `ClientInfo`

Send the `ORIGID` for the reference transaction in the transaction.

`TransactionResponse` contains the response parameters.

DORecurringAdd.java

Typically, the following data objects need to be created to use the `RecurringAddTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `Invoice`
- `BillTo`
- `CreditCard`
- `CardTender`
- `RecurringInfo`
- `ClientInfo`

`TransactionResponse` and `RecurringResponse` contain the response parameters.

DORecurringCancel.java

Typically, the following data objects need to be created to use the `RecurringCancelTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `RecurringInfo` with `ORIGPROFILEID`
- `ClientInfo`

`TransactionResponse` and `RecurringResponse` contain the response parameters.

DORecurringInquiry.java

Typically, the following data objects need to be created to use the `RecurringInquiryTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `RecurringInfo` with `ORIGPROFILEID`
- `ClientInfo`

`TransactionResponse` and `RecurringResponse` contain the response parameters.

DORecurringModify.java

Typically, the following data objects need to be created to use the `RecurringModifyTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `RecurringInfo` with `ORIGPROFILEID`
- `ClientInfo`

`TransactionResponse` and `RecurringResponse` contain the response parameters.

DORecurringPayment.java

Typically, the following data objects need to be created to use the `RecurringPaymentTransaction` object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `RecurringInfo`
- `Invoice`

- BillTo
- ClientInfo

TransactionResponse and RecurringResponse contain the response parameters.

DORecurringReActivate.java

Typically, the following data objects need to be created to use the RecurringReActivateTransaction object for doing the transaction.

- UserInfo
- PayflowConnectionData
- RecurringInfo with ORIGPROFILEID
- ClientInfo

TransactionResponse and RecurringResponse contain the response parameters.

DOReference.java

Typically, the following data objects need to be created to use the ReferenceTransaction object for doing the transaction.

- UserInfo
- PayflowConnectionData
- ClientInfo

Send the ORIGID for the reference transaction and the TRXTYPE in the transaction.

TransactionResponse contains the response parameters.

DOSale_ACH.java

Typically, the following data objects need to be created to use the SaleTransaction object for doing the transaction.

- UserInfo
- PayflowConnectionData
- Invoice
- BillTo
- BankAcct
- ACHTender
- ClientInfo

Send the ORIGID for the reference transaction and the TRXTYPE in the transaction.

TransactionResponse and FraudResponse contain the response parameters.

DOSale_Telecheck.java

Typically, the following data objects need to be created to use the SaleTransaction object for doing the transaction.

- `UserInfo`
- `PayflowConnectionData`
- `Invoice`
- `BillTo`
- `CheckPayment`
- `CheckTender`
- `ClientInfo`

Send the `ORIGID` for the reference transaction and the `TRXTYPE` in the transaction.

`TransactionResponse` and `FraudResponse` contain the response parameters.

DOVerbosity.java

Typically, the following data objects need to be created to use the `RecurringTransaction` object for doing the transaction.

- `UserInfo`
- `Invoice`
- `BillTo`
- `CreditCard`
- `CardTender`
- `ClientInfo`

Set the Fraud `Verbosity` to HIGH to get maximum information in the response.

`TransactionResponse` and `FraudResponse` contain the response parameters.

4 Transaction Samples

A Transaction Parameter Mapping

This appendix maps transaction parameters to Java methods for getting and setting properties. You can find information about transaction parameters in the *Developer's Guide*.

The parameters are presented alphabetically by parameter name.

