PayPal™

Payflow SDK for .NET 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: February 2008

Payflow SDK for .NET Developer's Guide

Document Number: 200032.en_US-200802

© 2008 PayPal, Inc. All rights reserved. PayPal is a registered trademark of PayPal, Inc. The PayPal logo is a trademark of PayPal, Inc. Other trademarks and brands are the property of their respective owners.
The information in this document belongs to PayPal, Inc. It may not be used, reproduced or disclosed without the written approval of PayPal, Inc.
PayPal Europe S.à.r.l. & Cie, S.C.A. is authorized and regulated by the Commission de Surveillance du Secteur Financier in Luxembourg as a bank.
PayPal RCS registration number: 118349

Notice of non-liability:
PayPal, Inc. is providing the information in this document to you "AS-IS" with all faults. PayPal, Inc. makes no warranties of any kind (whether express, implied or statutory) with respect to the information contained herein. PayPal, Inc. assumes no liability for damages (whether direct or indirect), caused by errors or omissions, or resulting from the use of this document or the information contained in this document or resulting from the application or use of the product or service described herein. PayPal, Inc. reserves the right to make changes to any information herein without further notice.

Contents

Preface

This Document

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

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

Intended Audience

This guide assumes that its readers:

- Are experienced web or application developers
- Use .NET 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"
- New COM sample

  For more information, see Chapter 2, "Getting Started With Transactions"
- New section describing how to run samples using Visual Studio 2005 and .NET 2.0

  For more information, see "Using Visual Studio 2005 and .NET Framework 2.0" on page 39
- Organizes transaction parameter/object mapping alphabetically by transaction parameter name

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

  See the `docs` directory of the Payflow SDK . It contains descriptions of the SDK classes.

- *Developer's Guide*

  This .NET 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 .NET," provides information on additional software requirements and describes the Payflow 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.

Chapter 5, "Registering and Running the Samples," provides instructions for registering and running the sample applications.

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* that you must use with this 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:

- Verify that you have .NET Framework 1.1 or later installed. The Payflow SDK for .NET is not supported on previous .NET Framework versions.
- Be sure that the settings for the keys and `PAYFLOW_HOST` are correct in the configuration file. For console applications, the configuration file is `App.config`; for web-based applications, the configuration file is `Web.config`. See "Setting Application Defaults" on page 18.
- If you are having problems with a specific type of transaction, refer to the Console 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 "Logging" on page 61. When reporting your problem, send your log files to us, along with your questions.

Revision History

Revision history for *Payflow SDK for .NET 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. Add COM sample. Update objects information. Add description of using Visual Studio 2005 and .NET 2.0.
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	Remove 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 .NET

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 for .NET, be sure that you:

- Verify that you have .NET Framework 1.1 or later installed. The Payflow SDK for .NET is not supported on previous .NET Framework versions.
- 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.

Payflow SDK for .NET Packages

The Payflow SDK for .NET 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

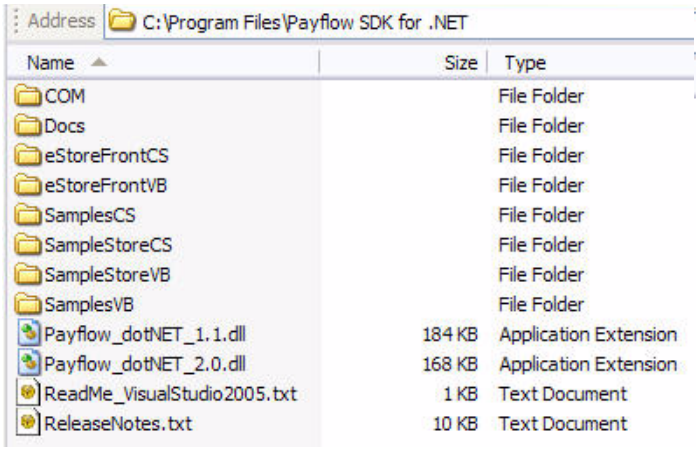


TABLE 1.1 Payflow SDK for .NET directory contents

Directory/File	Description
`COM`	Contains file and samples for using COM with .NET.
`docs`	Contains ReadMe files, SDK guide, and reference information.
`eStoreFrontCS`	Web-based sample written in C# and demonstrating Buyer Authentication and Express Checkout.
`eStoreFrontVB`	Web-based sample written in Visual Basic and demonstrating Buyer Authentication and Express Checkout.

TABLE 1.1 Payflow SDK for .NET directory contents

Directory/File	Description
`SamplesCS`	Contains samples written in C# that can be run from the console. For more information about individual samples, see Chapter 4, "Transaction Samples." **NOTE:** Use these samples as a starting point for learning more about the Payflow SDK.
`SampleStoreCS`	Web-based samples in C#. For information on how to register and run the Sample Store, see Chapter 5, "Registering and Running the Samples."
`SampleStoreVB`	Web-based samples in Visual Basic. For information on how to register and run the Sample Store, see Chapter 5, "Registering and Running the Samples."
`SamplesVB`	Contains samples written in VB.NET that can be run from the console. For more information about individual samples, see Chapter 4, "Transaction Samples." **NOTE:** Use these samples as a starting point for learning more about the Payflow SDK.
`Payflow_dotNET_1.1.dll`	The Payflow SDK for .NET API library for .NET Framework v1.1.
`Payflow_dotNET_2.0.dll`	The Payflow SDK for .NET API library for .NET Framework v2.0. See "Using Visual Studio 2005 and .NET Framework 2.0" on page 39 for more information.
`ReleaseNotes.txt`	Includes information specific to the release.
`ReadMe_VisualStudio 2005.txt`	Includes information specific to using this package with Visual Studio 2005. By default this package is designed for Visual Studio 2003 and uses .NET Framework v1.1.

Verifying the Installation

To verify the installation, run the `DoSaleComplete` (default application for the project) in the `samplesCS` or `SamplesVB` directory. This sample is highly commented and it is suggested that you review it.

NOTE: Prior to running any sample you will need to change the login credentials and replace `<VENDOR>`, `<USER>`, `<PARTNER>`, and `<PASSWORD>` with your credentials.

