First Data Global Gateway
Web Service API
Integration Guide

Version 5.0

November 1, 2011

Revision History

DOCUMENT: *First Data Global Gateway Web Service API*

The following table lists the additions, updates/modifications, and deletions made to this specification during the last one year. All the additions and updates/modifications for the most current version have been notated in yellow highlighted text.

Page	Description of Change	Version	Revision Date
35	In the chart it states a *Chargetotal* value is a required field under *RecurringPaymentInformation* when trying to modify an Active Periodic Billing. This creates an error after submitting the request. The *Chargetotal* value is not needed.	4.5	10.02.2011
16	TransactionOrign is Required, not optional. Included example of sample code	5.0	10.31.11

WEB SERVICE API INTEGRATION GUIDE

Contents

1 Introduction

The First Data Global Gateway Web Service API is an Application Programming Interface, which allows you to connect your application with the First Data Global Gateway. Using the Web Service API, you can seamlessly accept credit card and check payments in your application.

> **Note:** If you store or process cardholder data with your application, you must ensure that, your application meets the Payment Card Industry Data Security Standard (PCI DSS) requirements. Depending on transaction volume, you may be required to have your application audited by a Qualified Security Assessor.

The First Data Global Gateway Web Service API is a SOAP-based web service. Some of the advantages of offering integration using a web service include:

- **Platform Independence –** Any application that can send and receive SOAP messages can communicate with the Web Service API. Because the Web Service API is built using open standards, you can choose any technology that suits your needs (e.g. J2EE, .NET, PHP, ASP, etc.) for integrating with the First Data Global Gateway.

- **Ease of Integration –** The application builds a SOAP request message which encodes your transaction, sends it via HTTPS to the web service, and waits for a SOAP response message. The response contains your transaction's status. Since SOAP and HTTPS are designed to be lightweight protocols, building requests and parsing responses is a straightforward. Furthermore, rarely do you have to do this manually, since there are a number of libraries available in various technologies. Building a SOAP request and handling the response is reduced to a few lines of code.

  > **Note:** Integrators of the First Data Global Gateway's Web Service API should have a working knowledge of Web Service, SOAP and XML; in addition to a full understanding of how to integrate data into an application or web site. All supporting software needed to support Web Service API should be installed in a development environment prior to Web Service implementation. The Integrator must be proficient in their selected programming language.

- **Security –** All communication between your application and First Data Global Gateway Web Service API is SSL-encrypted. Your application has a client certificate, which identifies it uniquely with the web service. The Web Service API holds a server certificate, which your application checks to ensure that it is communicating with the Web Service API. The Web Service also requires HTTP basic authorization (user name and password) in order to communicate with the web service. These security mechanisms guarantee that the transaction data sent to First Data Global Gateway Web Service API stays private and is available only to your application.

This document will assist you in integrating your application with the Web Service API, and provide a brief summary of the Web Service API solution feature set.

1 Required Data

This section describes the data required for communicating securely with the Web Service API. The following checklist provides an overview enabling you to ensure that you have received the whole set when registering your application for the First Data Global Gateway:

- **Store ID –** Your store ID, assigned by First Data.

- **User ID and Password –** The user ID and password required for basic authorization with the Web Service API. The user ID is in the format WS<store_ID>._.1. For example, if your store ID is 111920, your user ID is WS111920._.1. This information is in the. WS<store_ID>._.1.auth.txt file.

- **Client Certificate p12 File –** The client certificate stored in a p12 file, named in the format WS<store_ID>._.1.p12. For example, if your store ID is 111920, your p12 file is named WS111920._.1.p12. This file is used for authenticating the client with the First Data Global Gateway. For connecting with Java, you need a ks file, for example WS111920._.1.ks.

- **Client Certificate Installation Password –** The password required for installing the p12 client certificate file. This information is in the WS<store_ID>._.1.p12.pw.txt file.

- **Client Certificate Private Key –** The private key of the client certificate stored in a key file, named in the format WS<store_ID>.key. Depending on your choice of tools, this may be required for authenticating with the Web Service API.

- **Client Certificate Private Key Password –** The password required for the private key, named in the format ckp_<creation_timestamp>. For example, this might be ckp_1193927132. Depending on your choice of tools, this may be required for authenticating with the Web Service API. This information is in the WS<store_ID>._.1.key.pw.txt file.

- **Client Certificate PEM File –** The client certificate stored in a pem file, named in the format WS<store_ID>._.1.pem. For example, if your store ID is 111920, your pem file is named WS111920._.1.pem. This file is used for authenticating the client with the First Data Global Gateway. Depending on your choice of tools, this may be required for authenticating with the Web Service API instead of the p12 file.

Note: These files are delivered in the .tar.gz format, which can be opened using recent versions of WinZip or most other archive applications.

2   How the Web Service API Works

The following section describes the process of performing a credit card transaction through the Web Service API.

In most cases, a customer starts the overall communication process by buying goods or services with her credit card in your online store. Your store sends a credit card transaction to the First Data Global Gateway using the Web Service API. Having received the transaction, the First Data Global Gateway forwards it to the credit card processor for authorization. Based on the result, your online store receives an approval or an error response from the Web Service API. This means that you only need to be able to communicate with the First Data Global Gateway Web Service API in order to accept payments.
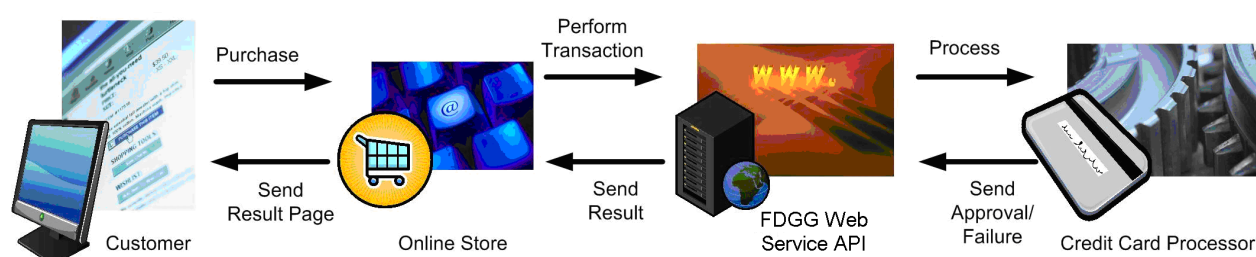
Web service interfaces are designed using the Web Service Definition Language (WSDL). The WSDL file for the Web Service API is located here:

https://ws.firstdataglobalgateway.com/fdggwsapi/services/order.wsdl

You must install the client certificate to access the WSDL file, for example, in a web browser. See 22 Installing the Client Certificate on page 97 for instructions on installing the client certificate.

After installing the client certificate, you can access the WSDL file. To access the WSDL file, follow these steps:

1. Open a Microsoft Internet Explorer window and enter the URL for the WSDL in the **Address** field.
2. After requesting the URL, the server will ask your browser to supply the client certificate ensure sure that it is talking to your application correctly. Since you have installed the certificate in the previous steps, you are seamlessly transferred to the server. Then, the First Data Global Gateway Web Service API sends its server certificate and the browser verifies that it comes from a trusted source. Again, this is done automatically without prompting you for any input. A secure connection is established and all data transferred between your application and the First Data API Web Service is SSL-encrypted.
3. The Web Service API WSDL file is displayed.

> **Note:** Your user ID and password are not required to view the WSDL but they are required to access the First Data Global Gateway Web Service API.

The WSDL file defines the operations offered by the Web Service API. It defines the request, response parameters, and call to operations. The First Data Global Gateway Web Service API WSDL file defines one operation, FDGGWSApiOrder, called by sending a SOAP request to the following URL:

https://ws.firstdataglobalgateway.com/fdggwsapi/services

This operation takes an XML-encoded transaction as a request and returns an XML-encoded response.

Depending on the tools you use to integrate with the Web Service API, you may need to provide the URL for the WSDL file. If so, you must tell your tool that the communication is SSL-enabled, provide your client certificate, and accept the server certificate as trusted. The process for this depends upon your tool. Consult the documentation for your tool for details.

The following chapters will guide you in setting up your store for building and performing custom credit card transactions.

3.1   Getting Started

After an account is loaded into our system, a merchant receives an automated Web Service API Welcome Email from 'secure@secure.linkpt.net'. It outlines the steps needed to complete the activation process for your account. This email contains the following information required to log in:

- DBA Name
- Store Name - a 6 or 10 digit numeric value
- User-ID
- Temporary Password - call the Support Desk at 888-477-3611 to receive your password. It is required to login to the site, process transaction reports, and perform administrative functions.
- Virtual Terminal URL - https://secure.linkpt.net/lpc/servlet/LPCLogin

3.2   LogIn

Enter your ten (10) digit Store Number or User ID

- Enter the Temporary Password
- Click **Login**
- Click **Accept** Agreement on the Global Gateway User Agreement

  Note: Temporary passwords are valid for only thirty (30) minutes after they are issued. We suggest you change your password regularly. Security specialists recommend merchants avoid using common words or numbers as passwords. Avoid words or numbers that might be associated with one's name or date of birth.

3.3   Password Guidelines

Password guidelines are as follows:

- Password length must be a minimum of eight (8), a maximum of fifteen (15) characters, and contain at least one (1) letter and one (1) number.
- Passwords expire every ninety (90) days
- Replaced passwords cannot be equivalent to any of the previous four (4) passwords for your account.
- After six (6) consecutive failed login attempts, a temporary (30) thirty minute account lockout will occur.

Once logged in, a merchant can begin processing transactions.

3 Supported Tools

The First Data Global Gateway Web Service API uses HTTPS and SOAP to communicate with your applications. As such, it is completely platform independent. The choice of languages, frameworks, or tools to integrate with the Web Service API is up to you.

First Data has tested the Web Service API with the following tools:

- PHP 5.2.9
- ASP
- .NET Framework
- Axis Framework 2-1.3
- Spring-WS 1.5.7

While you can use any tools to integrate with the API, these tools are First Data has tested. Integrating with the First Data Global Gateway Web Service API using other tools is outside the scope of this document.

4 Sending Transactions to the Gateway

This section describes the basic steps to take place when sending transactions to the First Data Global Gateway.

- The customer initiates checkout in the online store.
- The online store displays a form asking the customer to provide a credit card number and the expiration date.
- The customer enters and submits the data.
- The online store receives the data and builds an XML document encoding a Sale transaction, which includes the data provided by the customer and the total amount to be paid by the customer.
- After building the XML Sale transaction, the online store wraps it in a SOAP message, which describes the Web Service operation to be called with the transaction XML being passed as a parameter.
- The online store generates an HTTP POST request containing the soap message and sets the HTTP basic authorization headers.
- The online store establishes an SSL connection by providing the client and server certificate.
- The online store sends the HTTP POST request to the First Data Global Gateway Web Service API and waits for an HTTP response.
- The Web Service API receives the HTTPS request and parses the authorization information provided by the store in the HTTP headers.
- Having authorized the store, the Web Service API parses the SOAP message contained in the HTTP request body, triggering the call to the transaction operation.
- The First Data Global Gateway Web Service API performs the transaction processing, builds an XML response document, wraps it in a SOAP message, and sends the SOAP message back to the client in the body of an HTTP response.
- The online store receives the HTTP response.
- Depending upon the data contained in the XML response document, the online store displays the approval or error message.

While this example describes the case of a Sale transaction, other transactions follow the same process.

Your application performs the following steps in order to submit transactions and analyze the result:

- Build an XML document encoding your transactions.
- Wrap that XML document in a SOAP request message.
- Build an HTTP POST request with the information identifying your store provided in the HTTP header and the SOAP request message in the body.

- Establish an SSL connection between your application and First Data Global Gateway Web Service API.

- Send the HTTP POST request to the First Data Global Gateway Web Service API and receive the response.

- Read the SOAP response message out of the HTTPS response body.

- Analyze the XML response document contained in the SOAP response message.

The following chapters describe the information you need to perform these steps in detail and guide you through the process of setting up your application to perform transactions.

5 Building Transactions in XML

This chapter describes the XML formats for the submitting to the First Data Global Gateway Web Service API. After encoding the transaction in XML, the message is wrapped in SOAP envelope and submitted to the Web Service API.

While the tools you use to generate your request messages may allow you to avoid working with raw XML, you still need a basic understanding of the XML format in order to correctly build the XML transactions.

Credit card and check transactions are contained in the fdggwsapi:FDGGWSApiOrderRequest element.

Note: The First Data Global Gateway Web Service API only accepts ASCII characters. The Order ID field cannot contain the following characters: &, %, /, or exceed 100 characters in length.

6.1 Credit Card Transactions

Regardless of the transaction type, the basic XML document structure of a credit card transaction is as follows:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <v1:Transaction>
      <v1:CreditCardTxType>...</v1:CreditCardTxType>
      <v1:CreditCardData>...</v1:CreditCardData>
      <v1:Payment>...</v1:Payment>
      <v1:TransactionDetails>...</v1:TransactionDetails>
      <v1:Billing>...</v1:Billing>
      <v1:Shipping>...</v1:Shipping>
   </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

CreditCardDataTXType, CreditCardData, and Payment elements are mandatory for all credit card transactions. The other elements depend on the transaction type. The content depends on the type of transaction.

> See 7 XML Tag Reference on page 38 for details of all required and optional elements valid for submission for credit card transactions.

6.1.1 Sale

The following code is a sample of a Sale transaction using the minimum required number of elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
```

```
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <v1:Transaction>
      <v1:CreditCardTxType>
         <v1:Type>sale</v1:Type>
      </v1:CreditCardTxType>
      <v1:CreditCardData>
         <v1:CardNumber>4111111111111111</v1:CardNumber>
         <v1:ExpMonth>12</v1:ExpMonth>
         <v1:ExpYear>12</v1:ExpYear>
      </v1:CreditCardData>
      <v1:Payment>
         <v1:ChargeTotal>19.95</v1:ChargeTotal>
      </v1:Payment>
   </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required and optional fields for the Sale transaction. (v1:Billing, v1:Transaction details and v1:Shipping are optional)

Several situations that may occur which can cause a merchant's transactions to be downgraded.
1. **Failure to input the required data filed elements**
2. **Swiping a credit card in a designated CNP environment**
3. **Failure to input data into the Tax and PO# field**

The FDGG accepts $0.00 transactions for processing from **Visa**, **MasterCard*, JCB**, and **Discover** credit cards. The primary purpose for submitting a $0.00 "chargetotal" amount, with a billing address, is for credit card verification (not lost or stolen) or AVS (address match/mismatch). A $0.00 Authorization does not "hold" funds on the customer's account, cannot be submitted for settlement, nor have a Return transaction processed against it.

*MasterCard: Effective June 14, 2011 MasterCard will introduce a revised zero (0) dollar authorization functionality, officially named the Account Status Inquiry Service.
The revisions will *continue* to allow Merchants and Acquirers:
1. To validate the status of a cardholder's account by sending a transaction in the amount of USD $0
2. To submit an Address Verification (AVS), Card Verification Code 2 (CVC2) validations, or both in conjunction with the account status request
The new Account Status Inquiry Service will *discontinue*:
1. Accepting account status inquiries with a transaction amount greater than zero (0) dollars
2. Placing excessive or extended holds on customer's funds resulting from merchant account status inquiries

All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:CreditCardTxType/	
v1:Type	Required

FIELD	REQUIRED
v1:CreditCardData/	
v1:CardNumber	Required if v1:TrackData is not submitted.
v1:ExpMonth	Required if v1:TrackData is not submitted.
v1:ExpYear	Required if v1:TrackData is not submitted.
v1:CardCodeValue	Optional
v1:CardCodeIndicator	Optional
v1:TrackData	Required if v1:CardNumber is not submitted.
v1:CreditCard3DSecure/	
v1:PayerSecurityLevel	Required for 3D Secure transactions
v1:AuthenticationValue	See 0 8.3 CreditCard 3DSecure for details.
v1:XID	See 0 8.3 CreditCard 3DSecure for details.
v1:Payment/	
v1:ChargeTotal	Required – Can be ≥ $0.00
v1:SubTotal	Optional
v1:VATTax	Optional
v1:Shipping	Optional
v1:TransactionDetails/	
v1:UserID	Optional
v1:InvoiceNumber	Optional
v1:OrderId	Optional
v1:Ip	Optional – *Refers to cardholder's IP address for web transactions; not the IP address of the merchant's data center*.
v1:ReferenceNumber	Optional
v1:TDate	Optional
v1:Recurring	Optional
v1:TaxExempt	Optional
v1:TerminalType	Optional
v1:TransactionOrigin	Required –i.e. `<transactionorigin>RETAIL</transactionorigin>`
v1:PONumber	Optional
v1:DeviceID	Optional
v1:Billing/	*To prevent the possibility of downgrading, some Billing data is required for all MOTO & ECI transactions!*
v1:CustomerID	Optional
v1:Name	MOTO & ECI: **Required** Retail: **Optional**

FIELD	REQUIRED
v1:Company	Optional
v1:Address1	MOTO & ECI: **Required** Retail: **Optional**
v1:Address2	Optional
v1:City	MOTO & ECI: **Required** Retail: **Optional**
v1:State	MOTO & ECI: **Required** Retail: **Optional**
v1:Zip Code	MOTO & ECI: **Required** Retail: **Optional**
v1:Country	MOTO & ECI: **Required** Retail: **Optional**
v1:Phone	Optional
v1:Fax	Optional
v1:Email	Optional: *Required if receipts are to be emailed to customer and administrator*
v1:Shipping/	
v1:Type	Optional
v1:Name	Optional
v1:Address1	Optional
v1:Address2	Optional
v1:City	Optional
v1:State	Optional
v1:Zip	Optional
v1:Country	Optional

These elements must be submitted in the order defined in the XSD file: Transaction, CreditCardTxType, CreditCardData, and Payment.

6.1.2   PreAuth (Authorize Only in the Virtual Terminal)
A Pre-Auth transaction authorizes a dollar amount for a transaction, and places a hold on the customer's account equal to the transaction amount for a limited time. Depending on the cardholder's issuing bank, the reserve can be in place for several days.

The following code is a sample of a PreAuth transaction using the minimum required number of elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <v1:Transaction>
      <v1:CreditCardTxType>
         <v1:Type>preAuth</v1:Type>
```

```
        </v1:CreditCardTxType>
        <v1:CreditCardData>
            <v1:CardNumber>4111111111111111</v1:CardNumber>
            <v1:ExpMonth>12</v1:ExpMonth>
            <v1:ExpYear>12</v1:ExpYear>
        </v1:CreditCardData>
        <v1:Payment>
            <v1:ChargeTotal>100.00</v1:ChargeTotal>
        </v1:Payment>
    </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required and optional fields for the PreAuth transaction. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:CreditCardTxType/	
v1:Type	Required
v1:CreditCardData/	
v1:CardNumber	Required if v1:TrackData is not submitted.
v1:ExpMonth	Required if v1:TrackData is not submitted.
v1:ExpYear	Required if v1:TrackData is not submitted.
v1:CardCodeIndicator	Optional
v1:CardCodeValue	Optional
v1:TrackData	Required if v1:CardNumber is not submitted.
v1:CreditCard3DSecure/	
v1:PayerSecurityLevel	Required for 3D Secure transactions
v1:AuthenticationValue	See 0 8.3 CreditCard 3DSecure for details
v1:XID	See 0 8.3 CreditCard 3DSecure for details
v1:Payment/	
v1:ChargeTotal	Required – Can be > $0.00
v1:SubTotal	Optional
v1:VATTax	Optional
v1:Shipping	Optional
v1:TransactionDetails/	
v1:UserID	Optional
v1:InvoiceNumber	Optional
v1:OrderId	Optional
v1:Ip	Optional – *Refers to cardholder's IP address for web transactions; not the IP address of the merchant's data center.*

FIELD	REQUIRED
v1:ReferenceNumber	Optional
v1:TDate	Optional
v1:Recurring	Optional
v1:TaxExempt	Optional
v1:TerminalType	Optional
v1:TransactionOrigin	Optional, *default value if not provided.*
v1:PONumber	Optional
v1:DeviceID	Optional
v1:Billing/	*To prevent the possibility of downgrading, some Billing data is required for all MOTO & ECI transactions!*
v1:CustomerID	Optional
v1:Name	MOTO & ECI: **Required** Retail: **Optional**
v1:Company	Optional
v1:Address1	MOTO & ECI: **Required** Retail: **Optional**
v1:Address2	Optional
v1:City	MOTO & ECI: **Required** Retail: **Optional**
v1:State	MOTO & ECI: **Required** Retail: **Optional**
v1:Zip	MOTO & ECI: **Required** Retail: **Optional**
v1:Country	MOTO & ECI: **Required** Retail: **Optional**
v1:Phone	Optional
v1:Fax	Optional
v1:Email	Optional: *Required if receipts are to be emailed to customer and administrator*
v1:Shipping/	
v1:Type	Optional
v1: Name	Optional
v1:Address1	Optional
v1:Address2	Optional
v1:City	Optional
v1:State	Optional
v1:Zip	Optional
v1:Country	Optional

6.1.3   PostAuth (Ticket Only in the Virtual Terminal)

A PostAuth transaction finalizes its companion Pre-Auth transaction. It is done at the time of the product shipment or at the fulfilment of services. Once a Post-Auth transaction is submitted, the associated PreAuth transaction is eligible for settlement.