TABLE A.1 Parameter Mapping

Parameter	Object	Method
ABA	BankAcct	getAba()
ACCT	BankAcct CreditCard PurchaseCard SwipeCard CheckPayment RecurringResponse	getAcct()
ACCTTYPE	BankAcct	getAcctType() setAcctType(String acctType)
ACSURL	BuyerAuthResponse	getAcsUrl() setAcsUrl(String acsUrl)
ACTION	Value is set based on Transaction object used: RecurringAddTransaction RecurringModifyTransaction RecurringCancelTransaction RecurringInquiryTransaction RecurringReActivateTransaction RecurringPaymentTransaction	NA
ADDLMSGS	TransactionResponse	getAddlMsgs()
AGGREGATEAMT	RecurringResponse	getAggregateAmt()
AGGREGATEOPTIONALAMT	RecurringResponse	getAggregateOptionalAmt()
ALTTAXAMT	Invoice	getAltTaxAmt() setAltTaxAmt(Currency altTaxAmt)
AMT	Invoice RecurringResponse	getAmt() setAmt(Currency amt)
AMEXID	TransactionResponse	getAmexId()

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Method
AUTHCODE	VoiceAuthTransaction TransactionResponse	getAuthCode()
AUTHENTICATION_ID	BuyerAuthResponse BuyerAuthStatus	getAuthenticationId() setAuthenticationId(String authenticationId)
AUTHENTICATION_STATUS	BuyerAuthResponse BuyerAuthStatus	getAuthenticationStatus() setAuthenticationStatus(String authenticationStatus)
AUTHTYPE	ACHTender	getAuthType() setAuthType(String authType)
AVSADDR	TransactionResponse	getAvsAddr()
AVSZIP	TransactionResponse	getAvsZip()
BALAMT	TransactionResponse	getBalAmt()
BATCHID	TransactionResponse	getBatchId()
BILLTOCOUNTRY	BillTo	getBillToCountry() setBillToCountry(String country)
BILLTOPHONE2	BillTo	setBillToPhone2() setBillToPhone2(String billToPhone2)
BILLTOSTREET2	BillTo	getBillToStreet2() setBillToStreet2(String billToStreet2)
BROWSERCOUNTRYCODE	BrowserInfo	getBrowserCountryCode() setBrowserCountryCode(String browserCountryCode)
BROWSERTIME	BrowserInfo	getBrowserTime() setBrowserTime(String browserTime)
BROWSERUSERAGENT	BrowserInfo	getBrowserUserAgent() setBrowserUserAgent(String browserUserAgent)
CAPTURECOMPLETE	CaptureTransaction	getCaptureComplete() setCaptureComplete(String captureComplete)
CARDSECURE	TransactionResponse	getCardSecure()
CAVV	BuyerAuthResponse BuyerAuthStatus	getCavv() setCavv(String cavv)
CHKNUM	ACHTender CheckTender CardTender	setChkNum(String chkNum) getChkNum()

***TABLE A.1 Parameter Mapping** (Continued)*

Parameter	Object	Method
CHKTYPE	ACHTender CheckTender CardTender	setChkType(String chkNum) getChkType()
CITY	BillTo RecurringResponse	getCity() setCity(String city)
COMMCARD	PurchaseCard	getCommCard()
COMMCODE	Invoice	getCommCode() setCommCode(String commCode)
COMMENT1	Invoice	getComment1() setComment1(String comment1)
COMMENT2	Invoice	getComment2() setComment2(String comment2)
COMPANYNAME	BillTo RecurringResponse	getCompanyName() setCompanyName(String companyName)
COUNTRY	RecurringResponse	getCountry()
COUNTRYCODE	ExpressCheckoutRequest	getCountryCode() setCountryCode(String countryCode)
CUSTCODE	CustomerInfo	getCustCode() setCustCode(String custCode)
CUSTID	CustomerInfo	getCustId() setCustId(String custId)
CUSTIP	CustomerInfo	getCustIp() setCustIp(String custIp)
CUSTREF	Invoice TransactionResponse	getCustRef()
CUSTVATREGNUM	CustomerInfo	getCustVatRegNum() setCustVatRegNum(String custVatRegNum)
CVV2	PurchaseCard CreditCard	getCvv2() setCvv2(String cvv2)
CVV2MATCH	TransactionResponse	getCvv2Match()
DATETOSETTLE	TransactionResponse	getDateToSettle()
DESC	Invoice	getDesc() setDesc(String desc)
DESC1	Invoice	getDesc1() setDesc1(String desc1)