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:

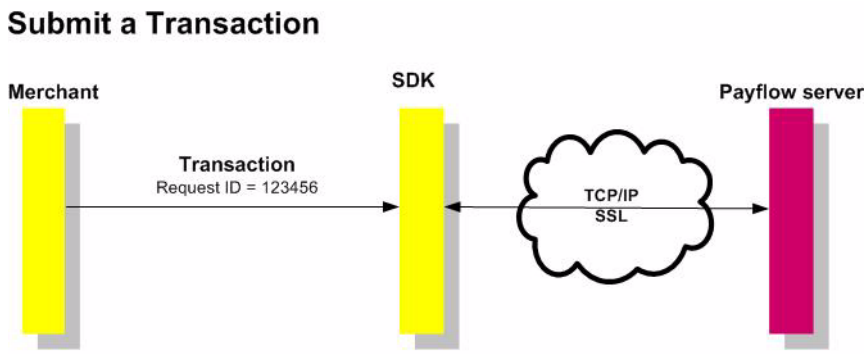
NOTE: Do not copy code samples from this guide. Code formatting may prevent the samples from compiling.

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 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 Payflow SDK constructs the request string from the data objects and sends it to the Payflow server. Because the Payflow 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.Street = "123 Main St.";
bill.Zip = "12345";
inv.BillTo = Bill;
...
SaleTransaction Trans = new SaleTransaction(User, Connection, Inv,
                                Card, PayflowUtility.RequestId);
```

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:

```
PayflowNETAPI PayflowNetApi = new PayflowNETAPI();
...
String Request = "...&STREET=123 Main St.&ZIP=12345...";
PayflowNetApi.SubmitTransaction(Request, PayflowUtility.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:

```
PayflowNETAPI PayflowNetApi = new PayflowNETAPI();
...
String Request = "<?xml version=\"1.0\"?> \
     ...
     <BillTo> \
        <PONum>12345</PONum> \
        <Address> \
           <Street>123 Main St.</Street> \
           <Zip>12345</Zip> \
        </Address> \
        ...
     </BillTo>";
PayflowNetApi.SubmitTransaction(Request, PayflowUtility.RequestId);
```

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

Setting Application Defaults

You can set application defaults in a configuration file for:

- The Payflow server (test or production)
- Logging and tracing

Console Applications

Add an Application Configuration File named `App.config` to your project. For a sample configuration file, see `App.config` in the `SamplesCS` or `SamplesVB` directory of the Payflow SDK.

Web-Based Applications

Add a Web Configuration File named `Web.config` to your project. For a sample configuration file, see `Web.config` in the `SampleStoreCS` or `SampleStoreVB` directory of the Payflow SDK.

Code Sample

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

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

C# Sample

The following are samples for each transaction mode written in C#:

- Object-based: See `DOSaleComplete.cs` in `SamplesCS/src/PayPal/Payments/Samples/CS/DataObjects/BasicTransactions` directory.
- Parameter-based
  - Name-value pairs: see `NVPSale.cs` in `SamplesCS/src/PayPal/Payments/Samples/CS/NameValuePairs`.
  - XMLPay: see `XMLPaySale.cs` in `SamplesCS/src/PayPal/Payments/Samples/CS/XMLPay`.

VB Sample

The following are samples for each transaction mode written in VB.NET:

- Object-based: See `DOSaleComplete.vb` in `SamplesVB/src/PayPal/Payments/Samples/VB/DataObjects/BasicTransactions` directory.
- Parameter-based
  - Name-value pairs: see `NVPSale.vb` in `SamplesVB/src/PayPal/Payments/Samples/VB/NameValuePairs`.
  - XMLPay: see `XMLPaySale.vb` in `SamplesVB/src/PayPal/Payments/Samples/VB/XMLPay`.

COM Sample

The following sample is written in ASP:

Name-value pairs: see `ASPComExample.asp` in the `COM` directory

Steps for a Simple Sale Transaction

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

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

1. Open `App.config` located in the `SamplesCS` directory. Check the value for `PAYFLOW_HOST`. Make sure that `PAYFLOW_HOST` is set to the test server as follows:

   ```
   <add key="PAYFLOW_HOST" value="pilot-payflowpro.paypal.com"/>
   ```

   NOTE: This assumes that the Payflow SDK for .NET is installed in the C:\Program Files\Payflow SDK for .NET directory.

2. Create the `PayflowConnectionData` object.

   This object refers to connection related information such as Payflow server host address, Payflow server port number, and so forth. Default value for `PAYFLOW_HOST` is specified in the `App.config` file as shown above. For more information about the `PayflowConnectionData` object, see "PayflowConnectionData Object" on page 21.

   For example, the following code creates the `PayflowConnectionData` object with default values.

   ```
   PayflowConnectionData Connection = new PayflowConnectionData();
   ```

3. 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.cs or DOSale.vb and DoSaleComplete.cs or DoSaleComplete.vb" on page 33.

4. 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.RequestId);
   ```

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

   ```
   Response Resp = Trans.SubmitTransaction();
   ```

   NOTE: The original request ID is set by the Payflow SDK in the `SaleTransaction` object.

6. 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 22.

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 `Payflow SDK for .NET\docs` directory.

This chapter consists of the following sections:

PayflowConnectionData Object

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

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

The constructor for the `PayflowNETAPI` object can override `hostAddress`, `timeout`, and other properties of the `PayflowConnectionData` object.

Connection Parameters

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

TABLE 3.1 Connection parameters

Parameter	Description
`HostAddress`	Test: `pilot-payflowpro.paypal.com` (Default) Live: `payflowpro.paypal.com`
`TimeOut`	45 (seconds) (Default)
`ProxyAddress`	`myproxy.mydomain.com` OR empty string/null (Optional)

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

Parameter	Description
ProxyPort	HTTP proxy port OR 0 (Optional)
ProxyLogin	proxy login name OR empty string/null (Optional)
ProxyPassword	proxy password OR empty string/null (Optional)

NOTE: If these properties are not set in the code, default values from the configuration file are used. For console applications, the configuration file is `App.config`; for web-based applications, the configuration file is `Web.config`.

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.Payments.Transactions namespace that are supported by the object-based transaction mode of the Payflow SDK. For example, the sale transaction type is in the following namespace:

```
PayPal.Payments.Transactions.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
Inquiry	RecurringInquiryTransaction
ReActivate	RecurringReActivateTransaction

TABLE 3.3 Transaction objects *(Continued)*

Transaction Type	Transaction Object
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.