The amount of a PostAuth transaction must be:
1. Less than or equal to the amount of its associated PreAuth transaction
2. Greater than $0.00

If the PreAuth transaction total is greater than the companion PostAuth transaction total, then the difference is 'reserved' on the cardholder's account until the transaction is settled.

The following code is a sample of a PostAuth transaction using the minimum required elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <v1:Transaction>
      <v1:CreditCardTxType>
         <v1:Type>postAuth</v1:Type>
      </v1:CreditCardTxType>
      <v1:Payment>
         <v1:ChargeTotal>59.45</v1:ChargeTotal>
      </v1:Payment>
      <v1:TransactionDetails>
         <v1:OrderId>
            703d2723-99b6-4559-8c6d-797488e8977
         </v1:OrderId>
      </v1:TransactionDetails>
   </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required and optional fields for the PostAuth transaction. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:CreditCardTxType/	
v1:Type	Required
v1:Payment/	
v1:ChargeTotal	Optional – *Amount must be less than or equal to the PreAuth transaction it is associated with and greater than $0.00.*
v1:TransactionDetails/	
v1:OrderId	Required

6.1.4 ForceTicket

The following code is a sample of a ForceTicket transaction using the minimum required elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
    <v1:Transaction>
        <v1:CreditCardTxType>
            <v1:Type>forceTicket</v1:Type>
        </v1:CreditCardTxType>
        <v1:CreditCardData>
            <v1:CardNumber>4111111111111111</v1:CardNumber>
            <v1:ExpMonth>12</v1:ExpMonth>
            <v1:ExpYear>12</v1:ExpYear>
        </v1:CreditCardData>
        <v1:Payment>
            <v1:ChargeTotal>59.45</v1:ChargeTotal>
        </v1:Payment>
        <v1:TransactionDetails>
            <v1:ReferenceNumber>123456</v1:ReferenceNumber>
        </v1:TransactionDetails>
    </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required and optional fields for the ForceTicket transaction. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:CreditCardTxType/	
v1:Type	Required
v1:CreditCardData/	
v1:CardNumber	Required if v1:TrackData is not submitted.
v1:ExpMonth	Required if v1:TrackData is not submitted.
v1:ExpYear	Required if v1:TrackData is not submitted.
v1:CardCodeValue	Optional
v1:CardCodeIndicator	Optional
v1:TrackData	Required if v1:CardNumber is not submitted.
v1:CreditCard3DSecure/	
v1:PayerSecurityLevel	Required for 3D Secure transactions
v1:AuthenticationValue	See 0 8.3 CreditCard 3DSecure for details
v1:XID	See 0 8.3 CreditCard 3DSecure for details
v1:Payment/	

FIELD	REQUIRED
v1:ChargeTotal	Required – Must be > $0.00
v1:SubTotal	Optional
v1:VATTax	Optional
v1:Shipping	Optional
v1:TransactionDetails/	
v1:UserID	Optional
v1:InvoiceNumber	Optional
v1:OrderId	Optional
v1:Ip	Optional
v1:ReferenceNumber	Required
v1:TDate	Optional
v1:Billing/	*To prevent the possibility of downgrading, some Billing data is required for all MOTO & ECI transactions!*
v1:CustomerID	Optional
v1:Name	MOTO & ECI: **Required** Retail: **Optional**
v1:Company	Optional
v1:Address1	MOTO & ECI: **Required** Retail: **Optional**
v1:Address2	Optional
v1:City	MOTO & ECI: **Required** Retail: **Optional**
v1:State	MOTO & ECI: **Required** Retail: **Optional**
v1:Zip	MOTO & ECI: **Required** Retail: **Optional**
v1:Country	MOTO & ECI: **Required** Retail: **Optional**
v1:Fax	Optional
v1:Email	Optional: *Required if receipts are to be emailed to customer and administrator*
v1:Shipping/	
v1:Type	Optional
v1:Name	Optional
v1:Address1	Optional
v1:Address2	Optional
v1:City	Optional
v1:State	Optional
v1:Zip	Optional
v1:Country	Optional

FIELD	REQUIRED
v1:Phone	Optional
v1:Fax	Optional
v1:Email	Optional

6.1.5   Return

The following code is a sample of a Return transaction using the minimum required elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
    <v1:Transaction>
        <v1:CreditCardTxType>
            <v1:Type>return</v1:Type>
        </v1:CreditCardTxType>
        <v1:Payment>
            <v1:ChargeTotal>19.95</v1:ChargeTotal>
        </v1:Payment>
        <v1:TransactionDetails>
            <v1:OrderId>
                62e3b5df-2911-4e89-8356-1e49302b1807
            </v1:OrderId>
        </v1:TransactionDetails>
    </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required fields for the Return transaction. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:CreditCardTxType/	
v1:Type	Required
v1:Payment/	
v1:ChargeTotal	Required – Must be > $0.00
v1:TransactionDetails/	
v1:OrderId	Required

6.1.6   Credit

The following code is a sample of a Credit transaction using the minimum required elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
```

```
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <v1:Transaction>
      <v1:CreditCardTxType>
         <v1:Type>credit</v1:Type>
      </v1:CreditCardTxType>
      <v1:CreditCardData>
         <v1:CardNumber>4111111111111111</v1:CardNumber>
         <v1:ExpMonth>12</v1:ExpMonth>
         <v1:ExpYear>12</v1:ExpYear>
      </v1:CreditCardData>
      <v1:Payment>
         <v1:ChargeTotal>50.00</v1:ChargeTotal>
      </v1:Payment>
   </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required and optional fields for the Credit transaction. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:CreditCardTxType/	
v1:Type	Required
v1:CreditCardData/	
v1:CardNumber	Required if v1:TrackData is not submitted.
v1:ExpMonth	Required if v1:TrackData is not submitted.
v1:ExpYear	Required if v1:TrackData is not submitted.
v1:CardCodeValue	Optional
v1:CardCodeIndicator	Optional
v1:TrackData	Required if v1:CardNumber is not submitted
v1:CreditCard3DSecure/	
v1:PayerSecurityLevel	Required for 3D Secure transactions
v1:AuthenticationValue	See 0 8.3 CreditCard 3DSecure for details
v1:XID	See 0 8.3 CreditCard 3DSecure for details
v1:Payment/	
v1:ChargeTotal	Required – Must be > $0.00
v1:SubTotal	Optional
v1:VATTax	Optional
v1:Shipping	Optional
v1:TransactionDetails/	
v1:UserID	Optional

FIELD	REQUIRED
v1:InvoiceNumber	Optional
v1:Ip	Optional
v1:Billing/	*To prevent the possibility of downgrading, some Billing data is required for all MOTO & ECI transactions!*
v1:CustomerID	Optional
v1:Name	MOTO & ECI: **Required** Retail: **Optional**
v1:Company	Optional
v1:Address1	MOTO & ECI: **Required** Retail: **Optional**
v1:Address2	Optional
v1:City	MOTO & ECI: **Required** Retail: **Optional**
v1:State	MOTO & ECI: **Required** Retail: **Optional**
v1:Zip	MOTO & ECI: **Required** Retail: **Optional**
v1:Country	MOTO & ECI: **Required** Retail: **Optional**
v1:Phone	Optional
v1:Fax	Optional
v1:Email	Optional: *Required if receipts are to be emailed to customer and administrator*
v1:Shipping/	
v1:Type	Optional
v1:Name	Optional
v1:Address1	Optional
v1:Address2	Optional
v1:City	Optional
v1:State	Optional
v1:Zip	Optional
v1:Country	Optional
v1:Phone	Optional
v1:Fax	Optional
v1:Email	Optional

6.1.7   Void

The following code is a sample of a Void transaction using the minimum required elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
```

```
  xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <v1:Transaction>
      <v1:CreditCardTxType>
         <v1:Type>void</v1:Type>
      </v1:CreditCardTxType>
      <v1:TransactionDetails>
         <v1:OrderId>
            62e3b5df-2911-4e89-8356-1e49302b1807
         </v1:OrderId>
         <v1:TDate>1190244932</v1:TDate>
      </v1:TransactionDetails>
   </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required and fields for the Void transaction. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:CreditCardTxType/	
v1:Type	Required
v1:TransactionDetails/	
v1:OrderId	Required
v1:TDate	Required

6.2   Check Transactions

Regardless of the transaction type, the basic XML document structure of a check transaction is as follows:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <v1:Transaction>
      <v1:TeleCheckTxType>...</v1:TeleCheckTxType>
      <v1:TeleCheckData>...</v1:TeleCheckData>
      <v1:Payment>...</v1:Payment>
      <v1:TransactionDetails>...</v1:TransactionDetails>
      <v1:Billing>...</v1:Billing>
      <v1:Shipping>...</v1:Shipping>
   </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The element TeleCheckTXType is mandatory for all check transactions. The other elements depend on the transaction type. The content depends on the type of transaction.

See 7 XML Tag Reference on page 38 for details of all required and optional elements needed for submission for check transactions

6.2.1 Sale

The following code is a sample of a check Sale transaction using the minimum required elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <v1:Transaction>
      <v1:TeleCheckTxType>
         <v1:Type>sale</v1:Type>
      </v1:TeleCheckTxType>
      <v1:TeleCheckData>
         <v1:CheckNumber>111</CheckNumber>
         <v1:AccountType>pc</AccountType>
         <v1:AccountNumber>1234567890</AccountNumber>
         <v1:RoutingNumber>055001054</RoutingNumber>
         <v1:DrivingLicenseNumber>U12345678</DrivingLicenseNumber>
         <v1:DrivingLicenseState>CA</DrivingLicenseState>
      </v1:TeleCheckData>
      <v1:Payment>
         <v1:ChargeTotal>19.95</v1:ChargeTotal>
      </v1:Payment>
   </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required and optional fields for the Sale transaction. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:TeleCheckTxType/	
v1:Type	Required
v1:TeleCheckData/	
v1:CheckNumber	Required
v1:AccountType	Required
v1:AccountNumber	Required
v1:RoutingNumber	Required
v1:DrivingLicenseNumber	Required
v1:DrivingLicenseState	Required
v1:Payment/	
v1:ChargeTotal	Required – Can be ≥ $0.00
v1:SubTotal	Optional
v1:VATTax	Optional
v1:Shipping	Optional

FIELD	REQUIRED
v1:TransactionDetails/	
v1:UserID	Optional
v1:InvoiceNumber	Optional
v1:OrderId	Optional
v1:Ip	Optional – *Refers to cardholder's IP address for web transactions; not the IP address of the merchant's data center.*
v1:ReferenceNumber	Optional
v1:TDate	Optional
v1:Recurring	Optional
v1:TaxExempt	Optional
v1:TerminalType	Optional
v1:TransactionOrigin	Optional, default value if not provided
v1:PONumber	Optional
v1:Billing/	*To prevent the possibility of downgrading, some Billing data is required for all MOTO & ECI transactions!*
v1:CustomerID	Optional
v1:Name	MOTO & ECI: **Required** Retail: **Optional**
v1:Company	Optional
v1:Address1	MOTO & ECI: **Required** Retail: **Optional**
v1:Address2	Optional
v1:City	MOTO & ECI: **Required** Retail: **Optional**
v1:State	MOTO & ECI: **Required** Retail: **Optional**
v1:Zip	MOTO & ECI: **Required** Retail: **Optional**
v1:Country	MOTO & ECI: **Required** Retail: **Optional**
v1:Phone	Optional
v1:Fax	Optional
v1:Email	Optional: *Required if receipts are to be emailed to customer and administrator*

6.2.2   Return

The following code is a sample of a Check Return transaction using the minimum required elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
```

```
   xmlns:fdggwsapi=
“http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
     <v1:Transaction>
       <v1:TeleCheckTxType>
          <v1:Type>return</v1:Type>
       </v1:TeleCheckTxType>
       <v1:Payment>
          <v1:ChargeTotal>19.95</v1:ChargeTotal>
       </v1:Payment>
       <v1:TransactionDetails>
          <v1:OrderId>
             62e3b5df-2911-4e89-8356-1e49302b1807
          </v1:OrderId>
       </v1:TransactionDetails>
     </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required fields for the Return transaction. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:TeleCheckTxType/	
v1:Type	Required
v1:Payment/	
v1:ChargeTotal	Required – Must be > $0.00
v1:TransactionDetails/	
v1:OrderId	Required

6.2.3   Void

The following code is a sample of a Check Void transaction using the minimum required elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
”http://secure.linkpt.net/fdggwsapi/schemas_us/v1”
 xmlns:fdggwsapi=
“http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
     <v1:Transaction>
       <v1: TeleCheckTxType>
          <v1:Type>void</v1:Type>
       </v1: TeleCheckTxType>
       <v1:Payment>
          <v1:ChargeTotal>19.95</v1:ChargeTotal>
       </v1:Payment>
       <v1:TransactionDetails>
          <v1:OrderId>
             62e3b5df-2911-4e89-8356-1e49302b1807
          </v1:OrderId>
       </v1:TransactionDetails>
```

```
      </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required fields for the Void transaction. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:TeleCheckTxType/	
v1:Type	Required
v1:Payment/	
v1:ChargeTotal	Required – Must be > $0.00
v1:TransactionDetails/	
v1:OrderId	Required
v1:TDate	Required

6.3   Shipping and Tax

Regardless of the transaction type, the basic XML document structure of a tax or shipping charge calculation is as follows:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <v1:Transaction>
      <v1:Calculate...>
          ...
      </v1:Calculate...>
   </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

See 7 XML Tag Reference on page 38 for details of all required and optional elements needed for tax or shipping charge calculations

6.3.1   Calculate Shipping

The following code is a sample of a shipping charge calculation using the minimum required elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <v1:Transaction>
      <v1:CalculateShipping>
         <v1:SubTotal>12.0</v1:SubTotal>
```

```
            <v1:Weight>1.2000000476837158</v1:Weight>
            <v1:ItemCount>1</v1:ItemCount>
            <v1:CarrierType>2</v1:CarrierType>
            <v1:ShipState>CA</v1:ShipState>
        </v1:CalculateShipping>
    </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required and optional fields for the shipping charge calculation. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:CalculateShipping/	
v1:SubTotal	Required
v1:Weight	Required
v1:ItemCount	Required
v1:CarrierType	Required
v1:ShipState	Required

6.3.2   Calculate Tax

The following code is a sample of a tax calculation using the minimum required elements:

```
<fdggwsapi:FDGGWSApiOrderRequest
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
    <v1:Transaction>
        <v1:CalculateTax>
            <v1:SubTotal>12.0</v1:SubTotal>
            <v1:ShipState>CA</v1:ShipState>
            <v1:ShipZip>93065</v1:ShipZip>
        </v1:CalculateTax>
    </v1:Transaction>
</fdggwsapi:FDGGWSApiOrderRequest>
```

The following table lists the required and optional fields for the tax calculation. All paths are relative to fdggwsapi:FDGGWSApiOrderRequest/v1:Transaction.

FIELD	REQUIRED
v1:CalculateTax/	
v1:SubTotal	Required
v1:ShipState	Required

FIELD	REQUIRED
v1:ShipZip	Required

6   Additional Web Service Actions

In addition to credit card and check transactions, the First Data Global Gateway Web Service API supports actions for recurring payments and a system check to test if the system is online.

Web service actions are contained in the fdggwsapi:FDGGWSApiActionRequest element.

7.1   Recurring Payments

The Recurring Payment action allows you to install, modify or cancel recurring credit card and check payments.

7.1.1   Install Recurring Payment

Use Recurring Payment to schedule credit card or check payments for future dates. The transactions can begin on the current date. If you set the start date as the current date, the first transaction processes immediately. This feature can schedule a single transaction in the future. You cannot set a start date in the past.

The following example shows how to install a recurring credit card, **once a month for 12 months**, starting on December 31, 2011:

```
<fdggwsapi:FDGGWSApiActionRequest
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi"
 xmlns:a1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/a1"
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1">
   <a1:Action>
      <a1:RecurringPayment>
         <a1:RecurringPaymentInformation>
            <a1:RecurringStartDate>20111231</a1:RecurringStartDate>
            <a1:InstallmentCount>12</a1:InstallmentCount>
            <a1:InstallmentFrequency>1</a1:InstallmentFrequency>
            <a1:InstallmentPeriod>month</a1:InstallmentPeriod>
         </a1:RecurringPaymentInformation>
         <a1:TransactionDataType>
            <a1:CreditCardData>
               <v1:CardNumber>4012000033330026</v1:CardNumber>
               <v1:ExpMonth>12</v1:ExpMonth>
               <v1:ExpYear>12</v1:ExpYear>
            </a1:CreditCardData>
         </a1:TransactionDataType>
         <v1:Payment>
            <v1:ChargeTotal>10.00</v1:ChargeTotal>
            <v1:SubTotal>5.00</v1:SubTotal>
         </v1:Payment>
         <v1:Shipping>
            <v1:Address1>...</v1:Address1>
            <v1:Carrier>...</v1:Carrier>
            <v1:City>...</v1:City>
```

```
            <v1:Country>...</v1:Country>
            <v1:Items>...</v1:Items>
            <v1:State>...</v1:State>
            <v1:Total>...</v1:Total>
            <v1:Weight>...</v1:Weight>
         </v1:Shipping>
         <v1:Billing>
            <v1:Address1>...</v1:Address1>
            <v1:City>...</v1:City>
            <v1:Country>...</v1:Country>
            <v1:State>...</v1:State>
            <v1:Zip>...</v1:Zip>
         </v1:Billing>
         <v1:TransactionDetails>
            <v1:InvoiceNumber>...</v1:InvoiceNumber>
            <v1:TransactionOrigin>...</v1:TransactionOrigin>
            <v1:UserID>...</v1:UserID>
            <v1:DeviceID>…</v1:DeviceID>
         </v1:TransactionDetails>
         <a1:Function>install</a1:Function>
      </a1:RecurringPayment>
   </a1:Action>
</fdggwsapi:FDGGWSApiActionRequest>
```

The following example shows how to install a **single** check payment on December 31, 2011:

```
<fdggwsapi:FDGGWSApiActionRequest
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi"
 xmlns:a1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/a1"
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1">
   <a1:Action>
      <a1:RecurringPayment>
         <a1:RecurringPaymentInformation>
            <a1:RecurringStartDate>20111231</a1:RecurringStartDate>
            <a1:InstallmentCount>1</a1:InstallmentCount>
            <a1:InstallmentFrequency>1</a1:InstallmentFrequency>
            <a1:InstallmentPeriod>month</a1:InstallmentPeriod>
         </a1:RecurringPaymentInformation>
         <a1:TransactionDataType>
            <a1:TeleCheckData>
               <v1:CheckNumber>111</CheckNumber>
               <v1:AccountType>pc</AccountType>
               <v1:AccountNumber>1234567890</AccountNumber>
               <v1:RoutingNumber>055001054</RoutingNumber>
               <v1:DrivingLicenseNumber>U12345678</DrivingLicenseNumber>
               <v1:DrivingLicenseState>CA</DrivingLicenseState>
            </a1:TeleCheckData>
         </a1:TransactionDataType>
         <v1:Payment>
            <v1:ChargeTotal>1</ns3:ChargeTotal>
         </v1:Payment>
         <v1:Shipping>
            <v1:Address1>...</v1:Address1>
            <v1:Carrier>...</v1:Carrier>
```

```
                <v1:City>...</v1:City>
                <v1:Country>...</v1:Country>
                <v1:Items>...</v1:Items>
                <v1:State>...</v1:State>
                <v1:Total>...</v1:Total>
                <v1:Weight>...</v1:Weight>
            </v1:Shipping>
            <v1:Billing>
                <v1:Address1>...</v1:Address1>
                <v1:City>...</v1:City>
                <v1:Country>...</v1:Country>
                <v1:State>...</v1:State>
                <v1:Zip>...</v1:Zip>
            </v1:Billing>
            <v1:TransactionDetails>
                <v1:InvoiceNumber>...</v1:InvoiceNumber>
                <v1:TransactionOrigin>...</v1:TransactionOrigin>
                <v1:UserID>...</v1:UserID>
            </v1:TransactionDetails>
            <a1:Function>install</a1:Function>
        </a1:RecurringPayment>
    </a1:Action>
</fdggwsapi:FDGGWSApiActionRequest>
```

The following table describes the optional or required fields for installing a recurring transaction. In addition, you must submit the data required for a credit card or check sale transaction. All paths are relative to fdggwsapi:FDGGWSApiActionRequest / a1:Action/ a1:RecurringPayment.