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Method
DESC2	Invoice	getDesc2() setDesc2(String desc2)
DESC3	Invoice	getDesc3() setDesc3(String desc3)
DESC4	Invoice	getDesc4() setDesc4(String desc4)
DISCOUNT	Invoice	getDiscount() setDiscount(Currency discount)
DL	CheckTender	getDL() setDL(String dL)
DOB	CustomerInfo	getDob() setDob(String dob)
DUPLICATE	TransactionResponse	getDuplicate()
DUTYAMT	Invoice	getDutyAmt() setDutyAmt(Currency dutyAmt)
ECI	BuyerAuthResponse BuyerAuthStatus	getEci() setEci(String eci)
EMAIL	BillTo RecurringResponse	getEmail() setEmail(String email)
ENDTIME	Invoice TransactionResponse	getEndTime()
EXPDATE	CreditCard PurchaseCard RecurringResponse	getExpDate()
FAX	BillTo	getFax() setFax(String fax)
FIRSTNAME	BillTo RecurringResponse	getFirstName() setFirstName(String firstName)
FPS_POSTXMLDATA	FraudResponse	getFpsPostXmlData()
FPS_PREXMLDATA	FraudResponse	getFpsPreXmlData()
FREIGHTAMT	Invoice	getFreightAmt() setFreightAmt(Currency freightAmt)
HANDLINGAMT	Invoice	getHandlingAmt() setHandlingAmt(Currency handlingAmt)
HOMEPHONE	BillTo	getHomePhone() setHomePhone(String homePhone)

***TABLE A.1 Parameter Mapping** (Continued)*

Parameter	Object	Method
HOSTADDRESS	PayflowConnectionData	getHostAddress()
HOSTCODE	TransactionResponse	getHostCode()
HOSTPORT	PayflowConnectionData	getHostPort()
IAVS	TransactionResponse	getIavs()
INVNUM	Invoice	getInvNum() setInvNum(String invNum)
INVOICEDATE	Invoice	getInvoiceDate() setInvoiceDate(String invoiceDate)
L_AMTn	LineItem	getAmt() setAmt(Currency amt)
L_CATALOGNUMn	LineItem	getCatalogNum() setCatalogNum(String catalogNum)
L_COMMCODEn	LineItem	getCommCode() setCommCode(String commCode)
L_COSTCENTERNUMn	LineItem	getCostCenterNum() setCostCenterNum(String costCenterNum)
L_COSTn	LineItem	getCost() SetCost(Currency cost)
L_DESCn	LineItem	getDesc() setDesc(String desc)
L_DISCOUNTn	LineItem	getDiscount() SetDiscount(Currency discount)
L_FREIGHTAMTn	LineItem	getFreightAmt() setFreightAmt(Currency freightAmt)
L_HANDLINGAMTn	LineItem	getHandlingAmt() setHandlingAmt(Currency handlingAmt)
L_MANUFACTURERn	LineItem	getManufacturer() setManufacturer(String manufacturer)
L_PICKUPCITYn	LineItem	getPickupCity() setPickupCity(String pickupCity)
L_PICKUPCOUNTRYn	LineItem	getPickupCountry() setPickupCountry(String pickupCountry)
L_PICKUPSTATEn	LineItem	getPickupState() setPickupState(String pickupState)
L_PICKUPSTREETn	LineItem	getPickupStreet() setPickupStreet(String pickupStreet)