Buyer Authentication requires Fraud Protection Services.

Data Objects

Table 3.4 lists and describes the important Payflow SDK data objects in the `PayPal.Payments.DataObjects` namespace. For example, the `CreditCard` data object is in the following namespace:

```
PayPal.Payments.DataObjects.CreditCard
```

NOTE: For complete information on data objects, see the class descriptions in `Payflow SDK for .NET\docs` 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
`CommitResponse`	Contains the parameters for the Commit response from the Payflow server
`CreditCard`	Contains the parameters for a credit card

TABLE 3.4 Important data objects *(Continued)*

Data object	Description
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. All source files are located in the SamplesCS or SamplesVB directory of the 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.cs for C# or DOSaleComplete.vb for Visual Basic.

All the parameter-based name-value pair samples start with the prefix "NVP" (for name-value pair), for example NVPSale.cs for C# or NVPSale.vb for Visual Basic.

All the parameter-based XMLPay-based samples start with the prefix "XMLPay," for example XMLPaySale.cs for C# or XMLPaySale.vb for Visual Basic.

Web Samples

The SampleStoreCS and SampleStoreVB directories contain samples demonstrating Buyer Authentication and Express Checkout. For more information, see "Registering the Sample Stores" on page 43, "Running the Sample Stores" on page 45, and ReadMeSampleStore.txt in the Payflow SDK for .NET\docs directory.

See the "Complete Listing of Console Samples" on page 28 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 SamplesCS or SamplesVB directory of the Payflow SDK. These samples may be used as a starting point for learning about the Payflow SDK.