FIELD	REQUIRED
a1:Function	Required
a1:RecurringPaymentInformation	
a1:RecurringStartDate	Required
a1:InstallmentCount	Required
a1:InstallmentFrequency	Required
a1:InstallmentPeriod	Required
a1:MaximumFailures	Required

7.1.2   Modify Recurring Payment

The following example shows how to modify an existing recurring payment using the Order ID of the original instalment:

```
<fdggwsapi:FDGGWSApiActionRequest
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi"
 xmlns:a1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/a1"
 xmlns:v1=
```

```
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1">

   <a1:Action>
      <a1:RecurringPayment>
         <a1:Function>modify</a1:Function>
         <v1:Billing>...</v1:Billing>
         <v1:Shipping>...</v1:Shipping>
         <a1:OrderId>
             e368a525-173f-4f56-9ae2-beb4023a6993
         </a1:OrderId>
         <a1:RecurringPaymentInformation>
             <a1:InstallmentCount>999</a1:InstallmentCount>
         </a1:RecurringPaymentInformation>
      </a1:RecurringPayment>
      <v1:TransactionDetails>
             <v1:InvoiceNumber>...</v1:InvoiceNumber>
             <v1:TransactionOrigin>...</v1:TransactionOrigin>
             <v1:UserID>...</v1:UserID>
             <v1:DeviceID>…</v1:DeviceID>
      </v1:TransactionDetails>
   </a1:Action>
</fdggwsapi:FDGGWSApiActionRequest>
```

You can modify both the recurring payment information and the transaction details. Simply, include the fields that need to be modified. Some dependent fields may be required, for example, you must update the expiration date if you update the card number.

The following table describes the optional or required fields for modifying a recurring transaction. For credit card transaction fields, see 0 6.1.1   Sale on page 14; for check, see 0 6.2.1   Sale on page 27 for details. (v1:Billing and v1:Shipping are optional; however, transactions that do not include these elements may downgrade.) The transaction data must be submitted as a child of a1:RecurringPayment. All paths are relative to fdggwsapi:FDGGWSApiActionRequest / a1:Action/ a1:RecurringPayment.

FIELD	REQUIRED
a1:Function	Required
a1:OrderId	Required
a1:RecurringPaymentInformation	
a1:RecurringStartDate	Required
a1:InstallmentCount	Required
a1:InstallmentFrequency	Required
a1:InstallmentPeriod	Required
a1:MaximumFailures	Required

7.1.3   Cancel Recurring Payment

The following example shows how to cancel an existing recurring payment using the Order ID of the original instalment:

```
<fdggwsapi:FDGGWSApiActionRequest
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi"
 xmlns:a1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/a1"
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1">
   <a1:Action>
      <a1:RecurringPayment>
          <a1:Function>cancel</a1:Function>
          <a1:OrderId>
             e368a525-173f-4f56-9ae2-beb4023a6993
          </a1:OrderId>
      </a1:RecurringPayment>
   </a1:Action>
</fdggwsapi:FDGGWSApiActionRequest>
```

The following table describes the optional or required fields for cancelling a recurring transaction. All paths are relative to fdggwsapi:FDGGWSApiActionRequest / a1:Action/ a1:RecurringPayment.

FIELD	REQUIRED
a1:Function	Required
a1:OrderId	Required

7.2   SystemCheck

The SystemCheck action allows you to check that the First Data Global Gateway Web Service API is currently available. Most integrators do not need to perform this check more frequently than once every 15 minutes; you should not perform this check more frequently than once every 5 minutes.

The following code is a sample of the SystemCheck call.

```
<fdggwsapi:FDGGWSApiActionRequest
 xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi"
 xmlns:a1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/a1"
 xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1">
   <a1:Action>
          <a1:SystemCheck/>
   </a1:Action>
</fdggwsapi:FDGGWSApiActionRequest>
```

7 XML Tag Reference

This chapter provides a reference for the XML elements used in sending transactions and actions to the First Data Global Gateway Web Service API.

8.1 CreditCardTxType

The following table describes the sub-elements of the v1:CreditCardTxType element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:Type	xs:string	The transaction type. Valid values are: sale ForceTicket preAuth postAuth Return Credit Void

8.2 CreditCardData

The following table describes the sub-elements of the v1:CreditCardData element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:CardNumber	xs:string	The customer's credit card number. The string contains *only* digits; passing the number in the format *xxxx-xxxx-xxxx-xxxx* will result in an error.
v1:ExpMonth	xs:string	The expiration month of the customer's credit card. The content of this element always contains *two* digits, for example, use 07 for July.
v1:ExpYear	xs:string	The expiration year of the customer's credit card. The content of this element always contains *two* digits, for example, use 09 for 2009.
v1:CardCodeValue	xs:string	The three (3) or four (4) digit card security code (CSC), card verification value (CVV) or code (CVC), which is typically printed on the back of the credit card. For information about using CSC, contact support.

ELEMENT	DATA TYPE	DESCRIPTION
v1:CardCodeIndicator	xs:string	Indicates why the card code value was not provided. Valid values are: NOT_PROVIDED PROVIDED ILLEGIBLE NO_IMPRINT NOT_PRESENT
v1:TrackData	xs:string	The track data of a card when using a card reader instead of keying in card data. Use this value instead CardNumber, ExpMonth and ExpYear when swiping the card. This field needs to contain either track 1 data, track 2 data, or concatenated track 1 and 2 data. Concatenated track data must include the track and field separators, as they are stored on the card. Track 1 and track 2 data are in the format: %<track 1?;<track 2>?

8.3   CreditCard 3DSecure

The following table describes the sub-elements of the v1:CreditCard3DSecure element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:PayerSecurityLevel	xs:string	The two-digit PayerSecurityLevel returned by your Merchant Plug-in.
v1:AuthenticationValue	xs:string	The AuthenticationValue (MasterCard: AAV or VISA: CAAV) returned by your Merchant Plug-in.
v1:XID	xs:string	The XID returned by your Merchant Plug-in.

Note: You receive these values from your Merchant Plug-in for 3D Secure or a 3D Secure provider. The 3D Secure functionality of First Data Global Gateway Connect cannot be used for transactions via the Web Service API.

8.4   Payment

The following table describes the sub-elements of the v1:Payment element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:ChargeTotal	xs:double	The total transaction amount, including tax, VAT, and shipping amounts. The number of positions after the decimal point must not exceed 2. 3.123 is invalid. 3.12, 3.1, and 3 are valid. For sale transaction type, can be $0.00.
v1:SubTotal	xs:double	The sub total amount of the transaction, not including tax, VAT, or shipping amounts.
V1:Tax	xs:double	Tax amount of the transaction
v1:VATTax	xs:double	VAT tax amount
v1:Shipping	xs:double	Shipping amount of the transaction

8.5   TransactionDetails

The following table describes the sub-elements of the v1:TransactionDetails element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:UserID	xs:string	User ID of the user who performed the transaction. This value is used for reporting.
v1:InvoiceNumber	xs:string	Invoice number assigned by the merchant.
v1:OrderId	xs:string	Order ID This must be unique for the Store ID. If no Order ID is transmitted, the Web Service API assigns a value. The Order ID generated by Web Service can have a maximum of 100 digits. The Order ID field should not contain the following characters: & % /.
v1:Ip	xs:string	**Customer's** IP address which can be used by the Web Service API for fraud detection by IP address. Must be in the format *xxx.xxx.xxx.xxx*, for example 128.0.10.2 is a valid IP.

ELEMENT	DATA TYPE	DESCRIPTION
v1:ReferenceNumber	xs:string	The six (6) digit reference number received as the result of a successful external authorization (for example, by phone). This value is required for mapping a ForceTicket transaction to a previous authorization.
v1:TDate	xs:string	The TDate of the Sale, PostAuth, ForceTicket, Return, or Credit transaction referred to by a Void transaction. The TDate value is returned in the response to a successful transaction. When performing a Void transaction, the TDate and OrderId of the original transaction are required.
v1:Recurring	xs:string	Indicates if the transaction is a recurring transaction. Valid values are: Yes No
v1:TaxExempt	xs:string	Indicates if the transaction is exempt from tax. Valid values are: Yes No
v1:TerminalType	xs:string	The type of the terminal performing the transaction, up to 32 characters. Valid values are: Standalone – point-of-sale credit card terminal POS – electronic cash register or integrated POS system Unattended – self-service station Unspecified – e-commerce, general, CRT, or other applications
v1:TransactionOrigin	xs:string	The source of the transaction. Valid values are: ECI - email or Internet MOTO - mail order / telephone order RETAIL - face to face
v1:PONumber	xs:string	The purchase order number of the transaction, if applicable.
v1:DeviceID	xs:string	Data to help identify potential fraud on the consumer's computer

8.6 Billing

The following table describes the sub-elements of the v1:Billing element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:CustomerID	xs:string	Merchant's ID for the customer.
v1:Name	xs:string	Customer's Name - If provided, it will appear on your transaction reports.
v1:Company	xs:string	Customer's company. If provided, it will appear on your transaction reports.
v1:Address1	xs:string	The first line of the customer's address. If provided, it will appear on your transaction reports.
v1:Address2	xs:string	The second line of the customer's address. If provided, it will appear on your transaction reports.
v1:City	xs:string	Customer's city. If provided, it will appear on your transaction reports.
v1:State	xs:string	Customer's state - If provided, it will appear on your transaction reports.
v1:Zip	xs:string	Customer's ZIP code - If provided, it will appear on your transaction reports.
v1:Country	xs:string	Customer's country - If provided, it will appear on your transaction reports.
v1:Phone	xs:string	Customer's phone number - If provided, it will appear on your transaction reports.
v1:Fax	xs:string	Customer's fax number - If provided, it will appear on your transaction reports.
v1:Email	xs:string	Customer's email address - If provided, it will appear on your transaction reports.

8.7 Shipping

The following table describes the sub-elements of the v1:Shipping element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:Type	xs:string	Shipping Method
v1:Name	xs:string	Recipient's name - If provided, it will appear on your transaction reports.

ELEMENT	DATA TYPE	DESCRIPTION
v1:Address1	xs:string	The first line of the shipping address. If provided, it will appear on your transaction reports.
v1:Address2	xs:string	The second line of the shipping address. If provided, it will appear on your transaction reports.
v1:City	xs:string	Recipient's city - If provided, it will appear on your transaction reports.
v1:State	xs:string	Recipient's state - If provided, it will appear on your transaction reports.
v1:Zip	xs:string	Recipient's ZIP Code - If provided, it will appear on your transaction reports.
v1:Country	xs:string	Recipient's country - If provided, it will appear on your transaction reports.
v1:Carrier	xs:integer	Integer code defined by the merchant identifying the carrier type
v1:Total	xs:double	The transaction amount prior to calculating shipping. The number of positions after the decimal point must not exceed 2. 3.123 is invalid. 3.12, 3.1, and 3 are valid.
v1:Weight	xs:double	The weight of the item shipped, in pounds or kilograms as determined by the merchant.

8.8 TeleCheckTxType

The following table describes the sub-elements of the v1:TeleCheckTxType element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:Type	xs:string	Valid transaction type values are: Sale Void Return

8.9 TeleCheckData

The following table describes the sub-elements of the v1:TeleCheckData element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:CheckNumber	xs:string	Customer's check number
v1:AccountType	xs:string	Valid type of account values are: PC – Primary checking PS – Primary savings BC – Backup checking BS – Backup savings
v1:AccountNumber	xs:string	Checking Account Number
v1:RoutingNumber	xs:string	Customer's Bank Routing Number
v1:DrivingLicenseNumber	xs:string	Customer's Driver's License Number
v1:DrivingLicenseState	xs:string	The two-digit abbreviation for the state that issues the Driver's License.

8.10 CalculateShipping

The following table describes the sub-elements of the v1:CalculateShipping element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:SubTotal	xs:double	Transaction amount prior to calculating shipping. The number of positions after the decimal point must not exceed 2. 3.123 is invalid. 3.12, 3.1, and 3 are valid.
v1:Weight	xs:double	The weight of the item being shipped, in pounds or kilograms as determined by the merchant.
v1:ItemCount	xs:integer	Number of items being shipped.
v1:CarrierType	xs:integer	Integer code defined by the merchant identifying the carrier type
v1:ShipState	v1:Zip	Two-digit state abbreviation for the shipping destination

8.11 CalculateTax

The following table describes the sub-elements of the v1:CalculateTax element:

ELEMENT	DATA TYPE	DESCRIPTION
v1:SubTotal	xs:double	Transaction amount prior to calculating tax. The number of positions after the decimal point must not exceed 2. 3.123 is invalid. 3.12, 3.1, and 3 are valid.

ELEMENT	DATA TYPE	DESCRIPTION
v1:ShipState	xs:string	Two-digit state abbreviation for the shipping destination
v1:ShipZip	v1:Zip	ZIP code of the shipping destination.

8.12   RecurringPayment

The following table describes the sub-elements of the a1:RecurringPayment element:

ELEMENT	DATA TYPE	DESCRIPTION
a1:Function	xs:string	The type of recurring payment transaction. Valid values are: install modify cancel
a1:OrderId	xs:string	Order ID of the recurring payment being modified or cancelled
a1:RecurringPaymentInformation	Complex	Contains the elements defining the recurring payment
a1:RecurringStartDate	xs:string	Start Date of the recurring payment transaction in YYYYMMDD format. This value cannot be in the past.
a1:InstallmentCount	xs:string	Number of instalments of the recurring payment
a1:InstallmentFrequency	xs:string	Frequency of the instalment. Combines with the InstallmentPeriod to determine when the instalments occur. For example, use 2 for InstallmentFrequency and week for InstallmentPeriod for bi-weekly payments. Use 1 and month for monthly.
a1:InstallmentPeriod	xs:string	The period of the instalment. Combines with the InstallmentFrequency to determine when the instalments occur. Valid values are: Day Week Month Year

8 Building a SOAP Request Message

The next step after building your transaction in XML is to build the SOAP envelope that wraps the transaction.

The format for a SOAP envelope wrapping an operation sent to the First Data Global Gateway Web Service API is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header />
    <SOAP-ENV:Body>
        <!-- Transaction or action XML -->
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The SOAP message contains a SOAP envelope with a header and message body. The Web Service API does not require any headers for the SOAP message. The body contains the transaction or action XML as defined in the previous sections. There are no further requirements for mapping the type of transaction or action in the SOAP envelope. The Web Service API maps the operation based on the content of the body.

For example, the complete SOAP message for a credit sale transaction looks like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header />
    <SOAP-ENV:Body>
        <fdggwsapi:FDGGWSApiOrderRequest xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
<v1:Transaction xmlns:v1=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1">
            <v1:CreditCardTxType>
                <v1:Type>sale</v1:Type>
            </v1:CreditCardTxType>
            <v1:CreditCardData>
                <v1:CardNumber>4012000033330026</v1:CardNumber>
                <v1:ExpMonth>12</v1:ExpMonth>
                <v1:ExpYear>12</v1:ExpYear>
            </v1:CreditCardData>
            <v1:Payment>
                <v1:ChargeTotal>120</v1:ChargeTotal>
            </v1:Payment>
        </v1:Transaction>
        </fdggwsapi:FDGGWSApiOrderRequest>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

9 Reading the SOAP Response Message

The First Data Global Gateway Web Service API returns a SOAP message in response to your transaction or action request.

- If your request is successful, the Web Service API returns an fdggwsapi:FDGGWSApiOrderResponse or fdggwsapi:FDGGWSApiActionResponse in the body of the SOAP message.
- If your request is unsuccessful, the Web Service API returns a SOAP fault message.

Both SOAP message types are contained in the body of the HTTP response message.

10.1 SOAP Response Message

10.1.1 Transaction

The First Data Global Gateway Web Service API returns a SOAP response message when your transaction is successful and the Web Service API is able to return an approved or declined response. The response message has the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header />
   <SOAP-ENV:Body>
      <fdggwsapi:FDGGWSApiOrderResponse xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <!-- transaction result -->
      </fdggwsapi:FDGGWSApiOrderResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The SOAP response contains no headers. The SOAP body contains the actual transaction result contained in the fdggwsapi:FDGGWSApiOrderResponse element. The sub-elements are defined in Analyzing the Transaction Response on page 52. The following is an example of the SOAP message returned for an approved Sale transaction:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header />
   <SOAP-ENV:Body>
      <fdggwsapi:FDGGWSApiOrderResponse xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <fdggwsapi:CommercialServiceProvider>
            CSI
        </fdggwsapi:CommercialServiceProvider>
        <fdggwsapi:TransactionTime>
           Tue Nov 03 09:35:05 2009
        </fdggwsapi:TransactionTime>
        <fdggwsapi:TransactionID>
```

```
                    2000486340
            </fdggwsapi:TransactionID>
            <fdggwsapi:ProcessorReferenceNumber>
                OK289C
            </fdggwsapi:ProcessorReferenceNumber>
            <fdggwsapi:ProcessorResponseMessage>
                APPROVED
            </fdggwsapi:ProcessorResponseMessage>
            <fdggwsapi:ErrorMessage />
            <fdggwsapi:OrderId>
                A-eb0406bc-7eb8-419b-aa1a-7a4394e2c83e
            </fdggwsapi:OrderId>
            <fdggwsapi:ApprovalCode>
                OK289C0003529354:NNN:
            </fdggwsapi:ApprovalCode>
            <fdggwsapi:AVSResponse>PPX</fdggwsapi:AVSResponse>
            <fdggwsapi:TDate>1256168682</fdggwsapi:TDate>
            <fdggwsapi:TransactionResult>
                APPROVED
            </fdggwsapi:TransactionResult>
            <fdggwsapi:ProcessorResponseCode>
                A
            </fdggwsapi:ProcessorResponseCode>
            <fdggwsapi:ProcessorApprovalCode>
                440368
            </fdggwsapi:ProcessorApprovalCode>
            <fdggwsapi:CalculatedTax/>
            <fdggwsapi:CalculatedShipping/>
            <fdggwsapi:TransactionScore>
                496
            </ fdggwsapi:TransactionScore>
            <fdggwsapi:FraudAction>
                ACCEPT
            </fdggwsapi:FraudAction>
            <fdggwsapi:AuthenticationResponseCode>
                XXX
            </fdggwsapi:AuthenticationResponseCode>
        </fdggwsapi:FDGGWSApiOrderResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.1.2 Action

If you send an action, the Web Service API returns a fdggwsapi:FDGGWSApiActionResponse.

The response for a successful instalment, modification or cancellation or for a system check contains the value true for the parameter <fdggwsapi:Success>. The following is an example of the message returned for a successful action:

```
<fdggwsapi:FDGGWSApiActionResponse xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
    <fdggwsapi:Success>
        true
    </fdggwsapi:Success>
    <fdggwsapi:CommercialServiceProvider/>
```

```
        <fdggwsapi:TransactionTime>
            Tue Nov 03 10:00:58 2009
        </fdggwsapi:TransactionTime>
        <fdggwsapi:TransactionID/>
        <fdggwsapi:ProcessorReferenceNumber/>
        <fdggwsapi:ProcessorResponseMessage/>
        <fdggwsapi:ErrorMessage/>
        <fdggwsapi:OrderId>
            A-3384d07e-699a-48d3-a44a-61ccefde0524
        </fdggwsapi:OrderId>
        <fdggwsapi:ApprovalCode/>
        <fdggwsapi:AVSResponse/>
        <fdggwsapi:TDate/>
        <fdggwsapi:TransactionResult>
            APPROVED
        </fdggwsapi:TransactionResult>
        <fdggwsapi:ProcessorResponseCode/>
        <fdggwsapi:ProcessorApprovalCode/>
        <fdggwsapi:TransactionScore>
            57
        </fdggwsapi:TransactionScore>
        <fdggwsapi:FraudAction>
            ACCEPT
        </fdggwsapi:FraudAction>

</fdggwsapi:FDGGWSApiActionResponse>
```

10.2 SOAP Fault Message

The First Data Global Gateway Web Service API returns a SOAP fault message when your request is unsuccessful. The fault message has the following format:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header />
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Client</faultcode>
            <faultstring xml:lang="en-US">
                <!-- fault message -->
            </faultstring>
            <detail>
                <!-- fault message -->
            </detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The SOAP fault message may contain the following elements:

ELEMENT	DATA TYPE	DESCRIPTION
faultcode	xs:string	Defines where the error occurred. Valid values are:

		SOAP-ENV:Server SOAP-ENV:Client
faultstring	xs:string	Defines the fault type
detail	xs:string	Additional data depending on the fault type

The possible return values by faultcode and faultstring are defined in the following sections.

10.2.1 SOAP-ENV:Server

The SOAP-ENV:Server faultcode indicates that the Web Service API has failed to process your transaction due to an internal system error. If you receive this as response, contact support to resolve the problem.

The SOAP-ENV:Server message has the following format:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header />
   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Server</faultcode>
         <faultstring xml:lang="en-US">
            unexpected error
         </faultstring>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The SOAP SOAP-ENV:Envelope/SOAP-ENV:Body/SOAP-ENV:Fault contains the following elements:

ELEMENT	DATA TYPE	DESCRIPTION
faultcode	xs:string	This value is always: SOAP-ENV:Server
faultstring	xs:string	This value is always: unexpected error

10.1.2 SOAP-ENV:Client

The SOAP-ENV:Client response includes a MerchantException faultcode indicating that the Web Service API has found an error with the transaction you submitted. The MerchantException indicates that the XML or authorization data provided by the merchant is faulty. This may have one of the following reasons:

- Your store is registered as being closed. If you receive this message even though you believe, your store should be registered as Open, contact support.

- The store ID / user ID combination you have provided for HTTPS authorization is syntactically incorrect.
- The XML does not match the schema.