***TABLE A.1 Parameter Mapping** (Continued)*

Parameter	Object	Method
L_PICKUPZIPn	LineItem	getPickupZip() setPickupZip(String pickupZip)
L_PRODCODEn	LineItem	getProdCode() setProdCode(String prodCode)
L_QTYn	LineItem	getQty() setQty(long qty)
L_SKUn	LineItem	getSku() setSku(String sku)
L_TAXAMTn	LineItem	getTaxAmt() setTaxAmt(Currency taxAmt)
L_TAXRATEn	LineItem	getTaxRate() setTaxRate(Currency taxRate)
L_TAXTYPEn	LineItem	getTaxType() setTaxType(String taxType)
L_TRACKINGNUMn	LineItem	getTrackingNum() setTrackingNum(String trackingNum)
L_TYPEn	LineItem	getType() setType(String type)
L_UNSPSCCODEn	LineItem	getUnspscCode() setUnspscCode(String unspscCode)
L_UOMn	LineItem	getUom() setUom(String uom)
L_UPCn	LineItem	getUpc() setUpc(String upc)
LASTNAME	BillTo RecurringResponse	getLastName() setLastName(String LastName)
LOCALTAXAMT	Invoice	getLocalTaxAmt() setLocalTaxAmt(Currency localTaxAmt)
MAXFAILPAYMENTS	RecurringResponse RecurringInfo	getMaxFailPayments()
MD	BuyerAuthResponse	getMd() setMd(String md)
MERCHDESCR	Invoice	getMerchDescr() setMerchDescr(String merchDescr)
MERCHSVC	Invoice	getMerchSvc() setMerchSvc(String merchSvc)
MICR	CheckPayment	NA

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Method
MIDDLENAME	BillTo RecurringResponse	getMiddleName() setMiddleName(String MiddleName)
NAME	BankAcct CreditCard PurchaseCard SwipeCard CheckPayment RecurringResponse	getName() setName(String name)
NATIONALTAXAMT	Invoice	getNationalTaxAmt() setNationalTaxAmt(Currency nationalTaxAmt)
NEXTPAYMENT	RecurringResponse	getNextPayment()
OPTIONALTRX	RecurringInfo	getOptionalTrx() setOptionalTrx(String optionalTrx)
OPTIONALTRXAMT	RecurringInfo	getOptionalTrxAmt() setOptionalTrxAmt(Currency optionalTrxAmt)
ORDERDATE	Invoice	getOrderDate() setOrderDate(String orderDate)
ORDERTIME	Invoice	getOrderTime() setOrderDate(String orderDate)
ORIGID	ReferenceTransaction CaptureTransaction FraudReviewTransaction InquiryTransaction VoidTransaction	getOrigId() setOrigId(String origId)
ORIGPROFILEID	RecurringInfo	getOrigProfileId() setOrigProfileId(String origProfileId)
ORIGRESULT	TransactionResponse	getOrigResult()
P_AMTn	RecurringResponse	getInquiryParams()
P_PNREFn	RecurringResponse	getInquiryParams()
P_RESULTn	RecurringResponse	getInquiryParams()
P_TENDERn	RecurringResponse	getInquiryParams()
P_TRANSTATEn	RecurringResponse	getInquiryParams()
P_TRANSTIMEn	RecurringResponse	getInquiryParams()
PAREQ	BuyerAuthResponse	getPaReq() setPaReq(String paReq)

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Method
PARTNER	UserInfo	NA
PAYMENTHISTORY	RecurringInfo	getPaymentHistory() setPaymentHistory(String paymentHistory)
PAYMENTNUM	RecurringInfo	getPaymentNum() setPaymentNum(String paymentNum)
PAYMENTSLEFT	RecurringResponse	getPaymentsLeft()
PAYPERIOD	RecurringResponse RecurringInfo	getPayPeriod()
PHONENUM	BillTo RecurringResponse	getPhoneNum() setPhoneNum(String phoneNum)
PNREF	TransactionResponse	getPnref()
PONUM	Invoice	getPoNum() setPoNum(String poNum)
POSTALCODE	ExpressCheckoutRequest	getPostalCode() setPostalCode(String postalCode)
POSTFPSMSG	FraudResponse	getPostFpsMsg()
PREFPSMSG	FraudResponse	getPreFpsMsg()
PRENOTE	ACHTender	getPreNote() setPreNote(String preNote)
PROCAVS	TransactionResponse	getProcAvs()
PROCCARDSECURE	TransactionResponse	getProcCardSecure()
PROCCVV2	TransactionResponse	getProcCVV2()
PROFILEID	RecurringResponse	getProfileId()
PROFILENAME	RecurringResponse RecurringInfo	getProfileName()
PROXYADDRESS	PayflowConnectionData	getProxyAddress()
PROXYLOGON	PayflowConnectionData	getProxyLogon()
PROXYPASSWORD	PayflowConnectionData	getProxyPassword()
PROXYPORT	PayflowConnectionData	getProxyPort()
PWD	UserInfo	NA
RECURRING	RecurringInfo Invoice	getRecurring() setRecurring(String recurring)
REQNAME	CustomerInfo	getReqName() setReqName(String reqName)