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

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

```
PayPal.Payments.Samples.CS.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.Payments.Samples.CS.DataObjects.BasicTransactions**		
Authorize	`DOAuth.cs or DOAuth.vb`	Doing a simple authorization transaction
Delayed Capture	`DOCapture.cs or DOCapture.vb`	Doing a simple capture transaction
Credit	`DOCredit.cs or DOCredit.vb`	Doing a simple credit transaction
Inquiry	`DOInquiry.cs or DOInquiry.vb`	Doing a simple inquiry transaction
Reference Credit	`DOReferenceCredit.cs or DOReferenceCredit.vb`	Doing a reference credit transaction
Sale	`DOSaleComplete.cs or DOSaleComplete.vb`	Doing a sale transaction **NOTE:** This is a fully commented sale transaction, complete with business logic, that you should examine first
Sale	`DOSale.cs or DOSale.vb`	Doing a simple sale transaction
Swipe	`DOSwipe.cs or DOSwipe.vb`	Doing a simple sale transaction using SWIPE
Force/Voice Authorization	`DOVoiceAuth.cs or DOVoiceAuth.vb`	Doing a simple voice authorization transaction
Void	`DOVoid.cs or DOVoid.vb`	Doing a simple void transaction
Fraud Protection **PayPal.Payments.Samples.CS.DataObjects.Fraud**		
NOTE: You must sign up for the Fraud Protection Service to be able to use the Fraud objects in the Payflow SDK		

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

Transaction Type	Source Code File Name	When to Use
Fraud Filters	`DOFraudFilters.cs or DOFraudFilters.vb`	Using a Total Purchase Price Ceiling filter
Fraud Review	`DOFraudReview.cs or DOFraudReview.vb`	Approving a fraudulent transaction
Recurring Billing **PayPal.Payments.Samples.CS.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.cs or DORecurringAdd.vb`	Adding a new Recurring Billing profile
Recurring Cancel	`DORecurringCancel.cs or DORecurringCancel.vb`	Deactivating/canceling a profile
Recurring Inquiry	`DORecurringInquiry.cs or DORecurringInquiry.vb`	Viewing the status of a profile
Recurring Modify	`DORecurringModify.cs or DORecurringModify.vb`	Modifying a Recurring Billing profile
Recurring Payment	`DORecurringPayment.cs or DORecurringPayment.vb`	Performing a real-time retry on a Recurring transaction that failed
Recurring ReActivate	`DORecurringReActivate.cs or DORecurringReActivate.vb`	Reactivating a profile with inactive status
General **PayPal.Payments.Samples.CS.DataObjects.Misc**		
Reference	`DOReference.cs or DOReference.vb`	Using the base Reference transaction to perform any type of Reference transaction
ACH	`DOSale_ACH.cs or DOSale_ACH.vb`	Doing a simple Sale – ACH transaction
Telecheck	`DOSale_Telecheck.cs or DOSale_Telecheck.vb`	Doing a simple Sale Telecheck transaction
Sale with Verbosity	`DOVerbosity.cs or DOVerbosity.vb`	Doing a simple Sale transaction with Verbosity set to “HIGH”
Sale with additional headers	`DOAdditionalHeaders.cs or DOAdditionalHeaders.vb`	Passing additional headers currently not supported by the SDK to the Payflow server
Name-Value Pairs (NVP) **PayPal.Payments.Samples.CS.NameValuePairs**		

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

Transaction Type	Source Code File Name	When to Use
NVP from Command Prompt	`NVPCommandLine.cs or NVPCommandLine.vb`	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.cs or NVPSale.vb`	Running a Sale transaction using the name-value pair parameter list string
XMLPay **PayPal.Payments.Samples.CS.XMLPay**		
XML Pay from Command Prompt	`XMLPayCommandLine.cs or XMLPayCommandLine.vb`	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.cs or XMLPaySale.vb`	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.cs or DOAdditionalHeaders.vb

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.cs or DOAuth.vb

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.cs or DOCapture.vb

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.cs or DOCredit.vb

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.cs or DOFraudFilters.vb

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.cs or DOFraudReview.vb

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.cs or DOInquiry.vb

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.cs or DOReferenceCredit.vb

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.cs or DOSale.vb and DoSaleComplete.cs or DoSaleComplete.vb

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.cs or DOSwipe.vb

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.cs or DOVoiceAuth.vb

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.cs or DOVoid.vb

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.cs or DORecurringAdd.vb

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.cs or DORecurringCancel.vb

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.cs or DORecurringInquiry.vb

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.cs or DORecurringModify.vb

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.cs or DORecurringPayment.vb

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.cs or DORecurringReActivate.vb

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.cs or DOReference.vb

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.cs or DOSale_ACH.vb

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.cs or DOSale_Telecheck.vb

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.cs or DOVerbosity.vb

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.

5 Registering and Running the Samples

In this Chapter

This chapter explains how to run the console samples and how to register and run the Web-based sample stores with IIS 5.0 and IIS 6.0.

This chapter consists of the following sections:

NOTE: Use the console samples as a starting point for learning about the Payflow SDK.