The MerchantException message has the following format:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header />
   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Client</faultcode>
         <faultstring xml:lang="en-US">
            MerchantException
         </faultstring>
         <detail>
            <!-- detailed explanation. -->
         </detail>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The SOAP SOAP-ENV:Envelope/SOAP-ENV:Body/SOAP-ENV:Fault contains the following elements:

ELEMENT	DATA TYPE	DESCRIPTION
faultcode	xs:string	This value is always: SOAP-ENV:Client
faultstring	xs:string	This value is always: MerchantException
detail/reason	xs:string	The Web Service API returns a minimum of one reason.

See 0 21.1 Merchant Exceptions on page 90 for detailed descriptions of errors.

10 Analyzing the Transaction Response

11.1 Approval Response

If a transaction is approved, the First Data Global Gateway Web Service API returns a SOAP response message. The body of the message contains

- fdggwsapi:FDGGWSApiOrderResponse

or

- fdggwsapi:FDGGWSApiActionResponse element.

The following table describes the sub-elements of the fdggwsapi:FDGGWSApiOrderResponse element. The Web Service API always returns all of the elements listed below; however, some of the elements may be empty.

ELEMENT	DATA TYPE	DESCRIPTION
fdggwsapi: CommercialServiceProvider	xs:string	Indicates your provider
fdggwsapi:TransactionTime	xs:string	The time stamp set by the First Data Global Gateway Web Service API before returning the transaction approval.
fdggwsapi: ProcessorReferenceNumber	xs:string	The reference number returned by the processor. This value may be empty, and is not needed in further transactions. This value may be needed if you have to contact support regarding a transaction.
fdggwsapi: ProcessorResponseMessage	xs:string	In case of an approval, this element contains the following string: APPROVED
fdggwsapi: ProcessorResponseCode	xs:string	Response Code from the credit card processor
fdggwsapi: ProcessorApprovalCode	xs:string	Approval Code from the credit card processor
fdggwsapi:ErrorMessage	xs:string	Error Message. This element is empty in case of an approval.

fdggwsapi:OrderId	xs:string	This element contains the order ID. For Sale, PreAuth, ForceTicket, and Credit transactions, a new order ID is returned. For PostAuth, Return, and Void transactions, supply this number in the v1:OrderId element for identifying the transaction to which you refer. The fdggwsapi:OrderId element of a response to a PostAuth, Return, or Void transaction simply returns the order ID of the original transaction. The OrderId generated by Web Service can have a maximum of 100 digits.
fdggwsapi:ApprovalCode	xs:string	The approval code returned by the processor. This value may be empty, and is not needed in further transactions. This value may be needed if you have to contact support regarding a transaction.
fdggwsapi:AVSResponse	xs:string	Address Verification System (AVS) response
fdggwsapi:TDate	xs:string	The TDate required for Void transactions. Only returned for Sale, ForcedTicket, and PostAuth.
fdggwsapi:TransactionResult	xs:string	The transaction result. Always APPROVED in case of an approval.
fdggwsapi:TransactionID	xs:string	The Transaction ID used for this transaction.
fdggwsapi:CalculatedTax	xs:string	Calculated tax for the transaction
fdggwsapi:CalculatedShipping	xs:string	Calculated shipping for the transaction.
fdggwsapi:TransactionScore	xs:string	A numerical value indicating the risk of fraud on the transaction. Higher values indicate a greater risk of fraud. The actual range used for this field has not yet been defined. This field is only returns a value for merchants who use the optional, add-on Fraud Service.

fdggwsapi: AuthenticationResponseCode	xs:string	Three (3) digit response code returned by processor for 3D Secure enabled transactions. See the 3DS integration guide for values and definitions.
		This field only returns a value for 3D Secure transactions for merchants who use this optional, add-on service. See Appendix 23 for values and definitions.
fdggwsapi : FraudAction	xs:string	This element can contain any of the below value. [ACCEPT, REJECT, ERROR and REVIEW]

11.2 Failure Response

If your transaction is declined or your action is rejected, the First Data Global Gateway Web Service API returns an fdggwsapi:FDGGWSApiOrderResponse or fdggwsapi:FDGGWSApiActionResponse element. The elements returned are the same as in the case of a successful transaction request. Only the values differ.

The following table describes the sub-elements of the fdggwsapi:FDGGWSApiOrderResponse element. The Web Service API always returns all of the elements listed below; however, some of the elements may be empty.

ELEMENT	DATA TYPE	DESCRIPTION
fdggwsapi: CommercialServiceProvider	xs:string	Indicates your provider
fdggwsapi:TransactionTime	xs:string	The time stamp set by the First Data Global Gateway Web Service API before returning the transaction approval.
fdggwsapi: ProcessorReferenceNumber	xs:string	Reference Number '000000' returned by the processor. This value may be empty, and is not needed in further transactions. This value may be needed if you have to contact support regarding a transaction.
fdggwsapi: ProcessorResponseMessage	xs:string	Error Message returned by the processor. This value might be empty.
fdggwsapi: ProcessorResponseCode	xs:string	Response Code from the credit card processor

ELEMENT	DATA TYPE	DESCRIPTION
fdggwsapi: ProcessorApprovalCode	xs:string	Approval Code from the credit card processor
fdggwsapi:ErrorMessage	xs:string	Error message returned by the First Data Global Gateway Web Service API. Returned in the format *SGS-XXXXXX: Message*, where *XXXXXX* is a six-digit error code and *Message* describing the error. This description might be different from the processor response message. For instance, in the above example the follow error message is returned: SGS-002304: Credit card is expired You may need this value if you have to contact support regarding a transaction.
fdggwsapi:OrderId	xs:string	The Order ID. In contrast to an approval, this Order ID is never required for any further transaction, but you may need this value if you have to contact support regarding a transaction. The Order ID generated by Web Service can have a maximum of 100 digits.
fdggwsapi:ApprovalCode	xs:string	This element is empty in case of a transaction failure.
fdggwsapi:AVSResponse	xs:string	Returns the Address Verification System (AVS) response NULL in declined transactions
fdggwsapi:TDate	xs:string	The TDate. Similar to the Order ID, the TDate is never required for any further transaction, but you may need this value if you have to contact support regarding a transaction.
fdggwsapi:TransactionResult	xs:string	Valid values are: DECLINED – the processor rejected the transaction, for example, for insufficient funds DUPLICATE – the OrderID is has been used previously FRAUD – fraud detected in the transaction FAILED – internal error at the Gateway

ELEMENT	DATA TYPE	DESCRIPTION
fdggwsapi:TransactionID	xs:string	Transaction ID used for this transaction
fdggwsapi:CalculatedTax	xs:string	Calculated tax for the transaction
fdggwsapi:CalculatedShipping	xs:string	Calculated shipping for the transaction.
fdggwsapi:TransactionScore	xs:string	A numerical value indicating the risk of fraud on the transaction. Higher values indicate a greater risk of fraud. The actual range used for this field has not yet been defined. This field is only returns a value for merchants who use the optional, fee-based fraud service.
fdggwsapi: AuthenticationResponseCode	xs:string	Returns a code in case of a transaction failure.

11.3 Authentication Response Code

A three (3) digit numeric response code returned on 3-D Secure enabled transactions. This field only returns a value for merchants who use this add-on service.

FIRST DIGIT	VALIDATION	DESCRIPTION
0	Failed	Validation and/or authentication
1	Failed	Validation and/or authentication
2	Passed	Validation and/or authentication
3	Passed	Authentication (A 3-D Secure authentication value of seven (7) from the issuer's ACS indicates authentication was attempted.)
4	Failed	Authentication (A 3-D Secure authentication value of seven (7) from the issuer's ACS indicates that authentication was attempted.)
5		Reserved for future use – not used
6	N/A	Issuer not participating in CAVV Validation
7	Failed	Authentication (A 3-D Secure authentication value of seven (7) from Visa's ACS indicates that an authentication attempt was performed.)
8	Passed	Authentication (A 3-D Secure authentication value of seven (7) from Visa's ACS indicates that an authentication attempt was performed and Visa generated this value from the use of Visa CAVV key[s])
9	Failed	Authentication (A 3-D Secure value of eight (8) from Visa ACS indicates that an authentication attempt was performed when the issuer's ACS was not available, and Visa generated this value from the use of Visa CAVV key[s].)
A	Passed	Validation – (A 3-D Secure authentication value of eight (8) from Visa's ACS indicates that an authentication attempt was performed when the issuer's ACS was not available. (Determined that Visa generated this value from the use of Visa's CAVV key[s].))
B	Passed	Validation - information only, no liability shift. If the ECI equals seven (7) and CAVV is present, the CAVV will be validated but no liability shift will occur. Visa will generate this value for card or transaction types that are not eligible for the 3-D Secure service.)
C	N/A	Not Validated – Issuer did not return a CAVV results code in the authorization response.
D	N/A	Not Validated authentication - Issuer did not return a CAVV results code in the authorization response.

11 Building an HTTPS POST Request

Generally, the tools one uses to communicate with the First Data Global Gateway Web Service API support the building of HTTPS POST requests. This document describes the process for doing this using the tools tested by First Data for accessing the Web Service API. If you are using another tool, consult the documentation

The following table describes the values you need to build an HTTPS POST request:

PARAMETER	VALUE	DESCRIPTION
URL	`https:// ws.firstdataglobalgateway.com/fdggwsapi/ services`	This is the full URL of the First Data Global Gateway Web Service API. Depending on the functionality you use for building HTTP requests, you might have to split this URL into host and service and provide this information in the appropriate HTTP request headers.
Content-Type	`text/xml`	Indicates that the SOAP message is encoded in XML and passed as content in the HTTP POST request body.
Authorization	Type: `Basic` Username: `WS<store ID>._.1` Password: *Password*	Identifies your store at the First Data Global Gateway Web Service API. The Authorization parameter takes the following format: Authorization: Basic <authstring> where <authstring> is the base-64 encoded result of the string <userid>:<password>. For example, if your user name is WS`101._.1`, and your password `myPW`, the complete HTTP authorization header would be: `Authorization: Basic V1MxMDEuXy4wMDc6bXlQV`

		w== The authorization string is the base 64 encoding result of the string `WS101._.1:myPW`.
HTTP Body	SOAP request XML	The HTTP POST request body contains SOAP request message.

12.1 PHP

One can use either the cURL library or the cURL command-line tool to communicate with the Web Service API using PHP. In recent PHP versions, the cURL library is included as an extension, which needs to be activated. While this is a straightforward task on Windows servers, it may require you to compile PHP on Unix/Linux machines. In this case, it may be easier to call the cURL command line tool from your PHP script.

12.1.1 Using the cURL PHP Extension

In PHP 5.2.9-2, activating the cURL extension simply requires you to uncomment the following line in your php.ini file:

```
;extension=php_curl.dll
```

Other PHP versions might require other actions in order to enable cURL. See your PHP documentation for more information. After activating cURL, use the following code to set up an HTTPS POST request:

```php
<?php
// storing the SOAP message in a variable – note that the plain XML code
// is passed here as string for reasons of simplicity, however, it is
// certainly a good practice to build the XML e.g. with DOM – furthermore,
// when using special characters, you should make sure that the XML string
// gets UTF-8 encoded (which is not done here):
$body = "<SOAP-ENV:Envelope ...>...</SOAP-ENV:Envelope>";
// initializing cURL with the FDGGWS API URL:
$ch =
curl_init("https://ws.firstdataglobalgateway.com/fdggwsapi/services/order.
wsdl");
// setting the request type to POST:
curl_setopt($ch, CURLOPT_POST, 1);
// setting the content type:
curl_setopt($ch, CURLOPT_HTTPHEADER, array("Content-Type: text/xml"));
// setting the authorization method to BASIC:
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
// supplying your credentials:
curl_setopt($ch, CURLOPT_USERPWD, "WS101._.1:myPW");
// filling the request body with your SOAP message:
curl_setopt($ch, CURLOPT_POSTFIELDS, $body);
...
?>
```

The next chapter discusses setting the security options, which are necessary for enabling SSL communication.

12.1.2 Using the cURL Command Line Tool

If one chooses to use the cURL command line tool, you do not need to perform any setup. The following script shows you how to call the command line tool from your PHP script and set the HTTPS POST request:

```php
<?php
// storing the SOAP message in a variable – note that you have to escape
// " and \n, since the latter makes the command line tool fail,
// furthermore note that the plain XML code is passed here as string
// for reasons of simplicity, however, it is certainly a good practice
// to build the XML e.g. with DOM – finally, when using special
// characters, you should make sure that the XML string gets UTF-8
// encoded  (which is not done here):
$body = "<SOAP-ENV:Envelope ...>...</SOAP-ENV:Envelope>";
// setting the path to the cURL command line tool – adapt this path to
// the path where you have saved the cURL binaries:
$path = "C:\curl\curl.exe";
// setting the FDGGWS API URL:
$apiUrl =
("https://ws.firstdataglobalgateway.com/fdggwsapi/services/order.wsdl");
// setting the content type:
$contentType = " --header \"Content-Type: text/xml\"";
// setting the authorization method to BASIC and supplying
// your credentials:
$user = " --basic --user WS101._.1:myPW";
// setting the request body with your SOAP message – this automatically
// marks the request as POST:
$data = " --data \"".$body."\"".
...
?>
```

12.2 ASP

WinHTTP 5.1 is included with Windows Server 2003 and Windows XP SP2. Use the following code to set up an HTTPS POST request:

```javascript
<%@ language="javascript"%>
<html>...<body>
<%
// storing the SOAP message in a variable – note that the plain XML code
// is passed here as string for reasons of simplicity, however, it is
// certainly a good practice to build the XML e.g. with DOM –
// furthermore, when using special characters, you should make sure that
// the XML string gets UTF-8 encoded (which is not done here):
var body = "<SOAP-ENV:Envelope ...>...</SOAP-ENV:Envelope>";
// constructing the request object:
var request = Server.createObject("WinHttp.WinHttpRequest.5.1");
// initializing the request object with the HTTP method POST
// and the FDGGWS API URL:
```

```
request.open("POST",
"https://ws.firstdataglobalgateway.com/fdggwsapi/services/order.wsdl");
// setting the content type:
request.setRequestHeader("Content-Type", "text/xml");
// setting the credentials:
request.setCredentials("WS111901._.1 ", "aTenvipB ", 0);
...
%>
</body></html>
```

The sample code fragment is written in JavaScript; using VB Script instead does not fundamentally change the code.

12 Establishing an SSL connection

One must establish a secure communication channel to send the HTTP request built in the previous chapter. This ensures that the data sent between your client application and the First Data Global Gateway Web Service API is encrypted and that both parties can be sure they are communicating with each other and no one else.

The Web Service API requires an SSL connection with client and server exchanging certificates to guarantee this level of security. The client and server certificates each uniquely identify the party. This process works as follows:

1. The client begins the process by sending its client certificate to the server.
2. The server receives the client certificate and verifies it against the client certificate it has stored for this client.
3. If valid, the server responds by sending its server certificate.
4. The client receives the server certificate and verifies it against the trusted server certificate.
5. If valid, both parties establish the SSL channel, as they can be sure that they are communicating with each other and no one else. All data exchanged between both parties is encrypted.

Following this process, your application has to do two things: First, start the communication by sending its client certificate. Second, verify the received server certificate. How this is accomplished differs from platform to platform. However, in order to illustrate the basic concepts, the PHP and ASP scripts started in the previous chapter will be continued by extending them with the relevant statements necessary for setting up an SSL connection.

13.1 PHP

Choose to use either the cURL extension or the cURL command line tool to integrate with the Web Service API using PHP. cURL requires the client certificate to be passed as a PEM file, the client certificate private key passed as a separate file, and the client certificate private key password to be supplied. While the private key is not technically necessary for establishing an SSL connection, it is required for doing so with cURL and PHP.

13.1.1 Using the PHP cURL Extension

The following code sample extends the script started in the previous chapter. The sample code shows how to supply the parameters necessary for establishing an SSL connection with the cURL extension:

```php
<?php
...
// configuring cURL not to verify the server certificate:
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
// setting the path where cURL can find the client certificate:
curl setopt($ch, CURLOPT SSLCERT, "C:\certs\WS101. .1.pem");
// setting the path where cURL can find the client certificate's
// private key:
curl_setopt($ch, CURLOPT_SSLKEY, "C:\certs\WS101._.1.key");
```

```
// setting the key password:
curl_setopt($ch, CURLOPT_SSLKEYPASSWD, " ckp_1256591851");
...
?>
```

The next chapter discusses sending the message and receiving the response.

13.1.2 Using the cURL Command Line Tool

The following code sample extends the script started in the previous chapter. The sample code shows how to supply the parameters necessary for establishing an SSL connection with the cURL command line tool:

```
<?php
...
// configuring cURL not to verify the server certificate:
$serverCert = " -k ";
// setting the path where cURL can find the client certificate:
$clientCert = " --cert C:\certs\WS101. .1.pem ";
// setting the path where cURL can find the client certificate's
// private key:
$clientKey = " --key C:\certs\WS101._.1.key";
// setting the key password:
$keyPW = " --pass  ckp_1256591851";
...
?>
```

The next chapter discusses sending the message and receiving the response.

13.2 ASP

Before one can communicate using SSL with the First Data Global Gateway Web Service API, you must install both client certificates in the certificate store. See 22 Installing the Client Certificate on page 97 for instructions on installing the client certificate.

The following code sample extends the script started in the previous chapter. The sample code shows how to set the path for WinHTTP to find the client certificate:

```
<%@ language="javascript"%>
<html>...<body>
<%
...
// setting the path where the client certificate to send can be found:
request.setClientCertificate("LOCAL_MACHINE\\My\\WS101._.1");
...
%>
</body></html>
```

If one uses VBScript instead of JavaScript, you must replace the double-backslashes in the path with single backslashes.

The next chapter discusses sending the message and receiving the response.

14 Sending the HTTPS POST Request and Receiving the Response

The final step in writing your client is sending the HTTPS POST request to the First Data Global Gateway Web Service API and receiving the response. Most HTTP libraries cover the underlying communication details and require only a single call that returns the HTTP response.

The First Data Global Gateway Web Service API returns a 200 status code and a SOAP response in response to a successful HTTP POST request. If you send any invalid HTTP POST parameters, the First Data Global Gateway Web Service API will return a standard HTTP error code. If you send invalid data (for example, an invalid credit card number) in the SOAP request message, the Web Service API will return a 500 status code and a SOAP fault message.

See 9 Reading the SOAP Response Message on page 47 for instructions on reading the SOAP response message.

14.1 PHP

Choose to use either the cURL extension or the cURL command line tool to integrate with the Web Service API using PHP.

14.1.1 Using the PHP cURL Extension

The sample code below shows how to complete the PHP cURL extension script by making the HTTPS POST request and receiving the response.

```php
<?php
...
// telling cURL to return the HTTP response body as operation result
// value when calling curl_exec:
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
// calling cURL and saving the SOAP response message in a variable which
// contains a string like "<SOAP-ENV:Envelope ...>...</SOAP-
ENV:Envelope>":
$result = curl_exec($ch);
// closing cURL:
curl_close($ch);
?>
```

The HTTPS call returns a SOAP response or fault message in the HTTP response body.

14.1.2 Using the cURL Command Line Tool

Performing the HTTPS POST request with the cURL command line tool simply requires executing the cURL command with the PHP exec command. The PHP exec command saves each line returned by an external program as an element of an array. Therefore, to get the complete HTTP response body, you must concatenate the elements of the array.

The sample code below shows how to complete the PHP cURL extension script by making the HTTPS POST request and receiving the response.

```php
<?php
...
// saving the whole command in one variable:
$curl = $path.
$data.
$contentType.
$user.
$serverCert.
$clientCert.
$clientKey.
$keyPW.
$apiUrl;
// preparing the array containing the lines returned by the cURL
// command line tool:
$returnArray = array();
// performing the HTTP call by executing the cURL command line tool:
exec($curl, $returnArray);
// preparing a variable taking the complete result:
$result = "";
// concatenating the different lines returned by the cURL command
// line tool – this result in the variable $result carrying the entire
// SOAP response message as string:
foreach($returnArray as $item)
$result = $result.$item;
?>
```

14.2 ASP

The sample code below shows how to complete the ASP script by calling the request's send method, with the SOAP XML as a parameter.

```asp
<%@ language="javascript"%>
<html>...<body>
<%
...
// doing the HTTP call with the SOAP request message as input:
request.send(body);
// saving the SOAP response message in a string variable:
var response = request.responseText;
%>
</body></html>
```

After the request is completed, you can access the response body through the responseText property of the request.

15 Using .NET Framework

First Data has tested the First Data Global Gateway Web Service API with the C# 2.0 .NET Framework.

15.1 Prerequisites

First, you need to install the client certificate (WS<Store_ID>._.1.p12*).* See 22 Installing the Client Certificate on page 97 for instructions on installing the client certificate.

The user executing the program has access to the certificate after installation. To do so, first download the WinHttpCertCfg tool from Microsoft. Use the following URL:

http://www.microsoft.com/downloads/details.aspx?familyid=c42e27ac-3409-40e9-8667-c748e422833f&displaylang=en

To grant access to the user, using the command line, navigate to the directory where you installed WinHttpCertCfg and enter the following command:

`winhttpcertcfg.exe -g -a ` *`OtherUserID`* ` -c LOCAL_MACHINE\MY -s WS`*`storeid._.1.p12`*

OtherUserID is the name of the user executing the application. `WS`*`storeid._.1.p12`* is the name of the client certificate. Replace this value with the name of your client certificate. The name should be in the format WS<store_ID>._.1. Verify this value when you install the client certificate using the instructions above.