***TABLE A.1 Parameter Mapping** (Continued)*

Parameter	Object	Method
RESPMSG	TransactionResponse	getRespMsg()
RESPTEXT	TransactionResponse	getRespText()
RESULT	TransactionResponse	getResult()
RETRYNUMDAYS	RecurringResponse RecurringInfo	getRetryNumDays()
RPREF	RecurringResponse	getRpRef()
SHIPCARRIER	ShipTo	getShipCarrier() setShipCarrier(String shipCarrier)
SHIPFROMZIP	ShipTo	getShipFromZip() setShipFromZip(String shipFromZip)
SHIPMETHOD	ShipTo	getShipMethod() setShipMethod(String shipMethod)
SHIPPEDFROMZIP	ShipTo	getShipFromZip() setShipFromZip(String shipFromZip)
SHIPTOCITY	ShipTo RecurringResponse	getShipToCity() setShipToCity(String shipToCity)
SHIPTOCOUNTRY	ShipTo RecurringResponse	getShipToCountry() setShipToCountry(String shipToCountry)
SHIPTOEMAIL	ShipTo	getShipToEmail() setShipToEmail(String shipToEmail)
SHIPTOFIRSTNAME	ShipTo RecurringResponse	getShipToFirstName() setShipToFirstName(String shipToFirstName)
SHIPTOLASTNAME	ShipTo RecurringResponse	getShipToLastName() setShipToLastName(String shipToLastName)
SHIPTOMIDDLENAME	ShipTo RecurringResponse	getShipToMiddleName() setShipToMiddleName(String shipToMiddleName)
SHIPTOPHONE	ShipTo	getShipToPhone() setShipToPhone(String shipToPhone)
SHIPTOPHONE2	ShipTo	getShipToPhone2() setShipToPhone2(String shipToPhone2)

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Method
SHIPTOSTATE	ShipTo RecurringResponse	getShipToState() setShipToState(String shipToState)
SHIPTOSTREET	ShipTo RecurringResponse	getShipToStreet() setShipToStreet(String shipToStreet)
SHIPTOSTREET2	ShipTo	getShipToStreet2() setShipToStreet2(String shipToStreet2)
SHIPTOZIP	ShipTo RecurringResponse	getShipToZip() setShipToZip(String shipToZip)
SS	CheckTender	getSS() setSS(String sS)
START	RecurringResponse RecurringInfo	getStart()
STARTTIME	Invoice TransactionResponse	getStartTime()
STATE	BillTo RecurringResponse	getState() setState(String state)
STATUS	RecurringResponse	getStatus()
STREET	BillTo RecurringResponse	getStreet() setStreet(String street)
SWIPE	SwipeCard	getSwipe()
TAXAMT	Invoice	getTaxAmt() setTaxAmt(Currency taxAmt)
TAXEXEMPT	Invoice	getTaxExempt() setTaxExempt(String taxExempt)
TENDER	Value is set based on Data object used: ACHTender CardTender CheckTender	setTender(BaseTender tender) getTender()
TERM	RecurringResponse RecurringInfo	getTerm()
TERMCITY	ACHTender	getTermCity() setTermCity(String termCity)
TERMSTATE	ACHTender	getTermState() setTermState(String termState)
TIMEOUT	PayflowConnectionData	getTimeOut()