The Sample Stores demonstrate functionality similar to the console samples, but the code is embedded in a Web application.

Using Visual Studio 2005 and .NET Framework 2.0

This chapter describes how to run the samples using Visual Studio 2003 and .NET Framework 1.1. By default, the samples use the 1.1 version of payflow_dotNET.dll.

To use the samples with Visual Studio 2005 and .NET Framework 2.0:

1. Rename `payflow_dotNET_2.0.dll` located in the root directory of the SDK to `payflow_dotNET.dll`.
2. Replace the current `payflow_dotNET.dll` in the project's BIN directory with the 2.0 version that you renamed.
3. Recompile the eStoreFront and Sample Store samples.

IMPORTANT: *You must rename the DLL file from payflow_dotNET_2.0.dll to payflow_dotNET.dll, replacing the original 1.1 version of the file.*

NOTE: `payflow_dotNET_2.0.dll` is the *only* DLL that is compiled for x64 systems.

Running Console Samples in Visual Studio .NET 2003

To use the samples with Visual Studio 2003 and .NET Framework 1.1:

1. Rename `payflow_dotNET_1.1.dll` located in the root directory of the SDK to `payflow_dotNET.dll`.
2. Replace the current `payflow_dotNET.dll` in the project's BIN directory with the 1.1 version that you renamed.
3. Recompile the eStoreFront and Sample Store samples.

IMPORTANT: *You must rename the DLL file from payflow_dotNET_1.1.dll to payflow_dotNET.dll.*

Running C# Console Samples

Step 1 Launching a project

Use the following procedure to launch Microsoft Visual Studio .NET 2003.

1. From the **Start** menu, select **Payflow SDK for .NET** → **Sample Console Applications**.
2. Select the **SamplesCS** folder.
3. In the **SamplesCS** window that appears, double-click the C# project file, `SamplesCS.csproj`.

   Microsoft Visual Studio .NET 2003 launches.

Step 2 Specifying the project startup object

This procedure assigns the class created as the startup object. A startup object is the class containing the Main method that is called at program startup.

1. In Microsoft Visual Studio .NET, click **View** → **Solution Explorer** to bring up the Solution Explorer window.
2. In **Solution Explorer**, right-click the project name, SamplesCS, and select **Properties**.

   The SamplesCS **Property Pages** dialog appears.
3. In the dialog under **Common Properties** → **General**, select **Startup Object**.
4. Type in the startup object name, for example:
   `PayPal.Payments.Samples.CS.DataObjects.BasicTransactions.DOSale`

5. Click **OK**.

Step 3 Running the project

1. To run your project, choose **Debug** → **Start**.

   The first time that you run the project, you are asked to save the solution (`.sln`) file. Click **Save**.

   A console window containing the results of running the application. If the application runs without error, a message indicates the success of the transaction. Otherwise, error messages appear.

2. Press **Enter** to exit the console window.

Running VB.NET Console Samples

Step 1 Launching a project in Microsoft Visual Studio .NET

Use the following procedure to launch Microsoft Visual Studio .NET 2003.

1. From the **Start** menu, select **Payflow SDK for .NET** → **Sample Console Applications**.
2. Select the **SamplesVB** folder.
3. In the **SamplesVB** window that appears, double-click the VB.NET project file, `SamplesVB.vbproj`.

   Microsoft Visual Basic .NET launches.

Step 2 Specifying the project startup object

This procedure assigns the class created as the startup object. A startup object is the class containing the Main method that is called at program startup.

1. Click **View** → **Solution Explorer** to bring up the Solution Explorer window.
2. In **Solution Explorer**, expand the SamplesVB project tree structure to view the list of sample source files.
3. Right-click SamplesVB, and select **Properties**.

   The SamplesVB **Property Pages** dialog appears.

4. In the dialog under **Common Properties** → **General**, select the sample that you want to run from the **Startup object** drop-down list.

   Samples are listed in the order that they appear in the Solution Explorer project tree structure. For example, the first sample listed is:
   `PayPal.Payments.Samples.VB.DataObjects.BasicTransactions.DOAuth`

5. Click **OK** to close the dialog.

Step 3 Running the project

1. To run your project, choose **Debug** →**Start**.

   The first time that you run the project, you are asked to save the solution (`.sln`) file. Click **Save**.

   A console window appears in which the results of running the application appear. If the application runs without error, a message indicates the success of the transaction. Otherwise, one or more error messages appear.

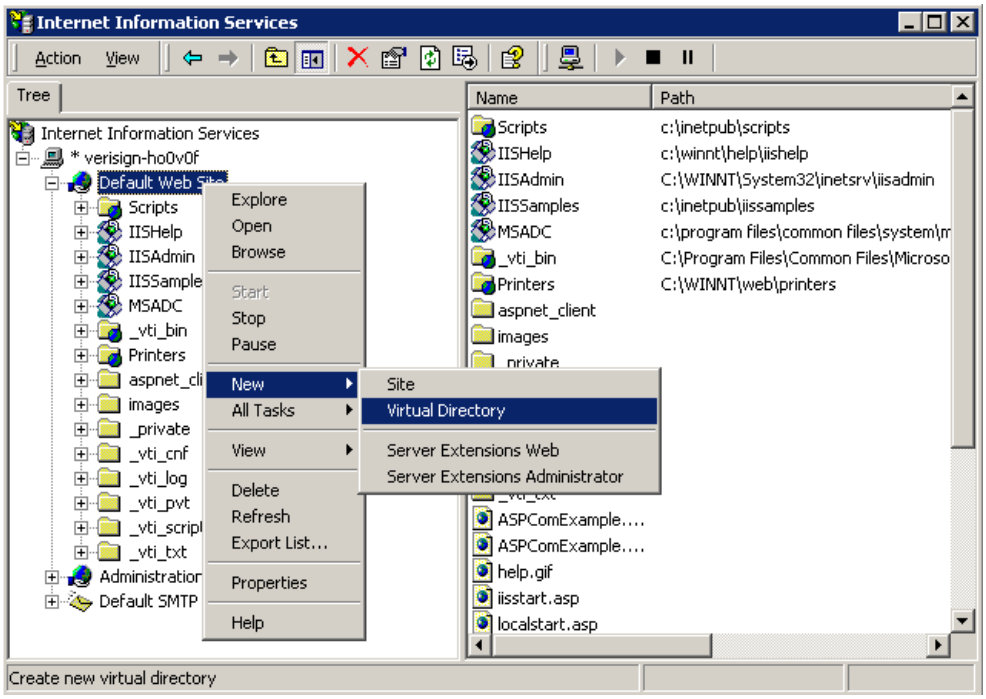
2. Press **Enter** to exit the console window.

Registering the Sample Stores

To register the Sample Stores into IIS 5.0 or 6.0, perform the following procedure:

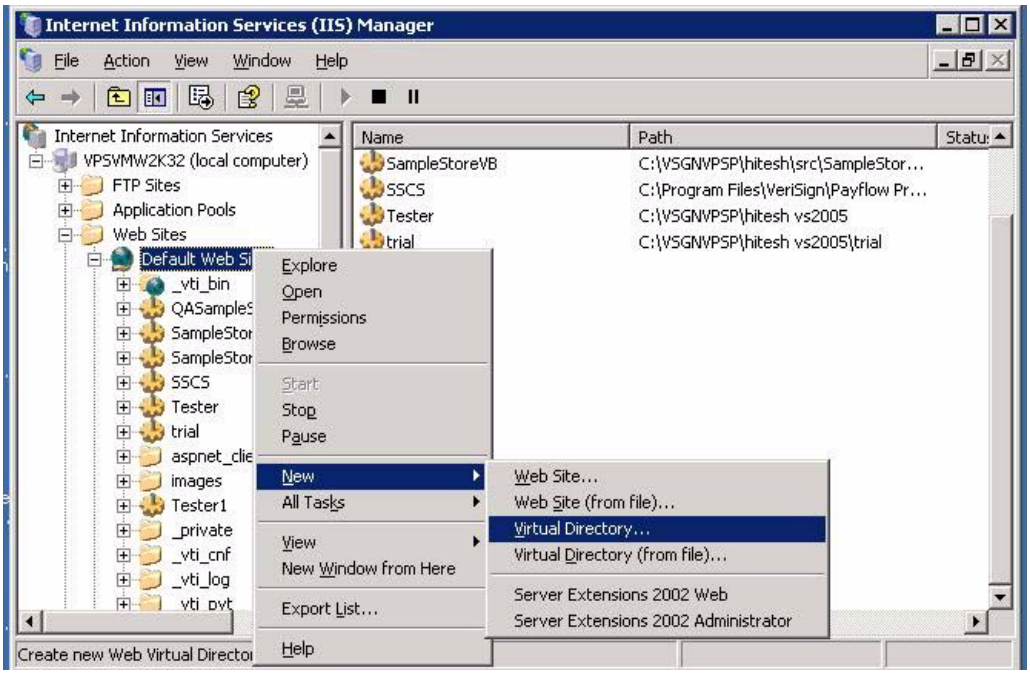
1. Go to **Start** → **Run**.
   a. Type in `inetmgr` and click **OK**. The **Internet Information Services** screen appears.
   b. For IIS 5.0, browse to your <hostname > → **Default Web Site**. See Figure 5.1.

FIGURE 5.1 Virtual directory IIS 5.0



   c. For IIS 6.0, browse to your <hostname > → **WebSites**→ **Default Web Site**. See Figure 5.2.

FIGURE 5.2 Virtual directory IIS 6.0



d. Right click **Default Web Site** → **New** and select **Virtual Directory**. The **Virtual Directory Creation Wizard** screen appears.

2. Click **Next**. The **Virtual Directory Alias** dialog appears.
3. In the text box for **Alias**, type in `SampleStoreCS` for the C# Sample Store. For the VB.NET Sample Store, type in `SampleStoreVB`.