You must also install the Web Service Enhancements (WSE) 3.0 for Microsoft .NET. Use the following URL to view the system requirements and download the installer:

http://www.microsoft.com/downloads/details.aspx?FamilyID=018a09fd-3a74-43c5-8ec1-8d789091255d&displaylang=en

15.2 Creating Web Service Reference Classes in .NET

To create the web service reference classes for your project in .NET, follow these steps:

1. Right-click on the project in the Solution Explorer and select **Add Web Reference**.

…
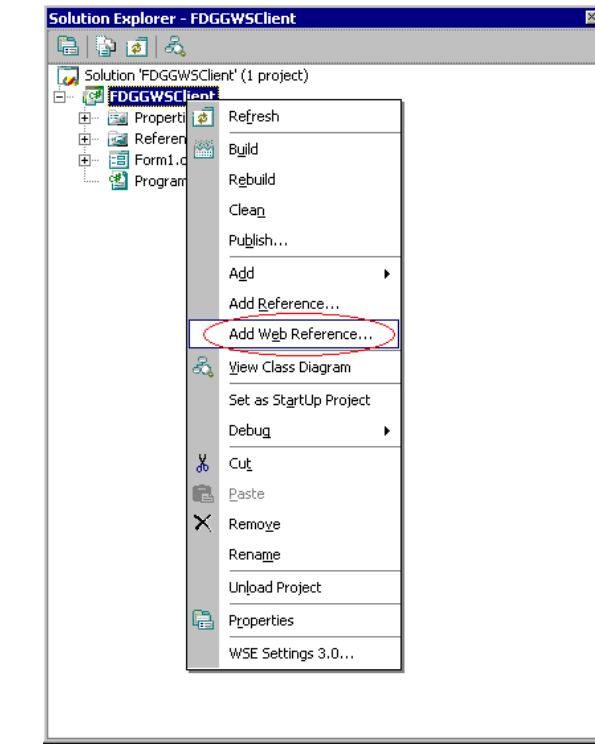
2. Download the wsdl from the below location.
    https://ws.firstdataglobalgateway.com/fdggwsapi/services/order.wsdl

3. Download the schemas from the below location.
    a. https://ws.firstdataglobalgateway.com/fdggwsapi/schemas_us/v1.xsd
    b. https://ws.firstdataglobalgateway.com/fdggwsapi/schemas_us/a1.xsd
    c. https://ws.firstdataglobalgateway.com/fdggwsapi/schemas_us/fdggwsapi.xsd

4. The following dialog displays. Enter the location of the WSDL file in the **URL** field.
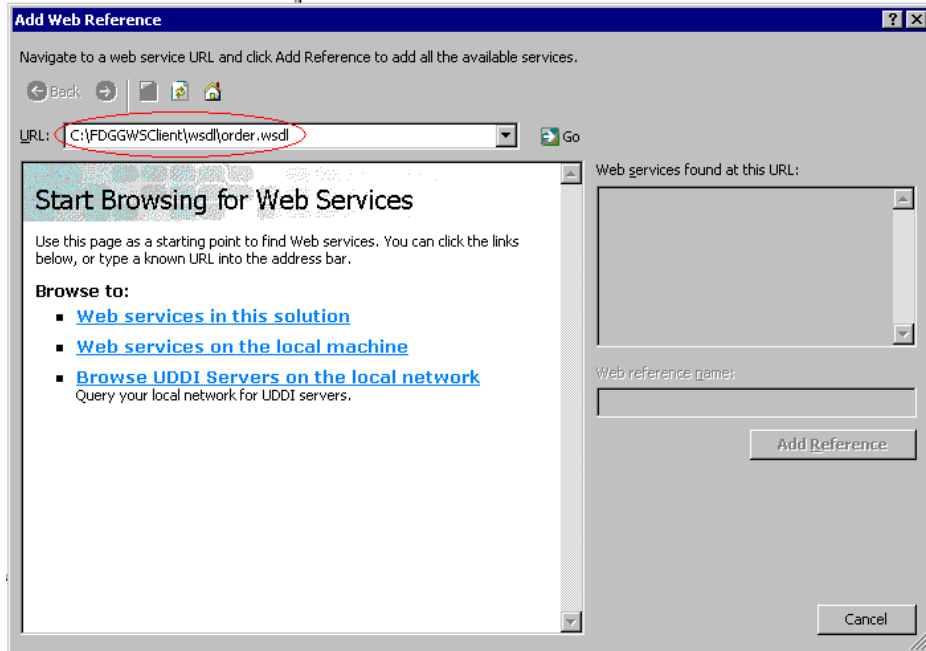
The root schema imports the other two schemas (v1 and a1) using relative URLs as
shown in the code below. The directory structure for your application needs to match the
directory structure shown in the schema file.

```
<xs:import namespace=" http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
schemaLocation="../schemas_us/v1.xsd" />
<xs:import namespace=" http://secure.linkpt.net/fdggwsapi/schemas_us/a1"
schemaLocation="../schemas_us/a1.xsd" />
```
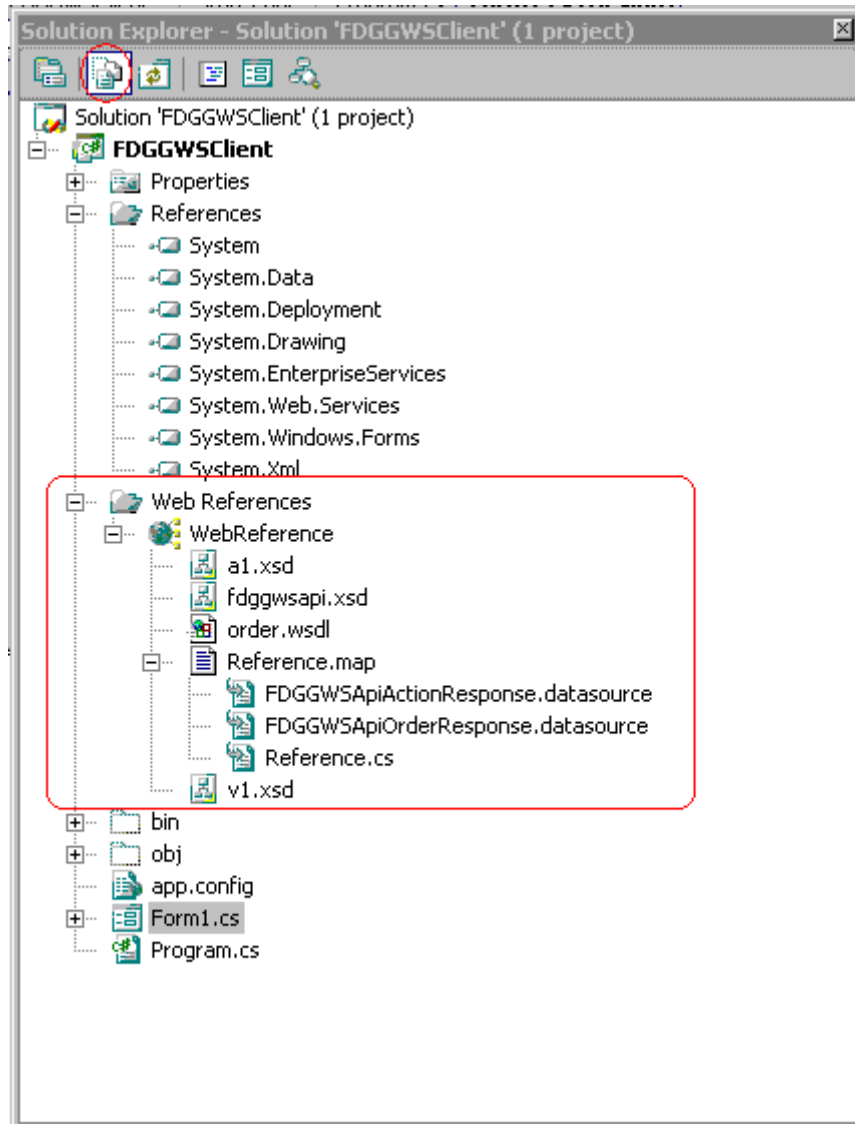
If you have saved the WSDL file at C:\FDGGWSClient\wsdl\order.wsdl, save the XSD
files in the path C:\FDGGWSClient\schemas_us\.

Click **Go**, next to the **URL** field.
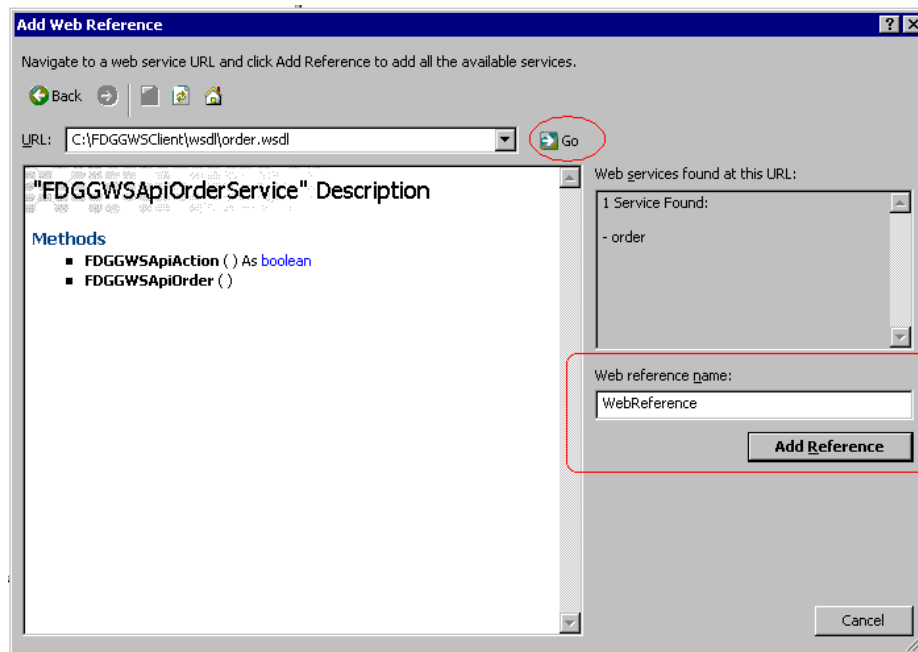
5. You can change the name of the web reference by editing the **Web reference name** field. Click **Add Reference**.

6. In the Project Solution Explorer, press the middle button (circled in the image below) to displays the files created:

7. Now you can create an instance of the client web service class in your code, using the following format:

```
FDGGWSApiOrderService oFDGGWSApiOrderService = new
FDGGWSApiOrderService();
```

15.3 Writing the .NET Client

The sample code below shows a C# .NET client.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```csharp
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using FDGGWSClient.FDGGWSRef;
using System.Security.Cryptography.X509Certificates;
using System.Net;

namespace FDGGWSClient
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
         ServicePointManager.Expect100Continue = false;
// Initialize Service Object          FDGGWSApiOrderService
oFDGGWSApiOrderService =
         new FDGGWSApiOrderService();
// Set the WSDL URL
         oFDGGWSApiOrderService.Url =
    @"
https://ws.firstdataglobalgateway.com/fdggwsapi/services/order.wsdl”,
// Configure Client Certificate
            oFDGGWSApiOrderService.ClientCertificates.Add
             (X509Certificate.CreateFromCertFile
             ("C:/FDGGWSClient/WS111901._.1.pem"));
// Set the Authentication Credentials
            NetworkCredential nc =
             new NetworkCredential("WS111901._.1", "JS2ND7Dc");
            oFDGGWSApiOrderService.Credentials = nc;
// Create Sale Transaction Request
            FDGGWSApiOrderRequest oOrderRequest =
             new FDGGWSApiOrderRequest();
            Transaction oTransaction = new Transaction();
            CreditCardTxType oCreditCardTxType = new CreditCardTxType();
            oCreditCardTxType.Type = CreditCardTxTypeType.sale;
            CreditCardData oCreditCardData = new CreditCardData();
             oCreditCardData.ItemsElementName =
             new ItemsChoiceType[] { ItemsChoiceType.CardNumber,
             ItemsChoiceType.ExpMonth, ItemsChoiceType.ExpYear };
            oCreditCardData.Items = new string[]
                { "4012000033330026", "12", "12" };
            oTransaction.Items = new object[]
                { oCreditCardTxType, oCreditCardData };
            Payment oPayment = new Payment();
            oPayment.ChargeTotal = 120;
            oTransaction.Payment = oPayment;
            oOrderRequest.Item = oTransaction;


    // Get the Response
             FDGGWSApiOrderResponse oReponse = null;
             try
```

```
        {
            oReponse =
            oFDGGWSApiOrderService.FDGGWSApiOrder(oOrderRequest);
            string sApprovalCode = oReponse.TransactionResult;
            this.textBox1.Text = oReponse.TransactionResult;


        }
        catch (System.Web.Services.Protocols.SoapException ex)
        {
            // Catch the Exception
        }
    }


    }
}
```

16 Using a Java Framework

First Data has tested the First Data Global Gateway Web Service API with the following Java frameworks:

- Axis Framework (version 2-1.5)
- Spring-WS (version 1.5.7)

The following sections discuss integrating with the Web Service API using these frameworks.

16.1 Axis Framework

The Axis Framework is a framework for building applications that create and process SOAP messages. This section discusses how to use the Axis Framework to connect with the First Data Global Gateway Web Service API.

The Axis Framework provides the WSDL2Java tool which creates stub code based on WSDL files.

16.1.1 Client Certificate Configuration

Before using the WSDL2Java tool, you must configure the tool to use the client certificate. To configure the WSDL2Java tool, open the wsdl2java.bat/wsdl2java.sh and add the following Java run-time optional parameter:

For wsdl2java.bat:

```
SET JAVA_OPTS=%JAVA_OPTS% -
Djavax.net.ssl.keyStore=<client_certificate_install_absolute_path>/WS<stor
e_id>._.1.ks
SET JAVA_OPTS=%JAVA_OPTS% -
Djavax.net.ssl.keyStorePassword=<keystore_password>
```

For wsdl2java.sh:

```
JAVA_OPTS="$JAVA_OPTS -
Djavax.net.ssl.keyStore=<client_certificate_install_absolute_path>/WS<stor
e_id>._.1.ks"
JAVA_OPTS="$JAVA_OPTS -
Djavax.net.ssl.keyStorePassword=<keystore_password>"
```

16.1.2 Generating Client Stubs

The WSDL2Java tool can be found in Axis' bin directory. To create the client stubs, enter the following command:

```
wsdl2java.bat -uri <WSDL URL> -S <destination folder for stub classes>
```

16.1.3 Writing the Axis Client

After generating the stubs, the next step is to write the client program that sends and receives the SOAP requests and responses. The following sample program makes an Order request for the Sale Transaction.

```
// all imports go here

public class FDGGWSAxisClient {

public static void main (String args[]) {

//the keystore file and password used on the next 2 lines are found within
the store CERT files archive
//which may be download via Virtual Terminal (under Support - Download
Center)
System.setProperty("javax.net.ssl.keyStore",
"<<PATH_TO_CLIENT_CERT_KEYSTORE_FILE>>");
System.setProperty("javax.net.ssl.keyStorePassword",
"<<KEYSTORE_PASSWORD>>");

HttpTransportProperties.Authenticator auth = new
HttpTransportProperties.Authenticator();
auth.setPreemptiveAuthentication(true);

//the following username and password values come from the WS<store
name>.__.1.auth.txt file
//which is found inside the store CERT files archive that may be download
via Virtual Terminal (under Support - Download Center)
auth.setUsername("WS<<STORE_NAME>>.__.1");
auth.setPassword("<<STORE_PASSWORD>>");

FDGGWSApiOrderServiceStub fdggWsStub = new FDGGWSApiOrderServiceStub();
Options options = fdggWsStub._getServiceClient().getOptions();
options.setProperty(HTTPConstants.AUTHENTICATE, auth);

Type_type1 sale_type = Type_type1.value6;

CreditCardTxType creditCardTxType = new CreditCardTxType();
creditCardTxType.setType(sale_type);

CardNumber_type1 cardNumber_type1 = new CardNumber_type1();
cardNumber_type1.setCardNumber_type0("4012000033330026");

ExpMonth_type1 expMonth_type1 = new ExpMonth_type1();
expMonth_type1.setExpMonth_type0("12");

ExpYear_type1 expYear_type1 = new ExpYear_type1();
expYear_type1.setExpYear_type0("12");

Card card = new Card();
card.setCardNumber(cardNumber_type1);
card.setExpMonth(expMonth_type1);
card.setExpYear(expYear_type1);
```

```
CreditCardDataSequence_type0 creditCardDataSequence_type0 = new
CreditCardDataSequence_type0();

creditCardDataSequence_type0.setCard(card);

CreditCardDataChoice_type0 dataChoice = new CreditCardDataChoice_type0();
dataChoice.setCreditCardDataSequence_type0(creditCardDataSequence_type0);

CreditCardData creditCardData = new CreditCardData();
creditCardData.setCreditCardDataChoice_type0(dataChoice);

BigDecimal bigDecimal = new BigDecimal("10");

ChargeTotal_type1 chargeTotal_type1 = new ChargeTotal_type1();
chargeTotal_type1.setChargeTotal_type0(bigDecimal);

Amount amount = new Amount();
amount.setChargeTotal(chargeTotal_type1);

Payment_type0 payment_type0 = new Payment_type0();
payment_type0.setAmount(amount);

Payment payment = new Payment();
payment.setPayment(payment_type0);

TransactionSequence_type0 transactionSequence_type0 = new
TransactionSequence_type0();

transactionSequence_type0.setCreditCardTxType(creditCardTxType);
transactionSequence_type0.setCreditCardData(creditCardData);

TransactionChoice_type0 transactionChoice_type0= new
TransactionChoice_type0();

transactionChoice_type0.setTransactionSequence_type0(transactionSequence_t
ype0);

Transaction t = new Transaction();
t.setTransactionChoice_type0(transactionChoice_type0);
t.setPayment(payment_type0);

FDGGWSApiOrderRequest fdggwsApiOrderRequest = new FDGGWSApiOrderRequest();

fdggwsApiOrderRequest.setTransaction(t);

FDGGWSApiOrderResponse response =
fdggwsstub.fDGGWSApiOrder(fdggwsApiOrderRequest);

System.out.println("The Transaction Result is
"+response.getFDGGWSApiOrderResponse().getTransactionResult());

System.out.println("The Order ID is
"+response.getFDGGWSApiOrderResponse().getOrderId());

log("The transaction result is " +response.getTransactionResult());
log("The order id is " +response.getOrderId());
```

```
}
}
```

17 SSL and HTTP Authentication

SSL

Your application must provide the client certificate for security.

The following code sample shows how to provide the client certificate.

```
// Needed for Client Certificate
System.setProperty("javax.net.ssl.keyStore",
"<<PATH_TO_THE_CLIENT_CERT_KEYSTORE FILE>>");
System.setProperty("javax.net.ssl.keyStorePassword",
"<<KEYSTORE_PASSWORD>>");
```

HTTP Authentication

The First Data Global Gateway Web Service API requires HTTP basic authorization on all calls to the web service.

The following code sample shows how to pass the user name and password for HTTP basic authorization.

```
Options options = ipgstub._getServiceClient().getOptions();
   HttpTransportProperties.Authenticator auth = new
HttpTransportProperties.Authenticator();
   auth.setPreemptiveAuthentication(true);
   auth.setUsername("WS111920._.1");
   auth.setPassword("0WRtTq1K");
   options.setProperty(HTTPConstants.AUTHENTICATE,auth);
```

17.1 Spring Web Services

Spring Web Services (Spring-WS) is designed for XML-based access to web services and supports the use of marshallers and unmarshallers. It enables your application to be coded solely using Java objects.

WebServiceTemplate is the core class for client-side web service access in Spring-WS. It contains methods for sending Source objects and receiving response messages as either Source or Result objects. Additionally, it can marshal objects to XML before sending them and unmarshal any response XML into an object again

17.2.1 Client Configuration

The following code sample shows the required configuration settings that go in the applicationContext.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns=http://www.springframework.org/schema/beans
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
```

```
<bean id="messagFactory"
class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory"/>

<bean id="abstractClient" abstract="true">
        <constructor-arg ref="messagFactory"/>
        <property name="destinationProvider">
             <bean class="org.springframework.ws.client.
                        support.destination.Wsdl11DestinationProvider">
              <property name="wsdl" value=

"https://ws.firstdataglobalgateway.com/services/order.wsdl"/>
           </bean>
        </property>
</bean>

<bean id="marshaller"
class="org.springframework.oxm.xmlbeans.XmlBeansMarshaller"/>

<bean id="httpClientParams"
class="org.apache.commons.httpclient.params.HttpClientParams">
      <property name="authenticationPreemptive" value="true"/>
<property name="connectionManagerClass"
value="org.apache.commons.httpclient.MultiThreadedHttpConnectionManager"/>
</bean>

<bean id="httpClient" class="org.apache.commons.httpclient.HttpClient">
      <constructor-arg ref="httpClientParams"/>
</bean>

<bean id="credentials"
class="org.apache.commons.httpclient.UsernamePasswordCredentials">
      <constructor-arg value="WS111901._.1"/>
      <constructor-arg value="qRAPL6FY"/>
</bean>

<bean id="messageSender"
class="org.springframework.ws.transport.http.CommonsHttpMessageSender">
      <constructor-arg ref="httpClient"></constructor-arg>
      <property name="credentials" ref="credentials"/>
</bean>

<bean id="fdggwsapiorder" parent="abstractClient"
class="com.firstdata.fdggwsapi.client.FDGGWSAPIOrder">
        <property name="marshaller" ref="marshaller"/>
        <property name="unmarshaller" ref="marshaller"/>
        <property name="messageSender" ref="messageSender"/>
</bean>

</beans>
```

The WebServiceTemplate class uses a URI as the message destination. The defaultUri property lets you specify the destination URI. Spring-WS creates a WebServiceMessageSender for the URI, which is responsible for sending the XML message. You can set one or more message

senders using the messageSender or messageSenders properties of the WebServiceTemplate class.