***TABLE A.1 Parameter Mapping** (Continued)*

Parameter	Object	Method
TOKEN	ExpressCheckoutRequest ExpressCheckoutResponse	getToken() (Get only for ExpressCheckoutResponse) setToken(String token)
TRANSSTATE	TransactionResponse	getTransState()
TRXPNREF	RecurringResponse	getTrxPNRef()
TRXRESPMSG	RecurringResponse	getTrxRespMsg()
TRXRESULT	RecurringResponse	getTrxResult()
TRXTYPE	Value is set based on the Transaction object used: SaleTransaction CreditTransaction VoidTransaction CaptureTransaction etc.	getTrxType()
USER	UserInfo	NA
VATREGNUM	Invoice	getVatRegNum() setVatRegNum(String vatRegNum)
VATTAXAMT	Invoice	getVatTaxAmt() setVatTaxAmt(Currency vatTaxAmt)
VATTAXPERCENT	Invoice	getVatTaxPercent() setVatTaxPercent(String vatTaxPercent)
VENDOR	UserInfo	NA
VERBOSITY	Value is set based on the Transaction object used: SaleTransaction CreditTransaction VoidTransaction CaptureTransaction AuthorizationTransaction VoiceAuthTransaction InquiryTransaction FraudReviewTransaction RecurringAddTransaction RecurringModifyTransaction RecurringCancelTransaction RecurringInquiryTransaction RecurringReActivateTransaction RecurringPaymentTransaction	getVerbosity() setVerbosity(String verbosity)

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Method
XID	BuyerAuthResponse BuyerAuthStatus	getXid() setXid(String xid)
ZIP	BillTo RecurringResponse	getZip() setZip(String zip)

B Header Parameter Mapping

Header parameters appear in the Payflow SDK log files. This appendix shows the correspondence between header parameters and Payflow SDK methods. The Payflow SDK sets the values of internal objects.

TABLE B.1 Header parameter mapping

Parameter	Object	Method
X-VPS-CLIENT-TIMEOUT	Input value (for TimeOut) in the constructor of PayflowconnectionData/Payflow API. If the value is not passed from the constructor, the default is 45 seconds.	SDKProperties.getTimeOut()
X-VPS-REQUEST-ID	Input value to transaction request. Also present in transaction response.	getRequestId() setRequestId()
X-VPS-VIT-CLIENT-VERSION	ClientInfo	getClientVersion()
X-VPS-VIT-CLIENT-TYPE	ClientInfo	getClientType()
X-VPS-VIT-OS-ARCHITECTURE	Internal	N/A
X-VPS-VIT-OS-NAME	Internal	N/A
X-VPS-VIT-OS-VERSION	Internal	N/A
X-VPS-VIT-PROXY	Internal	N/A
X-VPS-VIT-RUNTIME-VERSION	Internal	N/A
X-VPS-VIT-INTEGRATION-PRODUCT	ClientInfo	setIntegrationProduct()
X-VPS-VIT-INTEGRATION-VERSION	ClientInfo	setIntegrationVersion()

C Logging, Error Codes, and Exceptions

This appendix covers the following information:

Logging

Logging provides information for debugging and troubleshooting problems. The following events are logged:

- Entry/exit points of important methods
- Connection and debugging information
- Sent requests and received responses

Logging Priority Levels

The Payflow SDK logs messages using the priority levels described in Table C.1.

The priority levels for logging are listed in descending order. If you set logging to a high priority level, messages of lesser priority are also logged. For example, if you specify logging level as `DEBUG`, the Payflow SDK automatically logs `INFO`, `WARN`, `ERROR`, and `FATAL` messages. If you set the logging level to `WARN`, then the Payflow SDK automatically logs `ERROR` and `FATAL` but does not log `INFO` and `DEBUG` messages.