   NOTE: You *must* use `SampleStoreCS` and `SampleStoreVB` for your aliases.

4. Click **Next**. The **Web Site Content Directory** dialog appears.
5. Click **Browse**. The **Browse for Folder** screen appears.
6. Browse to the directory `SampleStoreCS` in the SDK for the C# Sample Store. For the VB.NET Sample Store, browse to the directory `SampleStoreVB`.

   NOTE: By default the SDK is installed in `C:\Program Files\Payflow SDK for .NET`.

7. Click **OK** to return to the **Web Site Content Directory** dialog.
8. Click **Next**. The **Access Permissions** dialog appears.
9. Leave the default permissions as-is (Read and Run scripts), and click **Next**. The **Virtual Directory Creation Wizard – Completion** dialog appears.
10. Click **Finish** to complete the creation of the virtual directory.

11. Right click the **Virtual Directory** and select **Properties**.

12. In the **Property** window, select the **Documents** tab.

- In IIS 6.0, select **Enable default content page**.
- In IIS 5.0, select **Enable default document**.

13. Select each entry in the listbox, and click **Remove**.

14. Click **Add.**

- In IIS 6.0, this opens the **Add Content Page** dialog box
- In IIS 5.0, this opens the **Add Default Page** dialog box

15. Enter `default.aspx` in the dialog box and click **OK**.

Running the Sample Stores

NOTE: Do not use the Sample Stores in your application code, as they demonstrate all possible parameters that can be passed and returned. Instead, use code from the individual Console samples or eStoreFront samples.

Follow these instructions to run the C# Sample Store (the procedure is the same for the VB .NET Sample Store):

1. Open your Internet browser.

For the C# Sample Store, browse to the following URL:

`http://localhost/SampleStoreCS`

For the VB.NET Sample Store, browse to the following URL:

`http://localhost/SampleStoreVB`

2. To do a name-value pair transaction, perform the following procedure:

a. Choose **Name-Value Pair** from the **REQUEST TYPE** list.

FIGURE 5.3 Name-value pair request

b. Modify the name-value pairs to include your user, vendor, partner, and password information.

c. Click **Submit Transaction** at the bottom of the Sample Store request form.

The Results page appears, indicating that the transaction and Commit responses were successful.

3. To do an XML Pay 2.0 transaction, perform the following procedure:
   a. Choose **XML Pay 2.0** from the **REQUEST TYPE** list.
   b. Modify the XML to include your user, vendor, partner, and password information.
   c. Click **Submit Transaction** at the bottom of the Sample Store request form.
      The Results page appears, indicating that the transaction and Commit responses were successful.
4. To do an object-based transaction, perform the following procedure:
   a. Choose **Object Based** from the **REQUEST TYPE** list.
   b. Provide a value for each of the following fields in the Sample Store request form to do a simple sale transaction:
      – VENDOR
      – USER
      – PARTNER
      – PASSWORD
      – TENDER (Credit Card)
      – TRXTYPE (Sale)
      – AMT
      – ACCT
      – EXPDATE
   c. Click **Submit Transaction** at the bottom of the Sample Store request form.
      The Results page appears.

FIGURE 5.4 Sale transaction result

PayPal .NET SDK C# Sample Store

Status **Transaction Successful.**

Perform Follow-On Transaction | Back

Transaction Response Field	Transaction Response Value
ENDTIME	
REQUESTID	f4d5b25273f24f2790e441ddbb61b960
STARTTIME	
AVSZIP	Y
ADDLMSGS	
PROCCARDSECURE	
	TRXTYPE[1]=S&ACCT[16] =5100XXXXXXXX0008&EXPDATE[4]=0109&CVV2[3] =XXX&TENDER[1]=C&INVNUM[8]=INV12345&AMT[5]

Uninstalling the Sample Stores

To remove the Sample Store virtual directories from IIS, if created previously, perform the following procedure:

1. Go to **Start** → **Run**.
2. Type in `inetmgr` and click **OK**.

   The **Internet Information Services** screen appears.
3. For IIS 5.0, browse to your <hostname> → **Default Web Site**. For IIS 6.0, browse to your <hostname> → **WebSites** → **Default Web Site**.
4. Right-click SampleStoreCS/SampleStoreVB and, from the drop-down, choose **Delete**.
5. On the confirmation box, click **Yes**.

Running Sample Stores From Visual Studio .NET 2003

You can run the sample stores using either of two methods:

- By specifying the URL as previously described in "Running the Sample Stores" on page 45.
- By launching the Web page from Microsoft Visual Studio .NET 2003, as described in this section.

The steps to running a sample store are listed below. This procedure runs the C# Sample Store, but it is the same for both C# and VB.NET sample stores (with slight differences in file names).

Step 1 Launching a sample store project

Use the following procedure to launch Microsoft Visual Studio .NET 2003.

1. From the **Start** menu, select **Payflow SDK for .NET** → **Sample Web Applications**.
2. For the C# Sample Store, select the SampleStoreCS folder. (For the VB.NET Sample Store, select the SampleStoreVB folder.)
3. In the SampleStore CS window that appears, double-click the sample store project file, `SampleStoreCS.csproj`. (The project file for VB.NET is `SampleStoreVB.vbproj`.)

   Microsoft Visual Studio .NET 2003 launches.

Step 2 Specifying the sample store project startup object

This procedure assigns the class created as the startup object. A startup object is the class containing the Main method that is called at program startup.

1. Click **View** → **Solution Explorer** to bring up the Solution Explorer window.
2. In **Solution Explorer**, right-click the project name, SamplesStoreCS (or SamplesStoreVB for VB.NET), and select Set as Startup Project from the menu that appears.