The following WebServiceMessageSender interfaces are available for sending messages via HTTP:

- HttpUrlConnectionMessageSender
- CommonsHttpMessageSender

The configuration sample above shows how to use CommonsHttpMessageSender to authenticate to the FDGG Web Service.

In addition to a message sender, the WebServiceTemplate requires a Web service message factory. The code in the following sections uses SaajSoapMessageFactory. This is the default used by Spring-WS, if a message factory is not specified via the messageFactory property.

17.2.2   Writing the Spring Client

WebServiceTemplate contains many convenience methods to send and receive web service messages. There are methods that accept and return a Source and those that return a Result. Additionally there are methods, which marshal and unmarshal objects to XML.

The preferred method of for creating messages and reading responses is to use the object/XML mapping provided by Spring-WS. The following three sections provide instructions for using object/XML mapping. If you must work directly with XML, see 0 17.6   Sending Direct XML Request on page 83 for instructions.

17.3   Configuring Object/XML Mapping

In order to facilitate the sending of plain Java objects, the WebServiceTemplate has a number of send methods that take an object as an argument. The marshalSendAndReceive method in the WebServiceTemplate class delegates the conversion of the request object to XML to a marshaller and the conversion of the response XML to an object to an unmarshaller. In order to use the marshalling functionality, you have to set values for the marshaller and unmarshaller properties of the WebServiceTemplate class. Spring provides support for the object/XML mapping through its org.springframework.oxm framework.

The following sample code shows how to set org.springframework.oxm.xmlbeans.XmlBeansMarshaller as the marshaller/unmarshaller in the applicationContext.xml file:

```
<bean id="marshaller"
class="org.springframework.oxm.xmlbeans.XmlBeansMarshaller"/>

<bean id="fdggwsapiorder" parent="abstractClient"
class="com.firstdata.fdggwsapi.client.FDGGWSAPIOrder">
        <property name="marshaller" ref="marshaller"/>
        <property name="unmarshaller" ref="marshaller"/>
</bean>
```

17.4   Generating XMLBean classes

Now you must generate Java objects based on the First Data Global Gateway Web Service API schema files. This allows you to work directly with Java objects when writing the client application.

To generate the Java objects, follow these steps:

- Download the following schema files and save them in a folder called **schemas_us**.

  https://ws.firstdataglobalgateway.com/fdggwsapi/schemas_us/fdggwsa pi.xsd
  https:// ws.firstdataglobalgateway.com /fdggwsapi/schemas_us/v1.xsd
  https://ws.firstdataglobalgateway.com/fdggwsapi/schemas_us/a1.xsd

- Provide the root schema as the parameter for the xmlbean ANT task as below.

```
<taskdef name="xmlbean" classname="org.apache.xmlbeans.impl.tool.XMLBean"
classpathref="classpath"/>
<xmlbean schema="fdggwsapi.xsd" srcgendir="${gen.dir}"
classgendir="${bin.dir}" classpathref="classpath" download="true"/>
```

- The root schema imports the other two schemas (v1 and a1) using relative URLs as shown in the code below. The directory structure on for your application needs to match the directory structure shown in the schema file.

```
<xs:import namespace=" http://secure.linkpt.net/fdggwsapi/schemas_us/v1"
schemaLocation="../schemas_us/v1.xsd" />

<xs:import namespace=" http://secure.linkpt.net/fdggwsapi/schemas_us/a1"
schemaLocation="../schemas_us/a1.xsd" />
```

To compile the schemas into XML beans, you need to download XMLBeans 2.2.0. See the following site for installation instructions:

http://xmlbeans.apache.org/documentation/conInstallGuide.html

You can generate the classes using one of the following tools:

- scomp
- XMLBean Ant task

To generate the classes using the XMLBeans scomp tool (located in the XMLBeans bin directory), enter the following command:

```
scomp –compiler <path to external java compiler> -src <target directory for
generated .java files> -d <target binary directory for .class and .xsb files>
<xsd>
```

If you use Ant in your build, you can use the the XMLBean Ant task instead of scomp. You need to download the xbean.jar from the XMLBeans developer kit at http://xmlbeans.apache.org/. The build script will need to include a taskdef for xmlbean. Add the following code to the build script to generate the classes for the schema:

```
<taskdef name="xmlbean" classname="org.apache.xmlbeans.impl.tool.XMLBean"
classpath="path/to/xbean.jar"/>
<xmlbean schema="<schema path>" srcgendir="<source generation directory>"
classgendir="<compiled class directory>" classpath="path/to/xbean.jar"/>
```

17.5   Writing the Client Program

The classes generated by XMLBeans allow your application work with Java objects instead of XML.

The following code sample shows how to send an order request using Spring-WS.

```java
// imports go here

public class FDGGWSAPIOrder extends WebServiceGatewaySupport {

    public FDGGWSAPIOrder(WebServiceMessageFactory messageFactory) {
            super(messageFactory);
        }

    public void ccSale(){

        try
        {
        // Instantiate the Order Request Document
    FDGGWSApiOrderRequestDocument orderRequestDoc =
FDGGWSApiOrderRequestDocument.Factory.newInstance();

        // Instantiate the Order Request
FDGGWSApiOrderRequest orderRequest =
orderRequestDoc.addNewFDGGWSApiOrderRequest();

        // Instantiate Transaction Object
            Transaction tran = orderRequest.addNewTransaction();

        // Create the Request
         CreditCardTxType ccTxType = tran.addNewCreditCardTxType();
         CreditCardTxType.Type.Enum sale = CreditCardTxType.Type.SALE;
         ccTxType.setType(sale);

         CreditCardData ccData = tran.addNewCreditCardData();
         ccData.setCardNumber("4012000033330026");
         ccData.setExpMonth("12");
         ccData.setExpYear("09");

         tran.setCreditCardTxType(ccTxType);
         tran.setCreditCardData(ccData);

         Payment pp = tran.addNewPayment();
         BigDecimal bd = new BigDecimal("31.23");
         pp.setChargeTotal(bd);

      // Add the Request to the Transaction
         tran.setCreditCardTxType(ccTxType);
         tran.setCreditCardData(ccData);
         tran.setPayment(pp);
```

```
        // Add the Transaction to the Order Request
           orderRequest.setTransaction(tran);

        // Add the Order Request to the Order Request document
           orderRequestDoc.setFDGGWSApiOrderRequest(orderRequest);

        // Send the Request and get the Response
    FDGGWSApiOrderResponseDocument orderResponseDoc
=(FDGGWSApiOrderResponseDocument)getWebServiceTemplate().marshalSendAndRec
eive(orderRequestDoc);

    FDGGWSApiOrderResponseDocument.FDGGWSApiOrderResponse response =
orderResponseDoc.getFDGGWSApiOrderResponse();

        // Get the Response Results
    System.out.println("The result of Sale Transaction is
"+response.getTransactionResult());

    System.out.println("The Order Id of Sale Transaction is
"+response.getOrderId());

System.out.println("The TDate of Sale Transaction is
"+response.getTDate());

    System.out.println("The Error Message is "+response.getErrorMessage());

          }
           // Handling the Exception
           catch (SoapFaultClientException e)
           {
               System.out.println("The Exception is "+e.toString());
               SoapFault sf = e.getSoapFault();
               if(sf != null) {
                   DOMSource s = (DOMSource)sf.getSource();
                   if(sf.getFaultDetail() != null){
                       Node detailNode = detailSource.getNode();
                       if(detailNode.getLocalName().
                             equalsIgnoreCase("detail")) {
    System.out.println("The Fault Detail is "+detailNode.getTextContent());

                       }
                   }
               }
           }
    }
public static void main(String[] args) {

    // SSL Configuration for Client Certs
System.setProperty("javax.net.ssl.keyStore", "/SSL/WS111901._.1.ks");

       System.setProperty("javax.net.ssl.keyStorePassword", "q6DbysArx1");

       // Get the Application Context configuration
ApplicationContext applicationContext = new
ClassPathXmlApplicationContext(
                 "com/firstdata/fdggwsapi/client/applicationContext.xml");
```

```
FDGGWSAPIOrder fdggwsapiOrder = (FDGGWSAPIOrder)
applicationContext.getBean("fdggwsapiorder", FDGGWSAPIOrder.class);

        // FDGGWSAPI Order Sale Request
        fdggwsapiOrder.ccSale();


}
}
```

17.6   Sending Direct XML Request

While object/XML mapping is the preferred method for using Spring-WS, if you must work
directly with the XML, that is also possible. The configuration discussed in the previous sections
for the applicationContext.xml file is not required.

The following code sample shows how to send an XML order request to the Web Service.

```
import java.io.IOException;
import javax.xml.transform.Source;

import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.core.io.Resource;
import
org.springframework.ws.client.core.support.WebServiceGatewaySupport;
import org.springframework.xml.transform.ResourceSource;
import org.springframework.xml.transform.StringResult;


public class SpringClient extends WebServiceGatewaySupport {
    private Resource request;
    public void setRequest(Resource request) {
        this.request = request;
    }
    public void fdggwsapi() throws IOException {
        Source requestSource = new ResourceSource(request);
        StringResult result = new StringResult();
        getWebServiceTemplate().sendSourceAndReceiveToResult
                             (requestSource, result);
        System.out.println(result);
    }

    public static void main(String[] args) throws IOException {

    // SSL Configuration for Client Certs
    System.setProperty("javax.net.ssl.keyStore", "/SSL/WS111901._.1.ks");
    System.setProperty("javax.net.ssl.keyStorePassword", "q6DbysArx1");

        // applicationContext.xml file contains the actual XML request.
            ApplicationContext applicationContext =
            new ClassPathXmlApplicationContext
                ("applicationContext.xml", SpringClient.class);
SpringClient springClient = (SpringClient)
applicationContext.getBean("springClient");
            springClient.fdggwsapi();
    }
```

```
}
```

The following code sample show the configuration required for the applicationContext.xml file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FDGGWSApiOrderRequest
xmlns=" http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
    <Transaction xmlns=
"http://secure.linkpt.net/fdggwsapi/schemas_us/v1">
        <CreditCardTxType>
            <StoreId>111901</StoreId>
            <Type>sale</Type>
        </CreditCardTxType>
        <CreditCardData>
            <CardNumber>4012000033330028</CardNumber>
            <ExpYear>12</ExpYear>
        </CreditCardData>
        <Payment>
            <ChargeTotal>120.222</ChargeTotal>
        </Payment>
    </Transaction>
</FDGGWSApiOrderRequest>
```

17.7   SSL/Certificate Configuration

Your application must provide the client certificate for security.

As the server certificate is issued by a well-known and trusted authority, which is already listed in the Trusted Store, you do not need to configure the server certificate.

The following code sample shows how to provide the keystore (.ks) file and password when calling the web service.

```
// SSL Configuration for Client Certs
System.setProperty("javax.net.ssl.keyStore", "/SSL/WS111901._.1.ks");
System.setProperty("javax.net.ssl.keyStorePassword", "q6DbysArx1");
```

18 Customer Test Environment (CTE)

The Customer Test Environment (CTE) allows your Development team to test applications and process transactions using the First Data Global Gateway Web Service API in a secure, no-cost environment. The CTE mimics the production environment. There is not a setup fee or processing charges when using the CTE.

To **APPLY** for a Test Account, access the following site, complete the form, and click **Submit**. You will receive a Welcome Email within 24 hours

- http://www.firstdata.com/gg/apply_test_account.htm

To test your integration to the First Data Global Gateway Web Service API, use these URLs listed below:

- https://ws.merchanttest.firstdataglobalgateway.com/fdggwsapi/services/order.wsdl

- https://ws.merchanttest.firstdataglobalgateway.com/fdggwsapi/services

- https://ws.merchanttest.firstdataglobalgateway.com/fdggwsapi/schemas_us/fdggwsapi.xsd

- https://ws.merchanttest.firstdataglobalgateway.com/fdggwsapi/schemas_us/v1.xsd

- https://ws.merchanttest.firstdataglobalgateway.com/fdggwsapi/schemas_us/a1.xsd

To assist you in testing the accuracy of your development efforts, a directory of CTE Response Codes (AVS, CVM, SGS Error Codes) can be found on the First Data Corp. website.

- http://www.firstdata.com/en_us/customer-center/merchants/support/first-data-global-gateway-api-software-landing#/content-product-1

To transition your test account to a production account replace the CTE URLs with these listed below:

- https://ws.firstdataglobalgateway.com/fdggwsapi/schemas_us/fdggwsapi.xsd>

- https://ws.firstdataglobalgateway.com/fdggwsapi/schemas_us/v1.xsd>

- https://ws.firstdataglobalgateway.com/schemas_us/a1.xsd>

19 Tax Calculator

The Tax Calculator module calculates the state and municipal sales tax.

To use the tax calculator module, create a fulltax line in your configuration file on the secure payment gateway. Next, send the fulltax line to Support in order to load it to the secure payment gateway.

The fulltax line provides information needed for the tax module to calculate sales tax for an order. The line includes entries for states where sales tax is charged. Entries are separated by a comma, which may be followed by a space.

Example:

```
fulltax: TX 8.25, AL 7.00, FL 7.00, UT mun
```

Most entries in the list consist of the two-digit code for the state, followed by a space and the tax rate charged for that state. See "U.S. State Codes" on page **Error! Bookmark not defined.** for state codes.

```
TX 8.25
```

If the tax includes municipal tax, the listing is the two-digit state code followed by mun.

```
UT mun
```

Municipal taxes are calculated according to the salestax.txt file on the secure payment gateway server. The salestax.txt file is updated monthly to ensure accuracy.

20 Shipping Calculator

With the shipping calculator, you can set rules for calculating shipping charges.

To use the shipping calculator module, you need to create a shipping and carrier file on the secure payment gateway server. When you create your shipping file, send it to Support along with your store number. The shipping calculator uses the shipping address and other information sent in the shipping entity along with the appropriate pricing data defined in the shipping file to calculate the charges.

The shipping file is a plain text file consisting of sets of code called zone type and zone definition lines. An example of how these lines might appear in a shipping file is shown below.

```
zone type line
zone definition line
zone definition line
zone type line
zone definition line
```

The fields within both types of lines go together to define the shipping charges. The zone type line describes the general shipping scheme, such as whether costs are based on item count, weight, or price.

The zone definition line gives specific parameters on pricing for each element in that pricing scheme. One or more zone definition lines must immediately follow each zone type line. Use zone definition lines to set shipping prices based on specific geographic areas or types of carriers to determine where price breaks occur. The fields within each line of code are separated by double-colons. For fields with multiple values, use commas (countries, states) or single colons (range definitions, prices).

Each zone type line is formatted with three fields:

- Tag Name
- Calculation Code
- Merchant-created range definitions

```
zone type::calculation method::range1:range2...
```

You can create as many zone type lines as you need for your business. You can use a separate zone type line for:

- Different shipping-cost calculations, such as the total weight or total cost of an order
- Separate freight or air transport carrier methods
- Division of the world shipping-zone prices

20.1 Creating Zone Type Lines

To create zone type lines:

1. Enter the following tag name. The zone type line must precede two colons:

```
zone type::
```

2. Determine how to charge customers for shipping your products and enter an applicable code number after the tag name followed by double colons with no spaces.

```
zone type::1::
zone type::3::
```

3. Create quantity ranges that share common pricing. Enter each range followed by a single colon or a comma.

```
zone type::1::1-3,4-5,6+
zone type::3::1-24,25-50,51+
```

20.2 Calculation Method

There are five choices for calculating the shipping charges. Select the applicable calculation methods for your business. Enter the code number after the Tag Name for each zone type line.

Method	Description
1	Charges based on the total number of items
2	Charges based on each item, then totaled
3	Charges based on the total weight of the order
4	Charges based on the weight of each item, then totaled
5	Charges based on the total price of the order

20.3 Assigning Ranges

A range is defined as a value or a set of values representing all items within a predetermined category, which use the same shipping charge. A range can be a single number, two numbers separated by a hyphen, or a number followed by a plus sign. You can specify an infinite number of ranges. The number of ranges in a zone type line must correlate exactly with the number of prices in the zone definition lines.

The following restrictions apply:

- Range definitions must be contiguous - you cannot skip numbers.
- Range definitions must start with the integer 1.
- The last range defined in each line must end with **+**.

A zone definition line specifies data that is required by the preceding zone type line of code. Several fields are specific to each business including the zone name, the shipping carrier code, and the shipping-cost codes for each range. See the example below.

```
zone name::country::carrier::range cost::range cost
```

20.4 Creating Zone Definition Lines

To create zone definition lines:

1. Enter a zone name for each shipping situation followed by two colons.

```
northamerica::
```

2. Select the applicable countries for your zone name followed by double colons. Use the two-digit country codes. See "Country Codes" on page **Error! Bookmark not defined.**.

```
northamerica::US,MX,CA::
```

For the U.S. only, enter each applicable two-letter state code after the country code, followed by two colons.

```
westcoast::US::CA,OR,WA,HI::
```

3. Determine the different shipping methods for your business. Enter one merchant-defined shipping carrier code only.

```
northamerica::US,MX,CA::1::
```

4. Determine the shipping cost for each range you specified in the zone type line. Enter the applicable shipping cost, followed by a colon or a comma.

```
zone type::1::1-3,4-5,6+
northamerica::US::MX::CA::1::25,40,75 NOTE:
```

Each shipping cost value in the zone definition line must match a range in the zone type line.

You determine the zone name for each zone definition line. Each name is an alphabetic string containing less than 20 letters and cannot include blank spaces.

If you offer different types of shipping, such as courier, overnight, two day, or ground transport, the zone definition line can list a shipping carrier option in the form of an integer. This will allow you to charge different amounts for premium shipping services.

The zone definition contains the actual charges for shipping items in the range specified by the preceding zone type. Merchants determine the charges for their products.

The following rules apply when you are creating zone definition code:

- If you are shipping internationally, the U.S. state code in a zone definition line is ignored.
- If shipping prices are the same for all U.S. states, you do not need to name the states individually.
- If you have a few exceptions for shipping, such as AK and HI, you can define a zone for them and include the remaining states in a non-specific U.S. zone.
- Any number of zone definition lines may follow a zone type line.
- The zone name and range charges must have values; all other fields can be blank.
- When the shipping calculator looks for a shipping file match, a blank field, such as carrier type, is treated as a match.

21 Troubleshooting

21.1 Merchant Exceptions

```
<detail>
   XML is not wellformed: Premature end of message.
</detail>
```

Explanation: You have sent an empty message. The message does not contain a SOAP message or any other text in the HTTP body.

```
<detail>
   XML is not wellformed: Content is not allowed in prolog.
</detail>
```

Explanation: The First Data Global Gateway API cannot interpret the content as XML.

```
<detail>
   XML is not wellformed:
   XML document structures must start and end within the same entity.
</detail>
```

Explanation: Your SOAP message is missing the end tag of the first open tag.

```
<detail>
   XML is not wellformed:
   The element type "SOAP-ENV:Body" must be terminated
   by the matching end-tag "&lt;/SOAP-ENV:Body&gt;".
</detail>
```

Explanation: An open internal tag (not the top level tag) is missing the end tag. In this example, the end tag </SOAP-ENV:Body> is missing.

```
<detail>
   XML is not wellformed:
   Element type "irgend" must be followed by either attribute
   specifications, "&gt;" or "/&gt;".
</detail>
```

Explanation: A tag is malformed. In this example, a ">" character is missing for the tag irgend.

```
<detail>
   XML is not wellformed:
   Open quote is expected for attribute "xmlns:ns3"
   associated with an element type "ns3:FDGGWSApiOrderRequest".
</detail>
```

Explanation: The value of one attribute is not enclosed in quotation marks. In the Web Service API, XML attributes are used only for the namespaces.