TABLE C.1 Priority Levels for Logging

Priority Level	Usage in the Payflow SDK
`PayflowConstants.SEVERITY_DEBUG`	Entry and exit messages from methods
`PayflowConstants.SEVERITY_INFO`	Parameter values
`PayflowConstants.SEVERITY_WARN`	Soft validation that allows transactions to go through
`PayflowConstants.SEVERITY_ERROR`	Validation errors
`PayflowConstants.SEVERITY_FATAL`	Exceptions in the Payflow SDK
`PayflowConstants.LOGGING_OFF`	No logging

Enabling Logging in the Payflow SDK for Java

By default, logging is off. To enable logging, add code to:

1. Set the log file name and path:

   `SDKProperties.setLogFileName(<pathToLogFile>);`

2. Set the logging level to one of the priority levels defined in Table C.1, "Priority Levels for Logging":

   `SDKProperties.setLoggingLevel(<loggingLevel>);`

3. Set the log file size:

   `SDKProperties.setMaxLogFileSize(<fileSize>);`

   For example, a *<fileSize>* of 10000 sets the log file size to 10 KB.

   NOTE: If the location of the log file is not set and logging is turned on, then log messages default to the log file `payflow_java.log` in the current working directory.

Error Codes

This section describes the error codes that are returned by the Payflow SDK. Errors are returned if logging is set to priority `PayflowConstants.SEVERITY_ERROR` or higher. For more information on setting logging priority, see "Logging" on page 55.

NOTE: The error codes in this section are just the codes returned by the Payflow SDK. For a complete list of the transaction error codes, see the *Developer's Guide* that you are using with this SDK guide. *Developer's Guide*s are described in "Where to Go For More Information" on page 8."

Error Message Format

Error messages are in the Format:

```
[Severity]-RESULT=<Error Code>&RESPMSG=<Error Message>
```

NOTE: XMLPay 2.0 error messages are returned in XMLPay format.

Example Message

The following is an example error message:

```
[FATAL]-RESULT=-1&RESPMSG=Failed to connect to host, Input Server Uri
=https://pilot-payflowpro.verisign.com
```

Result Values for Communications Errors

This table lists the RESULT values returned by the Payflow SDK.

TABLE C.2 Result values for communications errors

RESULT	Description
-1	Failed to connect to host, Input Server Uri = <PF Pro Host Name>, [Input Proxy info = <Proxy Information> Usually caused by firewall and/or proxy setup issues or rules.
- 6	Parameter list format error: <incorrect string> in <parameter name>, <value where the error occurred>
- 7	Parameter list format error: invalid [] name length clause, Parameter length in '<value where the error occurred>' does not match actual value length. Parameter Name = <parameter name>
-12	Timeout waiting for response Input timeout value in millisec: <Timeout value> Usually caused by firewall and/or proxy setup issues or rules.
- 23	Host address not specified
- 40	Unexpected Request ID found in request. Input Request Id = <value from merchant> Request id from param list = <value from parameter list>
- 41	Required Request ID not found in request
-100	Parameter List cannot be empty
-103	Context Initialization failed
-104	Unexpected transaction state
-105	Invalid name value pair request
-106	Invalid response format
-107	This XMLPay version is not supported
-108	The server certificate chain did not validate
-109	Unable to do logging
-111	The following error occured while initializing from message file: <Details of the error message>
-113	Unable to round and truncate the currency value simultaneously. You can set only one of the two properties Round OR Truncate.

Exception Trace Messages

When you enable exception trace messages, you get the entire stack trace of the exception that occurred while processing a transaction.

When a processing error occurs in the SDK, the SDK returns a negative result code and a response message describing the error. The following table shows where to retrieve the result code and response message.

TABLE C.3 Retrieve result code and response message

Method	Result code	Reponse message
Object-based API	`Response.getTransactionResponse( ).getResult()`	`Response.getTransactionResponse( ).getRespMsg`
Name-value pairs	`RESULT`	`RESPMSG`
XMLPay	`<result></result>`	`<respmsg></respmsg>`

By default, stack tracing is turned off.

You can turn on exception stack tracing to get more information about the error. The stack trace is returned in the Response message of the transaction response.

Use the following code to turn on stack tracing:

```
SDKProperties.setStackTraceOn(true);
```

Index

P

R

S

T

U

V

W

X