Step 3 Running the sample store project

To run the sample store and transactions:

1. Choose **Debug** →**Start**.

   The first time that you run the project, you are asked to save the solution (`.sln`) file. Click **Save**.

   This launches the sample store in **Internet Explorer**.
2. To run a transaction, follow steps 2 through 4 in "Running the Sample Stores" on page 45.

A Transaction Parameter Mapping

This appendix maps transaction parameters to .NET objects. 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	Property
ABA	BankAcct	Aba (Get only)
ACCT	BankAcct CreditCard PurchaseCard SwipeCard CheckPayment RecurringResponse	Acct (Get only)
ACCTTYPE	BankAcct	AcctType
ACSURL	BuyerAuthResponse	AcsUrl
ACTION	Value is set based on Transaction object used: RecurringAddTransaction RecurringModifyTransaction RecurringCancelTransaction RecurringInquiryTransaction RecurringReActivateTransaction RecurringPaymentTransaction	NA
ADDLMSGS	TransactionResponse	AddlMsgs (Get only)
AGGREGATEAMT	RecurringResponse	AggregateAmt (Get only)
AGGREGATEOPTIONALAMT	RecurringResponse	AggregateOptionalAmt (Get only)
ALTTAXAMT	Invoice	AltTaxAmt
AMT	Invoice RecurringResponse	Amt
AMEXID	TransactionResponse	AmexId (Get only)
AUTHCODE	VoiceAuthTransaction TransactionResponse	AuthCode (Get only)

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Property
AUTHENTICATION_ID	BuyerAuthResponse BuyerAuthStatus	AuthenticationId
AUTHENTICATION_STATUS	BuyerAuthResponse BuyerAuthStatus	AuthenticationStatus
AUTHTYPE	ACHTender	AuthType
AVSADDR	TransactionResponse	AvsAddr (Get only)
AVSZIP	TransactionResponse	AvsZip (Get only)
BALAMT	TransactionResponse	BalAmt (Get only)
BATCHID	TransactionResponse	BatchId (Get only)
BILLTOCOUNTRY	BillTo	BillToCountry
BILLTOPHONE2	BillTo	BillToPhone2
BILLTOSTREET2	BillTo	BillToStreet2
BROWSERCOUNTRYCODE	BrowserInfo	BrowserCountryCode
BROWSERTIME	BrowserInfo	BrowserTime
BROWSERUSERAGENT	BrowserInfo	getBrowserUserAgent
CAPTURECOMPLETE	CaptureTransaction	CaptureComplete
CARDSECURE	TransactionResponse	CardSecure (Get only)
CAVV	BuyerAuthResponse BuyerAuthStatus	CAVV
CHKNUM	ACHTender CheckTender CardTender	ChkNum
CHKTYPE	ACHTender CheckTender CardTender	ChkType
CITY	BillTo RecurringResponse	City
COMMCARD	PurchaseCard	CommCard (Get only)
COMMCODE	Invoice	CommCode
COMMENT1	Invoice	Comment1
COMMENT2	Invoice	Comment2
COMPANYNAME	BillTo RecurringResponse	CompanyName
COUNTRY	RecurringResponse	Country (Get only)

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

Parameter	Object	Property
COUNTRYCODE	ExpressCheckoutRequest	CountryCode
CUSTCODE	CustomerInfo	CustCode
CUSTID	CustomerInfo	CustId
CUSTIP	CustomerInfo	CustIp
CUSTREF	Invoice TransactionResponse	CustRef (Get only)
CUSTVATREGNUM	CustomerInfo	CustVatRegNum
CVV2	PurchaseCard CreditCard	Cvv2
CVV2MATCH	TransactionResponse	Cvv2Match (Get only)
DATETOSETTLE	TransactionResponse	DateToSettle (Get only)
DESC	Invoice	Desc
DESC1	Invoice	Desc1
DESC2	Invoice	Desc2
DESC3	Invoice	Desc3
DESC4	Invoice	Desc4
DISCOUNT	Invoice	Discount
DL	CheckTender	DL
DOB	CustomerInfo	Dob
DUPLICATE	TransactionResponse	Duplicate (Get only)
DUTYAMT	Invoice	DutyAmt
ECI	BuyerAuthResponse BuyerAuthStatus	ECI
EMAIL	BillTo RecurringResponse	Email
ENDTIME	Invoice TransactionResponse	EndTime (Get only)
EXPDATE	CreditCard PurchaseCard RecurringResponse	ExpDate (Get only)
FAX	BillTo	Fax
FIRSTNAME	BillTo RecurringResponse	FirstName
FPS_POSTXMLDATA	FraudResponse	FpsPostXmlData (Get only)

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Property
FPS_PREXMLDATA	FraudResponse	FpsPreXmlData (Get only)
FREIGHTAMT	Invoice	FreightAmt
HANDLINGAMT	Invoice	HandlingAmt
HOMEPHONE	BillTo	HomePhone
HOSTADDRESS	PayflowConnectionData	HostAddress (Get only)
HOSTCODE	TransactionResponse	HostCode (Get only)
HOSTPORT	PayflowConnectionData	HostPort (Get only)
IAVS	TransactionResponse	Iavs (Get only)
INVNUM	Invoice	InvNum
INVOICEDATE	Invoice	InvoiceDate
L_AMTn	LineItem	Amt
L_CATALOGNUMn	LineItem	CatalogNum
L_COMMCODEn	LineItem	CommCode
L_COSTCENTERNUMn	LineItem	CostCenterNum
L_COSTn	LineItem	Cost
L_DESCn	LineItem	Desc
L_DISCOUNTn	LineItem	Discount
L_FREIGHTAMTn	LineItem	FreightAmt
L_HANDLINGAMTn	LineItem	HandlingAmt
L_MANUFACTURERn	LineItem	Manufacturer
L_PICKUPCITYn	LineItem	PickupCity
L_PICKUPCOUNTRYn	LineItem	PickupCountry
L_PICKUPSTATEn	LineItem	PickupState
L_PICKUPSTREETn	LineItem	PickupStreet
L_PICKUPZIPn	LineItem	PickupZip
L_PRODCODEn	LineItem	ProdCode
L_QTYn	LineItem	Qty
L_SKUn	LineItem	Sku
L_TAXAMTn	LineItem	TaxAmt
L_TAXRATEn	LineItem	TaxRate
L_TAXTYPEn	LineItem	TaxType
L_TRACKINGNUMn	LineItem	TrackingNum

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

Parameter	Object	Property
L_TYPEn	LineItem	Type
L_UNSPSCCODEn	LineItem	UnspscCode
L_UOMn	LineItem	Uom
L_UPCn	LineItem	Upc
LASTNAME	BillTo RecurringResponse	LastName
LOCALTAXAMT	Invoice	LocalTaxAmt
MAXFAILPAYMENTS	RecurringResponse RecurringInfo	MaxFailPayments (Get only)
MD	BuyerAuthResponse	MD
MERCHDESCR	Invoice	MerchDescr
MERCHSVC	Invoice	MerchSvc
MICR	CheckPayment	NA
MIDDLENAME	BillTo RecurringResponse	MiddleName
NAME	BankAcct CreditCard PurchaseCard SwipeCard CheckPayment RecurringResponse	Name
NATIONALTAXAMT	Invoice	NationalTaxAmt
NEXTPAYMENT	RecurringResponse	NextPayment (Get only)
OPTIONALTRX	RecurringInfo	OptionalTrx
OPTIONALTRXAMT	RecurringInfo	OptionalTrxAmt
ORDERDATE	Invoice	OrderDate
ORDERTIME	Invoice	OrderTime
ORIGID	ReferenceTransaction CaptureTransaction FraudReviewTransaction InquiryTransaction VoidTransaction	OrigId
ORIGPROFILEID	RecurringInfo	OrigProfileId
ORIGRESULT	TransactionResponse	OrigResult (Get only)
P_AMTn	RecurringResponse	InquiryParams (Get only)

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Property