```
<detail>
   XML is not wellformed:
   The prefix "fdggwsapi" for element "fdggwsapi:FDGGWSApiOrderRequest"
   is not bound.
```

```
    </detail>
```

Explanation: The name space **fdggwsapi** is not declared. To declare a name space use the **xmlns** prefix. Add the following as an attribute to the FDGGWSApiOrderRequest or FDGGWSApiAction request element:

```
xmlns:fdggwsapi=
http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi
```

```
<detail>
    XML is not wellformed:
    The prefix "xmln" for attribute "xmln:ns2" associated
    with an element type "ns3:FDGGWSApiOrderRequest" is not bound.
</detail>
```

Explanation: You must use the pre-defined namespace **xmlns** to declare a custom namespace. In this example, the prefix is written as xmln and not as xmlns.

```
<detail>
    XML is not wellformed:
    Unable to create envelope from given source
    because the namespace was not recognized
</detail>
```

Explanation: The message could be interpreted as an XML message and the enclosing SOAP message is correct, but the included API message in the soap body has no name spaces or the name spaces are not declared correctly. The correct name spaces are described in the XSD.

```
<detail>
    XML is not wellformed:
    The processing instruction target matching "[xX][mM][lL]"
    is not allowed.
</detail>
```

Explanation: The SOAP body must not contain the XML declaration <?xml … ?>.

```
<detail>
    Unexpected characters before XML declaration
</detail>
```

Explanation: The XML must start with <?xml. Do not include an empty line or another white space character in front of the XML.

```
<detail>
    XML is not a SOAP message:
    Unable to create envelope from given source
    because the root element is not named "Envelope"
</detail>
```

Explanation: The XML appears to be valid but is not a SOAP message. Enclose your message in a SOAP envelope.

```
<detail>
    XML is not a valid SOAP message:
    Error with the determination of the type.
```

```
   Probably the envelope part is not correct.
</detail>
```

Explanation: The SOAP body tag is missing.

```
<detail>
   Source object passed to ''{0}'' has no contents.
</detail>
```

Explanation: The SOAP body is empty.

```
<detail>
   Included XML is not a valid FDGGWS API message:
   unsupported top level {namespace}tag "irgendwas" in the soap body. Only
one of [
{http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi}FDGGWSApiActionRe
quest,
   {http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi}FDGGWSApiOrder
Request
] allowed.
</detail>
```

Explanation: The first tag in the Web Service API message contained in the SOAP body must be either FDGGWSApiActionRequest or FDGGWSApiOrderRequest. In this case, the tag has no namespace.

```
<detail>
   Included XML is not a valid FDGGWS API message:
unsupported top level {namespace}tag
"{http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi}FDGGWSApiOrderRe
quest" in the soap body. Only one of [
{http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi}FDGGWSApiActionRe
quest,
{http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi}FDGGWSApiOrderReq
uest
] allowed.
</detail>
```

Explanation: The first tag in the including Web Service API message contained in the SOAP body must be either FDGGWSApiActionRequest or FDGGWSApiOrderRequest. In this case, the namespace is wrong.

```
<detail>
   cvc-pattern-valid:
   Value '1.234' is not facet-valid with respect to pattern
'([1-9]([0-9]{0,12}))?[0-9](\.[0-9]{1,2})?' for type
'#AnonType_ChargeTotalAmount'
   cvc-type.3.1.3:
   The value '1.234' of element 'ns3:ChargeTotal' is not valid.
</detail>
```

Explanation: The value of a tag does not correspond with the declaration in the XSD. The value has three decimal places but the XSD only allows two.

```
<detail>
```

```
   cvc-complex-type.2.4.a:
   Invalid content was found starting with element 'ns2:ExpYear'.
   One of '{"http://secure.linkpt.net/fdggwsapi/schemas_us/v1":ExpMonth}'
   is expected.
</detail>
```

Explanation: The tags must be included in the order declared in the XSD. In this case the tag ExpMonth is expected and not ExpYear.

```
<fdggwsapi:FDGGWSApiOrderResponse Xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <fdggwsapi:CommercialServiceProvider/>
   <fdggwsapi:TransactionTime>
      Tue Nov 03 13:34:02 2009
   </fdggwsapi:TransactionTime>
   <fdggwsapi:TransactionID/>
   <fdggwsapi:ProcessorReferenceNumber/>
   <fdggwsapi:ProcessorResponseMessage/>
   <fdggwsapi:ErrorMessage>
      SGS-005002: The merchant is not setup to support the requested
service.
   </fdggwsapi:ErrorMessage>
   <fdggwsapi:OrderId>
      A-bf98ecb3-c3f7-44e2-97cf-5c965ca84f93
   </fdggwsapi:OrderId>
   <fdggwsapi:ApprovalCode/>
   <fdggwsapi:AVSResponse/>
   <fdggwsapi:TDate/>
   <fdggwsapi:TransactionResult>
      DECLINED
   </fdggwsapi:TransactionResult>
   <fdggwsapi:ProcessorResponseCode/>
   <fdggwsapi:ProcessorApprovalCode/>
   <fdggwsapi:CalculatedTax/>
   <fdggwsapi:CalculatedShipping/>
   <fdggwsapi:TransactionScore/>
   <fdggwsapi:AuthenticationResponseCode/>
</fdggwsapi:FDGGWSApiOrderResponse>
```

Explanation: The type of transaction submitted is not allowed for this merchant. If you receive this result for a transaction type, which is included in your agreement, please contact our technical support team.

```
<fdggwsapi:FDGGWSApiOrderResponse xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
   <fdggwsapi:CommercialServiceProvider/>
   <fdggwsapi:TransactionTime>
      Tue Nov 03 17:10:51 2009
   </fdggwsapi:TransactionTime>
   <fdggwsapi:TransactionID/>
   <fdggwsapi:ProcessorReferenceNumber/>
   <fdggwsapi:ProcessorResponseMessage/>
   <fdggwsapi:ErrorMessage>
      SGS-005005: Duplicate transaction.
```

```
      </fdggwsapi:ErrorMessage>
      <fdggwsapi:OrderId>
         A-e981664e-546f-4db9-895b-6633ee163f69
      </fdggwsapi:OrderId>
      <fdggwsapi:ApprovalCode/>
      <fdggwsapi:AVSResponse/>
      <fdggwsapi:TDate/>
      <fdggwsapi:TransactionResult>FRAUD</fdggwsapi:TransactionResult>
      <fdggwsapi:ProcessorResponseCode/>
      <fdggwsapi:ProcessorApprovalCode/>
      <fdggwsapi:CalculatedTax/>
      <fdggwsapi:CalculatedShipping/>
      <fdggwsapi:TransactionScore/>
      <fdggwsapi:AuthenticationResponseCode/>
</fdggwsapi:FDGGWSApiOrderResponse>
```

Explanation: This transaction is a duplicate transaction. Transactions with the same data submitted within a configurable amount of time are rejected as duplicate transactions.

```
<fdggwsapi:FDGGWSApiOrderResponse xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
      <fdggwsapi:CommercialServiceProvider/>
      <fdggwsapi:TransactionTime>
         Tue Nov 03 14:07:13 2009
      </fdggwsapi:TransactionTime>
      <fdggwsapi:TransactionID/>
      <fdggwsapi:ProcessorReferenceNumber/>
      <fdggwsapi:ProcessorResponseMessage/>
      <fdggwsapi:ErrorMessage>
         SGS-002311: Internal Error.
      </fdggwsapi:ErrorMessage>
      <fdggwsapi:OrderId>
         A-8a07eaad-26d7-4233-b13a-8a102287f6c8
      </fdggwsapi:OrderId>
      <fdggwsapi:ApprovalCode/><fdggwsapi:AVSResponse/>
      <fdggwsapi:TDate>1257286033</fdggwsapi:TDate>
      <fdggwsapi:TransactionResult>DECLINED</fdggwsapi:TransactionResult>
      <fdggwsapi:ProcessorResponseCode/>
      <fdggwsapi:ProcessorApprovalCode/>
      <fdggwsapi:CalculatedTax/>
      <fdggwsapi:CalculatedShipping/>
      <fdggwsapi:TransactionScore/>
      <fdggwsapi:AuthenticationResponseCode/>
</fdggwsapi:FDGGWSApiOrderResponse>
```

Explanation: The SOAP Request XML might be incorrect. Check for the correct namespaces for the tags.

```
<fdggwsapi:FDGGWSApiOrderResponse xmlns:fdggwsapi=
"http://secure.linkpt.net/fdggwsapi/schemas_us/fdggwsapi">
      <fdggwsapi:CommercialServiceProvider/>
      <fdggwsapi:TransactionTime>
         Tue Nov 03 17:10:51 2009
      </fdggwsapi:TransactionTime>
      <fdggwsapi:TransactionID/>
```

```
    <fdggwsapi:ProcessorReferenceNumber/>
    <fdggwsapi:ProcessorResponseMessage/>
    <fdggwsapi:ErrorMessage>
        SGS-005999: There was an unknown error in the database.
    </fdggwsapi:ErrorMessage>
    <fdggwsapi:OrderId>
        A-e981664e-546f-4db9-895b-6633ee163f69
    </fdggwsapi:OrderId>
    <fdggwsapi:ApprovalCode/>
    <fdggwsapi:AVSResponse/>
    <fdggwsapi:TDate/>
    <fdggwsapi:TransactionResult>DECLINED</fdggwsapi:TransactionResult>
    <fdggwsapi:ProcessorResponseCode/>
    <fdggwsapi:ProcessorApprovalCode/>
    <fdggwsapi:CalculatedTax/>
    <fdggwsapi:CalculatedShipping/>
    <fdggwsapi:TransactionScore/>
    <fdggwsapi:AuthenticationResponseCode/>
</fdggwsapi:FDGGWSApiOrderResponse>
```

Explanation: You may have tried to void a credit card transaction as a different payment type.

21.2 cURL Login Error Messages

```
* unable to set private key file: 'C:\API\config\WS120666668._.1.key' type
PEM
* Closing connection #0
curl: (58) unable to set private key file: 'C:\API\config\
WS120666668._.1.key' type PEM
```

Explanation: Your keystore and password do not match. Ensure that you used the right keystore and password. Check that you used the WS<storeId>._.1.pem file. You can append .cer to the file name so that you can open the certificate with a double click. The certificate must be exposed for your store. Remove the extension .cer after the check.

```
<html>
    <head>
        <title>Apache Tomcat/5.5.20 - Error report</title>
        <style>
        ...
        </style>
    </head>
    <body>
        <h1>HTTP Status 401 -</h1>
        <HR size="1" noshade="noshade">
        <p>
            <b>type</b>
            Status report
        </p>
        <p>
            <b>message</b>
            <u></u>
        </p>
        <p>
            <b>description</b>
```

```
        <u>This request requires HTTP authentication ().</u>
    </p>
    <HR size="1" noshade="noshade">
    <h3>Apache Tomcat/5.5.20</h3>
  </body>
</html>
```

Explanation: Your user ID and/or password are incorrect. The First Data Global Gateway Web Service API accepted your certificates.

21.3 Java Client Login Error Messages

```
java.io.IOException: Keystore was tampered with, or password was incorrect
```

Explanation: Your keystore password does not match. You can check the password with the keytool of the JDK. Password is case sensitive. Run the following command:

```
keytool -list -v -keystore <absolute path of your WS{store_id}._.1.ks
keystore> -storepass <your keystore password>

javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: No trusted certificate found
```

Check the cacerts file is available under {JAVA_HOME}/jre/lib/security folder

22 Installing the Client Certificate

The following instructions assume you are running ASP on Microsoft IIS 5.1 on Windows XP. To install the client certificate, follow these steps:

1. Select **Run** from the **Start** menu. Enter **mmc** in the Run dialog and click **OK**.
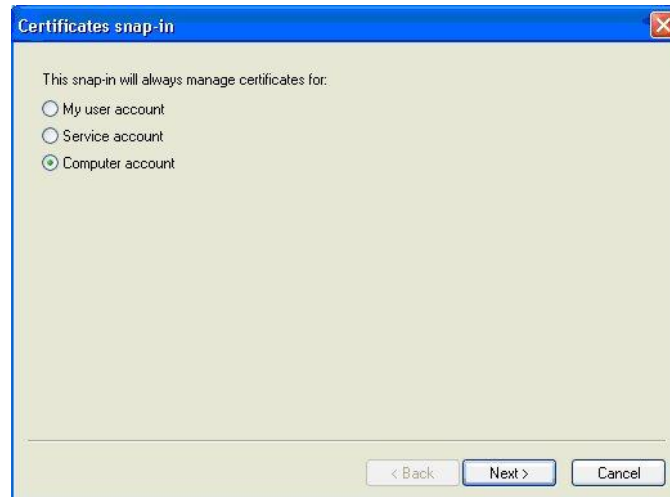2. From the **File** menu, select **Add/Remove Snap-In**.
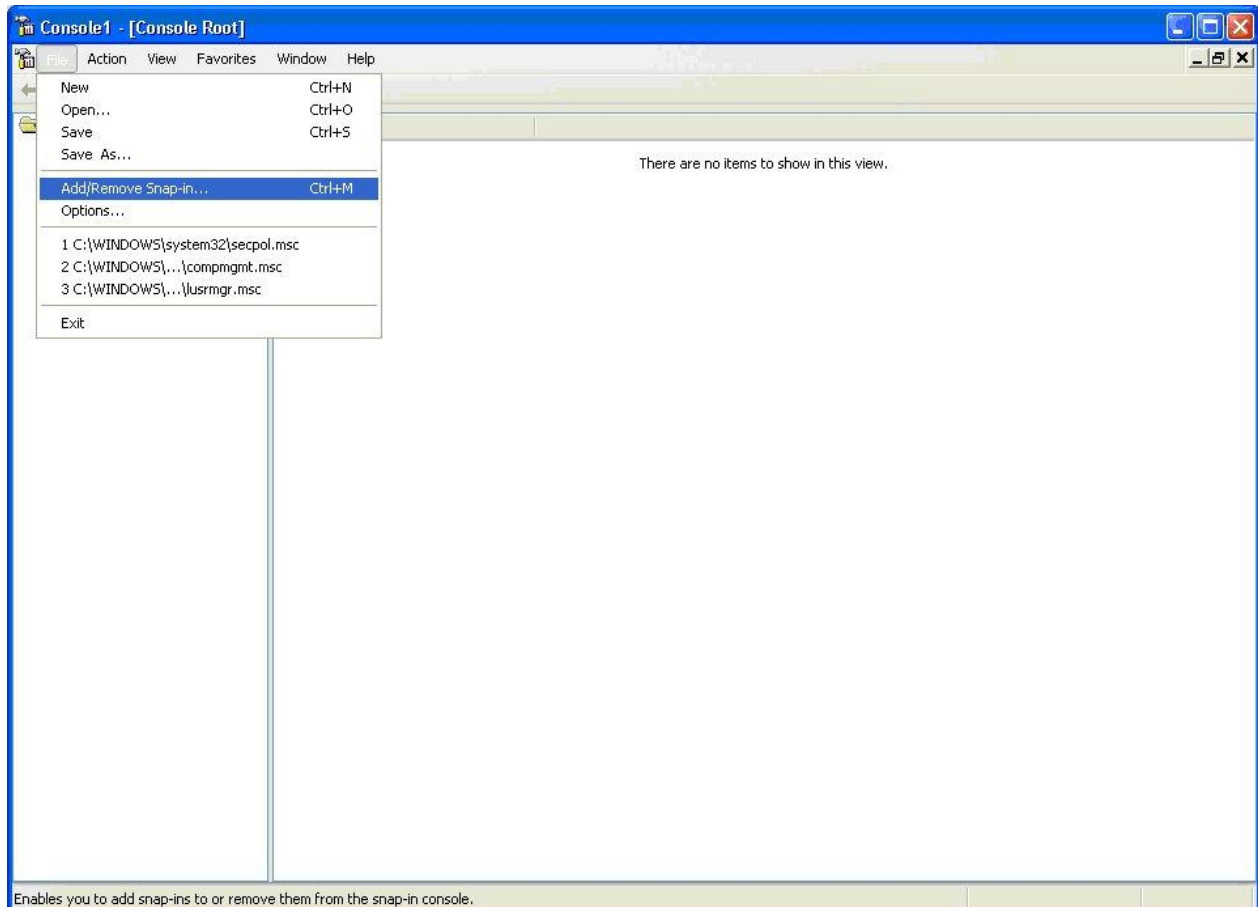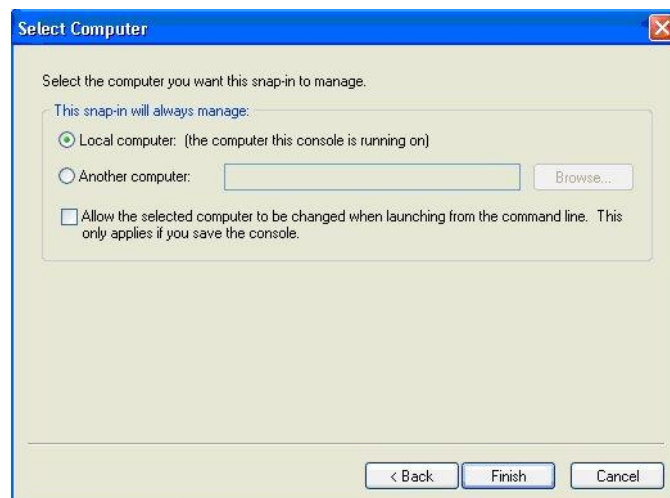

3. Click **Add**.

4. Under **Snap-In**, select **Certificates** and click **Add**.
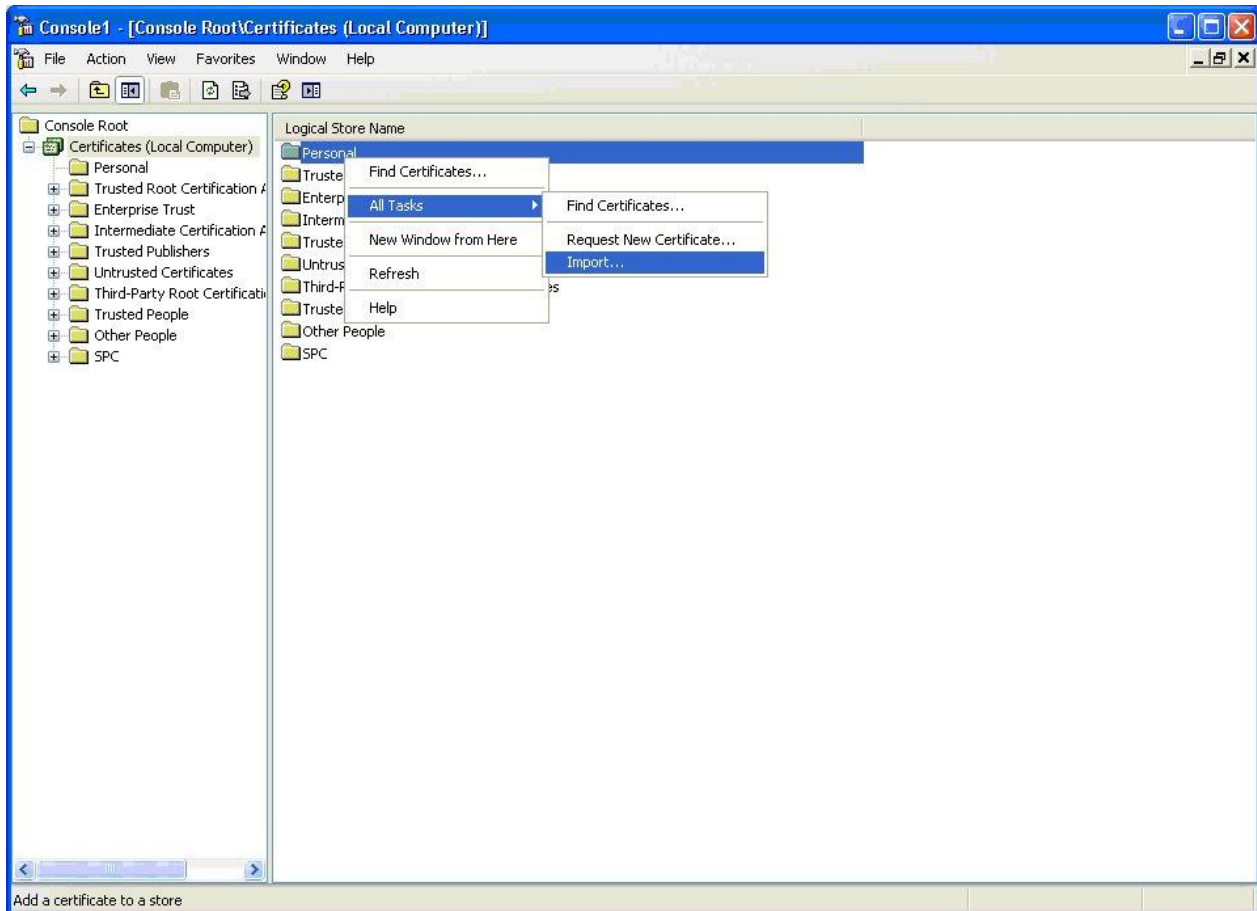

5. Select the account for which you want to manage the certificates. Since IIS uses the computer account, choose **Computer Account** and click **Next**.

6. Choose **Local Computer** and click **Finish**.
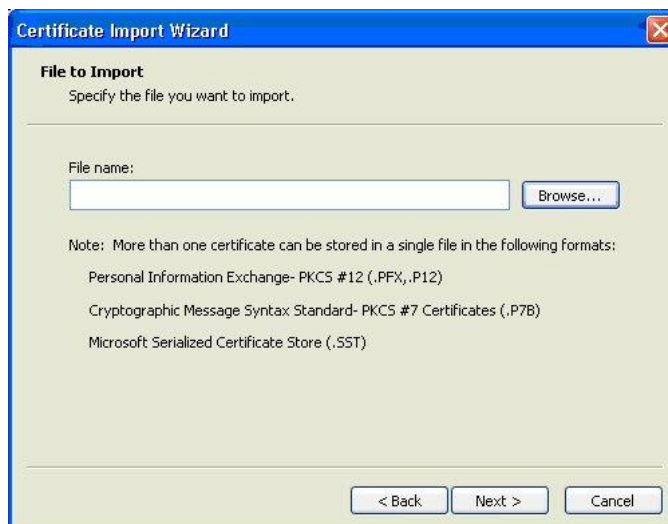

7. Click **Close** and then **OK**.
8. Expand the **Certificates (Local Computer)** tree. The client certificate will be installed in the **Personal** folder.
9. Right click the **Certificates** folder, select **All Tasks**, and click **Import**. The Certificate Import Wizard displays.

10. Click **Next**.

11. Choose your client certificate p12 file and click **Next**.

Certificate Import Wizard

File to Import
Specify the file you want to import.

File name:

[   ] [Browse...]

Note: More than one certificate can be stored in a single file in the following formats:

Personal Information Exchange- PKCS #12 (.PFX,.P12)

Cryptographic Message Syntax Standard- PKCS #7 Certificates (.P7B)

Microsoft Serialized Certificate Store (.SST)

[< Back] [Next >] [Cancel]

12. Enter the client certificate installation password and click **Next**.

Certificate Import Wizard

Password
To maintain security, the private key was protected with a password.

Type the password for the private key.

Password:

[********]

☐ Enable strong private key protection. You will be prompted every time the private key is used by an application if you enable this option.

☐ Mark this key as exportable. This will allow you to back up or transport your keys at a later time.

[< Back] [Next >] [Cancel]

13. Select **Place all certificates in the following store** and browse for the **Personal** folder if not yet displayed. Click **Next**.
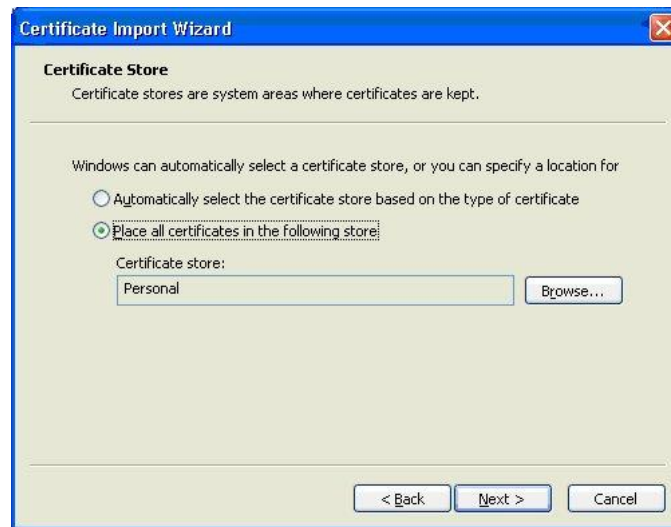
14. Check the displayed settings and click **Finish**. Your client certificate is now installed in the local computer's personal certificates store. Now, IIS (running ASP) can find the client certificate when communicating with another server via HTTP.

Next-

- Grant the IIS user access to the client certificate private key. To do so, first download the WinHttpCertCfg tool from Microsoft. Use the following URL:

    http://www.microsoft.com/downloads/details.aspx?familyid=c42e27ac-3409-40e9-8667-c748e422833f&displaylang=en

- Grant access to the IIS user, using the command line, navigate to the directory where you installed WinHttpCertCfg and enter the following command:

    winhttpcertcfg -g -c LOCAL_MACHINE\My -s WS101._.1 -a IWAM_MyMachine

`WS101._.1` is the name of the client certificate.

- Replace this value with the name of your client certificate. The name should be in the format WS<store_ID>._.1.

- Verify this value when you install the client certificate using the instructions above.

`IWAM_MyMachine` is the IIS user name. IIS 5.1 uses `IWAM_MachineName` by default. Replace MachineName with the name of your machine. For example, if your machine has the name IISServerMachine, the IIS user will be called `IWAM_IISServerMachine`. Other IIS versions might use a different naming scheme. If you do not know your machine name or IIS user name, check the IIS documentation and contact your administrator.

23. Glossary

Account Number
The account number for a checking or savings account is a unique number that identifies the customer's account. The account number appears on the check next to the transit routing number. The numbers are usually separated by a non-alphabetic, non-numeric symbol.

ACH
ACH is an abbreviation for Automated Clearing House. Automated Clearing House (ACH) is the name of an electronic network for financial transactions in the United States. ACH processes large volumes of both credit and debit transactions which are originated in batches. ACH allows merchants to accept payments from a customer's checking or savings account.

Acquiring Bank
An acquiring bank is a bank, which provides a service to its business customers allowing them to accept card payments for goods and services.

Address Verification System
The Address Verification System (AVS) is a system that checks the billing address of the credit card provided by the user with the address on file at the credit card company. The gateway provides an AVS code in each approved transaction result that tells you how well the two addresses match. If they match, there is a lower probability of fraud if there is a discrepancy in either the address or zip code; the probability of fraud is higher. Merchants can use AVS codes to help protect themselves from chargebacks and fraud.

Antivirus Software
Antivirus software consists of computer programs that attempt to identify, deter, and eliminate computer viruses and other malicious software. Antivirus software typically uses two different techniques to accomplish this: Examining (scanning) files to look for known viruses matching definitions in a virus dictionary. Identifying suspicious behavior from any computer program, which might indicate infection. Such analysis may include data captures, port monitoring, and other methods. Because of the risk of computer viruses doing harm to your computer files, antivirus software is recommended for all Internet users.

Application Programming Interface (API)
First Data Global Gateway API is a tool that allows a merchant to create a customer commerce solution. Our Application Programming Interface (API) allows you to add payment functionality to custom built web sites or online applications.

Authentication Response Code
A three (3) digit numeric response code returned on 3-D Secure enabled transactions. This field only returns a value for merchants who use this add-on service.

Authorization
An authorization does not charge the card until you perform a Ticket Only transaction or confirm shipment of the order. The period during which funds are reserved may be as little as three days or as long as several months. An authorization reserves funds on a customer's credit card.

Authorize Only (PreAuth)
An Authorize Only transaction reserves funds on a customer's credit card. An Authorize Only transaction does not charge the card until you perform a Ticket Only transaction and confirm shipment of the order using an option available in the Reports section. Authorize-only transactions reserve funds for varying periods, depending on the issuing credit card company's policy. The period may be as little as three days or as long as several months. For your protection, you should confirm shipment as soon as possible after authorization.

Batch
A group of credit card or check transactions that are submitted together to the payment gateway for settlement. On the Global Gateway, batches are submitted automatically once a day.

Blocking and Limiting
If you suspect certain transactions might be fraudulent, you can block further purchases by blocking credit card numbers, persons' names, domain names, and IP addresses or Class C addresses from purchasing at your store. You can limit the amount that any customer can spend at your store by setting a maximum purchase amount. You can set how long automatic lockouts and duplicate lockouts will continue to be blocked.

Browser
Short for web browser, a browser is a software application that enables a user to display and interact with text, images, videos, music, and other information typically located on a web page at a web site on the Internet.

Cable Modem
A cable modem is a type of modem that provides access to the Internet through the cable television infrastructure. Cable modems are primarily used to deliver broadband Internet access, taking advantage of unused bandwidth on a cable television network. If the cable network is shared with many other Internet subscribers, Internet access speed may go down.

Card Code
The card code is the card security code, sometimes called Card Verification Value or Code (CVV or CVC). It is a security feature for credit or debit card transactions, giving increased protection against credit card fraud This code (also known as a CCID or Credit Card ID) is often asked for by merchants to secure transactions when the card is not present, usually occurring over the Internet, by mail, fax, or over the phone. The payment gateway will compare the card code with the code on file at the card-issuing bank. Results of this comparison will show in the transaction approval code. Using the card code results in conjunction with the Address Verification Service (AVS), you can make better-informed decisions about whether to accept transactions. MasterCard, Visa, and Discover credit and debit cards have a three-digit code, called the "CVC2" (card validation code), "CVV2" (card verification value), and "CID" (card identification number), respectively. It is always the final group of numbers printed on the back signature

panel of the card. New North American MasterCard and Visa cards feature the card code in a separate panel to the right of the signature strip. American Express cards have a four-digit code printed on the front side of the card above the number, referred to as the CID.

Card-Issuing Bank
A financial institution or bank that issues a credit, debit, or purchasing card to a business or an individual. The card-issuing bank has an address on file for the card, which the Address Verification System (AVS) compares to the address given to the merchant.

Chargeback
A chargeback is a forced refund to the customer through your bank account. Chargebacks can occur with any type of business whether it is online or at an actual store location. Each fraudulent credit card transaction typically results in a chargeback. Credit card associations penalize merchant banks for chargebacks. Naturally, the bank passes the fines on to the responsible merchant, and these penalties can be severe. While consumers are provided with a certain degree of protection if their credit card numbers are stolen and misused, Internet merchants are fully liable for all transactions because Internet transactions are classified as "card-not-present."

Check Number
The check number is a number unique to each check. The check number is always found in the top right corner of the check. The check number is only provided as a reference to process the ACH transaction.

Commerce Service Provider (CSP)
The commerce service provider (CSP) supplies businesses with the tools and services they need to buy and sell products and services over the Internet, and to manage their online enterprises. CSPs can generally host a secure web site that could be connected to a secure payment gateway for selling products or services over the Internet.

Credit
A Credit transaction returns funds to a customer's credit card on orders without an order number. This transaction is intended for returns against orders processed outside the system. Credit transactions are marked as Returns in your reports.

Credit Card
A credit card is a card (usually plastic) that assures a seller that the person using it has a satisfactory credit rating, and that the issuer will see to it that, the seller receives payment for the merchandise delivered.

CVC2
The CVC2 is the card validation code or card code for MasterCard. See the definition for card codes for more information.

CVV2
The CVC2 is the card verification value or card code for Visa cards. See the definition for card codes for more information.

Data Field
A data field is an area on a web form or software application where you can enter information relevant to the name of the field. For example, you would enter the zip code in the data field named zip code.

DDA Number
The DDA (demand deposit account) number is the deposit account held at a bank or other financial institution for the purpose of securely and quickly providing frequent access to funds on demand.

Dial-Up Connection
A dial-up connection is a way to access the Internet through a telephone line. A modem is connected to a computer and a telephone line to dial into an Internet service provider's (ISP) node to establish a modem-to-modem link, which is then routed to the Internet. The speed of dial up connections is usually slower than other Internet access options.

Digital Certificate
A digital certificate is an electronic certificate that establishes the merchant's credentials for performing business on the Internet. It is an encrypted set of information issued by an Internet certification authority such as Thawte. Digital certificates are required for merchants who choose to use the API. For other products, the merchant does not need a digital certificate.

Domain Name
A name that identifies a computer or computers on the internet. These names appear as a component of a web site's URL, such as microsoft.com. This type of domain name is also called a hostname.

DSL
DSL (Digital Subscriber Line) is a technology for bringing fast Internet service to homes and small businesses over the wires of a local telephone network.

E-commerce (ECI)
E-commerce (ECI) or electronic commerce consists of the buying and selling of products or services over electronic systems, such as the Internet and other computer networks.

Electronic Check Acceptance (ECA)
With electronic check acceptance (ECA), the check is electronically submitted as a check. The check is no longer usable and the paper check must be voided. The customer signs and receives a paper receipt. ECA services may include a check guarantee service. ECA is used for retail payments only.

Field
A field is an area on a web form or software application where you can enter information relevant to the name of the field. For example, you would enter the zip code in the field named zip code.

Firewall
A firewall is a hardware or software device, which is configured to permit, deny, or proxy data through a computer network which has different levels of trust. A firewall protects the resources of a private network from users of other networks.

First Data Global Gateway Connect
The First Data Global Gateway Connect service is an e-Commerce solution using a hosted payment page. This eliminates some of the complexity and is great for a merchant with limited resources or expertise.

Forced Ticket
A Forced Ticket transaction is a credit card transaction for authorizations you obtained over the phone. It requires a reference number (or approval code) that you should have received when you made the phone authorization.

Hierarchy
A term used to describe the organizational tree structure for multi-store reports. Merchants describe their organization by defining an org chart in the form of a tree structure. The structure is used for combining store reports into groups at different levels. The term hierarchy refers to the entire organizational tree structure containing levels and elements.

HTML
HTML is short for HyperText Markup Language. HTML is a markup language used to structure text and multimedia documents and to set up hypertext links between documents used extensively on the Internet. Other than manually entering transactions using the virtual POS terminal, HTML is the simplest way to send payment transactions to the payment gateway.

HTTP
HTTP (Hypertext Transfer Protocol) is a communications protocol used to transfer or convey information on the Internet. For example, when you enter a URL in your browser, it sends an HTTP command to the web server directing it to receive and transmit the requested web page.

Hyperlink
A hyperlink is a reference or navigation element in a document or web page linking to another section of the same document or web page or to another document or web page that may be on a different web site.

Internet Check Acceptance (ICA)
Internet Check Acceptance (ICA) is the type of check service provided on the payment gateway. ICA uses the Automated Clearing House (ACH) to transfer funds from the customer's account. The account information is entered in an online payment form, and no check is used. The customer may or may not sign a payment form. In either case, the merchant needs a documented record of the customer's authorization to transfer funds from the account. ICA includes an electronic receipt. There is no check guarantee service with ICA. ICA is typically used for mail orders/telephone orders (MO/TO) or e-commerce transactions, but may also be used for retail.

Internet Service Provider (ISP)
An Internet service provider (ISP) is a business or organization that provides consumers or businesses access to the Internet and related services. An ISP can also host a web site.

IP Address
IP address is short for Internet Protocol address. An IP address is a number that is used to identify a specific computer on a network or on the Internet. The format of an IP address is

written as four numbers separated by periods. Each number can be from zero (0) to 255. For example, 1.160.10.240 could be an IP address.

Issuing Bank
The financial institution or bank that issues a credit, debit, or purchasing card to a business or consumer. The issuing bank has an address on file for the card, which the Address Verification System (AVS) compares to the address given to the merchant.

Level
A level is a single tier in the hierarchy or organizational tree structure for multi-store users. The top level (1) is typically the root (or corporate) level containing one (1) element. The lowest level of the tree is always the User level; the next level up from the lowest is the Store level. Merchants define the number of levels and names of each level for their own organization up to 10 total levels.

Local Area Network (LAN)
A local area network (LAN) is a computer network covering a small geographic area, like a home, office, or group of buildings. The defining characteristics of LANs, in contrast to Wide Area Networks (WANs), include their much higher data transfer rates, smaller geographic range, and lack of a need for leased telecommunication lines.

Log In
To log in is the process by which individual access to a computer system is controlled by identification of the user in order to obtain credentials to permit access. It is an integral part of computer security. A user can log in to a system to obtain access, and then log out when the access is no longer needed.

Log Off
To log off (aka to log out, sign out, or sign off) is to close off one's access to a computer system after previously having logged in. To log out of the system, click the Logout link in the top right corner of the application. To prevent unauthorized users from accessing their account, merchants should always log off and close the browser window when they are finished using the system.

Multi-Store
Multi-stores are multiple accounts with different store numbers.

Network
A network is a group of two or more computer systems linked together.

OrderID
A number assigned to a transaction. You can manually enter a number in this field to assign a proprietary number to the transaction. All OrderID numbers are unique. If you do not enter a number in this field, the FDGG Virtual Terminal automatically generates an OrderID number. The First Data Global Gateway Web Service API only accepts ASCII characters. The OrderID field cannot contain the following characters: &, %, or /, or exceed 100 characters in length.

Password
A password is a form of secret authentication data that is used to control access to a resource. It is recommend that users change their password frequently and do not share it with anyone to prevent unauthorized access to their accounts.

Payment Gateway
A payment gateway is an e-commerce application service that authorizes payments for e-businesses and online retailers. It is the equivalent of a physical POS (Point-of-sale) terminal located in most retail outlets. Payment gateways encrypt sensitive information, such as credit card numbers, to ensure that information passes securely between the customer and the merchant.

PDF File
PDF is short for Portable Document Format. It is the file format created by Adobe Systems in 1993 for document exchange. PDF is used for representing two-dimensional documents in a device-independent and display resolution-independent fixed-layout document format. Internet users need an Adobe Acrobat viewer to open a PDF file, which can be downloaded free at http://www.adobe.com.

Periodic Billing (Recurring Billing)
Periodic billing is recurring payments or the capability to charge customers on a recurring basis according to merchant-defined rules. Gateway products allow a merchant to charge a customer's card in exchange for products and services one or more times every day, week, month, or year.

Plug-In
A plug-in is a hardware or software module that adds a specific feature or service to a larger system. For example, a number of plug-ins for the Mozilla Firefox browser, enable it to display different types of audio or video files.

Point of Sale (POS)
Point of Sale (POS) - purchasing a product from the merchant and the merchant is processing the payment transaction. POS is commonly used to refer to the payment terminals or software merchants use to process the payment transaction.

Protocol
A Protocol is a set of guidelines or rules that help in governing an operation on the Internet and communications over it. There are several different protocols. HTTP is the protocol used for the Internet.

Purchasing Card
A purchasing card is a corporate card used by some companies for their business purchases. When a customer pays for goods or services using a purchasing card, the following information must be included with the order information. This information is optional for a regular credit card transaction: An indication of whether the order is tax exempt. The amount of tax applied to the order. If the order is tax exempt, the tax amount should be zero. A purchase order number associated with the order. One purchase order can apply to several individual orders to allow for delivery of goods over time. If there is not a purchase order associated with the order, the customer must supply some value for the order.

Recurring Payment (Periodic Billing)
The capability to charge customers on a recurring basis according to merchant-defined rules. Gateway products allow a merchant to charge a customer's card in exchange for products and services one or more times every day, week, month, or year.

Return
A Return transaction returns funds to a customer's credit card for an existing order on the system. To perform a return, you need the order number (which you can find in your reports). After you perform a Return for the full order amount, the order will appear in your reports with a transaction amount of 0.00.

Sale
A sale transaction immediately charges a customer's credit card when the batch of transactions is closed.

Secure Shell (SSH)
Secure Shell (SSH) is a network protocol that allows data to be exchanged over a secure channel between two computers.

Secure Sockets Layer (SSL)
Secure Sockets Layer (SSL) is cryptographic protocols that provide secure communications on the Internet, such as transmitting credit card data and other data transfers.

Settlement
Settlement is the completion of a payment transaction. When a transaction is settled, it has been funded and the monies deposited in the merchant account.

Store Name
The store name (also called storename or store number) is a six to ten-digit number needed to identify the merchant. The store name is given to the merchant in the Welcome E-mail. Merchants need the store name, user ID, and password to access the Virtual POS Terminal, as well as reports, admin, and customization functions. The store name is also needed for using the API and other products.

Ticket Only (PostAuth)
A Ticket Only transaction is a post-authorization transaction that captures funds from an Authorize Only transaction. Funds are transferred when your batch of transactions is settled. If you enter a larger total for the Ticket Only transaction than was specified for the Authorize Only transaction, the Ticket Only transaction may be blocked. If you enter a smaller amount than was authorized, an adjustment is made to the Authorization to reserve only the smaller amount of funds on the customer's card for the transaction.

Transit Routing Number
A transit routing number is a nine-digit bank code, used in the United States, which appears on the bottom of checks. This code is used by the Automated Clearing House to process direct deposits and other automated transfers.

URL
URL is short for Uniform Resource Locator. The URL is the address for documents and other pages on the Internet. The first part of the address indicates what protocol to use, and the second part specifies the IP address or the domain name where the resource is located.

User ID
For accounts with multiple users, each individual user will be assigned a User ID. The user will need this User ID, along with the store name and password, to log in to the system.

Virtual
Virtual is often used on the Internet to denote a web-based program that functions similarly to a physical device or system. For example, a virtual point-of-sale terminal is a computer program that performs the same functions as a physical point-of-sale terminal.

Void
To void a transaction is to cancel a payment transaction. Merchants can void transactions prior to settlement. Once the transaction has settled, the merchant has to perform a return or credit to reverse the charges and credit the customer's card.

WAN
A WAN is a wide-area computer network that spans a relatively large geographical area. Typically, a WAN consists of two or more local-area networks (LANs). Computers connected to a wide-area network are often connected through public networks, such as the telephone system. They can also be connected through leased lines or satellites.

Web Server
A web server is a computer program responsible for accepting HTTP requests from clients and serving HTTP responses along with optional data contents. The responses are usually web pages, such as HTML documents and linked objects (images, etc.).

XML
XML is the Extensible Markup Language, which is a universal format for the representation of documents and data. It is classified as an extensible language because it allows its users to define their own tags. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly through the Internet.

First Data™