P_PNREFn	RecurringResponse	InquiryParams (Get only)
P_RESULTn	RecurringResponse	InquiryParams (Get only)
P_TENDERn	RecurringResponse	InquiryParams (Get only)
P_TRANSTATEn	RecurringResponse	InquiryParams (Get only)
P_TRANSTIMEn	RecurringResponse	InquiryParams (Get only)
PAREQ	BuyerAuthResponse	PaReq
PARTNER	UserInfo	NA
PAYMENTHISTORY	RecurringInfo	PaymentHistory
PAYMENTNUM	RecurringInfo	PaymentNum
PAYMENTSLEFT	RecurringResponse	PaymentsLeft (Get only)
PAYPERIOD	RecurringResponse RecurringInfo	PayPeriod (Get only)
PHONENUM	BillTo RecurringResponse	PhoneNum
PNREF	TransactionResponse	Pnref (Get only)
PONUM	Invoice	PoNum
POSTALCODE	ExpressCheckoutRequest	PostalCode
POSTFPSMSG	FraudResponse	PostFpsMsg (Get only)
PREFPSMSG	FraudResponse	PreFpsMsg (Get only)
PRENOTE	ACHTender	PreNote
PROCAVS	TransactionResponse	ProcAvs (Get only)
PROCCARDSECURE	TransactionResponse	ProcCardSecure (Get only)
PROCCVV2	TransactionResponse	ProcCVV2 (Get only)
PROFILEID	RecurringResponse	ProfileId (Get only)
PROFILENAME	RecurringResponse RecurringInfo	ProfileName (Get only)
PROXYADDRESS	PayflowConnectionData	ProxyAddress (Get only)
PROXYLOGON	PayflowConnectionData	ProxyLogon (Get only)
PROXYPASSWORD	PayflowConnectionData	ProxyPassword (Get only)
PROXYPORT	PayflowConnectionData	ProxyPort (Get only)
PWD	UserInfo	NA
RECURRING	RecurringInfo Invoice	Recurring

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Property
REQNAME	CustomerInfo	ReqName
RESPMSG	TransactionResponse	RespMsg (Get only)
RESPTEXT	TransactionResponse	RespText (Get only)
RESULT	TransactionResponse	Result (Get only)
RETRYNUMDAYS	RecurringResponse RecurringInfo	RetryNumDays (Get only)
RPREF	RecurringResponse	RpRef (Get only)
SHIPCARRIER	ShipTo	ShipCarrier
SHIPFROMZIP	ShipTo	ShipFromZip
SHIPMETHOD	ShipTo	ShipMethod
SHIPPEDFROMZIP	ShipTo	ShipFromZip
SHIPTOCITY	ShipTo RecurringResponse	ShipToCity
SHIPTOCOUNTRY	ShipTo RecurringResponse	ShipToCountr
SHIPTOEMAIL	ShipTo	ShipToEmail
SHIPTOFIRSTNAME	ShipTo RecurringResponse	ShipToFirstNam
SHIPTOLASTNAME	ShipTo RecurringResponse	ShipToLastName
SHIPTOMIDDLENAME	ShipTo RecurringResponse	ShipToMiddleNam
SHIPTOPHONE	ShipTo	ShipToPhone
SHIPTOPHONE2	ShipTo	ShipToPhone2
SHIPTOSTATE	ShipTo RecurringResponse	ShipToState
SHIPTOSTREET	ShipTo RecurringResponse	ShipToStreet
SHIPTOSTREET2	ShipTo	ShipToStreet2
SHIPTOZIP	ShipTo RecurringResponse	ShipToZip
SS	CheckTender	SS
START	RecurringResponse RecurringInfo	Start (Get only)

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Property
STARTTIME	Invoice TransactionResponse	StartTime (Get only)
STATE	BillTo RecurringResponse	State
STATUS	RecurringResponse	Status (Get only)
STREET	BillTo RecurringResponse	Street
SWIPE	SwipeCard	Swipe (Get only)
TAXAMT	Invoice	TaxAmt
TAXEXEMPT	Invoice	TaxExempt
TENDER	Value is set based on Data object used: ACHTender CardTender CheckTender	Tender
TERM	RecurringResponse RecurringInfo	Term (Get only)
TERMCITY	ACHTender	TermCity
TERMSTATE	ACHTender	TermState
TIMEOUT	PayflowConnectionData	TimeOut (Get only)
TOKEN	ExpressCheckoutRequest ExpressCheckoutResponse	Token (Get only for ExpressCheckoutResponse)
TRANSSTATE	TransactionResponse	TransState (Get only)
TRXPNREF	RecurringResponse	TrxPNRef (Get only)
TRXRESPMSG	RecurringResponse	TrxRespMsg (Get only)
TRXRESULT	RecurringResponse	TrxResult (Get only)
TRXTYPE	Value is set based on the Transaction object used: SaleTransaction CreditTransaction VoidTransaction CaptureTransaction etc.	TrxType (Get only)
USER	UserInfo	NA
VATREGNUM	Invoice	VatRegNum
VATTAXAMT	Invoice	VatTaxAmt

TABLE A.1 Parameter Mapping *(Continued)*

Parameter	Object	Property
VATTAXPERCENT	Invoice	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	Verbosity
XID	BuyerAuthResponse BuyerAuthStatus	XID
ZIP	BillTo RecurringResponse	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 properties. The Payflow SDK sets the values of internal objects.

TABLE B.1 Header parameter mapping

Parameter	Object	Property
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.	TimeOut
X-VPS-REQUEST-ID	Input value to transaction request. Also present in transaction response.	RequestId
X-VPS-VIT-CLIENT-VERSION	ClientInfo	ClientVersion (Get only)
X-VPS-VIT-CLIENT-TYPE	ClientInfo	ClientType
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	IntegrationProduct (Set only)
X-VPS-VIT-INTEGRATION-VERSION	ClientInfo	IntegrationVersion (Set only)

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 by Modifying the Configuration File

By default, logging is off. To enable logging:

1. Open the configuration file.
   - For console applications, open `App.config`
   - For web-based applications, open `Web.config`
2. `Change the setting for LOG_LEVEL` from `OFF` to one of the following values: `DEBUG, INFO, WARN, ERROR, or FATAL:`

   ```
   <add key="LOG_LEVEL" value="DEBUG"/>
   ```
3. Set the log file name and path:

   ```
   <add key="LOG_FILE" value="Logs\MyLog.LOG"/>
   ```
4. Set the log file size:

   ```
   <add key="LOGFILE_SIZE" value="10000" />
   ```

   For example, a `LOGFILE_SIZE` 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_dotNET.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 61.

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.paypal.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 Payflow 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.

Enabling Tracing by Modifying the Configuration File

To turn on stack tracing in the configuration file:

1. Open the configuration file
   - For console applications, open `App.config`
   - For web-based applications, open `Web.config`
2. Change the setting for `TRACE` from `OFF` to `ON`:

```
<add key="TRACE" value="ON"/>
```

Index

R

S

T

U

V

W

X