



API Reference Guide for Web Services

June 2008

Version 1.0

This manual and accompanying electronic media are proprietary products of Optimal Payments Inc. They are to be used only by licensed users of the product.

© 1999–2008 Optimal Payments Inc. All rights reserved.

The information within this document is subject to change without notice. The software described in this document is provided under a license agreement, and may be used or copied only in accordance with this agreement. No part of this manual may be reproduced or transferred in any form or by any means without the express written consent of Optimal Payments Inc.

All other names, trademarks, and registered trademarks are the property of their respective owners.

Optimal Payments Inc. makes no warranty, either express or implied, with respect to this product, its merchantability or fitness for a particular purpose, other than as expressly provided in the license agreement of this product. For further information, please contact Optimal Payments Inc.

International Head Office

Optimal Payments Inc.

2 Place Alexis Nihon, Suite 700

Westmount, Quebec H3Z 3C1

Canada

Tel.: (514) 380-2700

Fax: (514) 380-2760

Email: info@optimalpayments.com

Technical support: support@optimalpayments.com

Web: www.optimalpayments.com

U.K. Office

Optimal Payments Ltd.

Quern House, Mill Court

Hinton Way, Great Shelford

Cambridge CB2 5LD

United Kingdom

Email: info@optimalpayments.co.uk

Technical Support: support@optimalpayments.co.uk

Web: www.optimalpayments.co.uk

U.S. Office

Optimal Payments Corp.

1800 West Loop South, #1440

Houston, TX 77027

Gatineau Office

Optimal Payments Inc.

75 Promenade du Portage

Gatineau, Quebec J8X 2J9

Canada

1 Optimal Payments Web Services

What are Optimal Payments Web Services?	1-1
Web Services supported for Direct Debit	1-1
Web Services supported for credit cards	1-1
System requirements	1-2
Accessing the Optimal Payments WSDLs and links	1-3
Direct Debit	1-3
Credit card	1-3
Testing Optimal Payments Web Services.	1-3
Security features	1-3
AVS	1-4
CVD	1-5
Negative database	1-5
3D Secure	1-5
Using this guide	1-6
Audience	1-6
Functionality.	1-6
Symbols	1-6

2 Direct Debit Transactions

Introduction	2-1
.NET example.	2-2
Building charge, verify, or credit requests	2-3
charge example – C#	2-3
ddCheckRequestV1 schema	2-5
ddCheckRequestV1 elements	2-6
Building updateShippingInfo requests	2-12
updateShippingInfo example – C#	2-12
ddShippingRequestV1 schema	2-14
ddShippingRequestV1 elements	2-15
Processing the response	2-16

3 Credit/Debit Card Transactions

Introduction	3-1
.NET example.	3-2
Building Purchase/Authorization/Verification requests	3-4

Purchase example – C#	3-4
ccAuthRequestV1 schema	3-7
ccAuthRequestV1 elements	3-8
Building Settlement/Credit requests	3-14
Settlement example – C#	3-14
ccPostAuthRequestV1 schema	3-15
ccPostAuthRequestV1 elements	3-16
Building Stored Data Requests	3-17
Stored Data Authorization example – C#	3-17
ccStoredDataRequestV1 schema	3-19
ccStoredDataRequestV1 elements	3-19
Building Cancel requests.	3-20
Cancel Settle example – C#	3-20
ccCancelRequestV1 schema	3-22
ccCancelRequestV1 elements	3-23
Building Payment requests	3-24
Payment example – C#	3-24
ccPaymentRequestV1 schema	3-26
ccPaymentRequestV1 elements	3-26
Building Authentication requests.	3-30
Authentication example – C#	3-30
ccAuthenticateRequestV1 schema	3-31
ccAuthenticateRequestV1 elements	3-31
Processing the response	3-33

A Using the HTTP Post Method

Introduction	A-1
Direct Debit requests	A-1
charge/verify/credit	A-1
HTML example – charge	A-2
updateShippingInfo	A-4
HTML example – updateShippingInfo	A-4
Credit card requests	A-5
Purchase/Authorization/Verification	A-5
HTML example – Authorization	A-7
Settlement/Credit	A-8
HTML example – Settlement	A-9
Stored Data Authorization/Purchase	A-10
HTML example – Stored Data Authorization	A-11
Cancel Settle/Credit/Payment	A-11
HTML example – Cancel	A-12
Payment request.	A-13
HTML example – Payment.	A-14
Authentication	A-15
HTML example – Authentication	A-15

Sample HTTP Post	A-16
Sample HTTP Post responses	A-16

B Response Codes

Overview	B-1
Response codes	B-1
Action codes	B-12
Return codes	B-12

C Geographical Codes

Province codes	C-1
State codes	C-2
Country codes	C-3

Optimal Payments Web Services

What are Optimal Payments Web Services?

Web Services are a technology that allows applications to communicate with each other in a platform- and programming language-independent manner. A Web Service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging. It uses protocols based on the XML language to describe an operation to execute or data to exchange with another Web Service. Web Services are built on open standards such as SOAP and WSDL.

Optimal Payments Web Services offer the following benefits:

- Merchants can integrate easily with the Web Service API, using their favourite platform and language.
- Merchants can automate operation and avoid manually keying in information via the Optimal Payments Web page.
- Merchants can operate independently of changes and updates to the Optimal Payments Web site.

Web Services supported for Direct Debit

Optimal Payments currently supports the following Web Services for Direct Debit:

Table 1-1: Direct Debit Operations

Operation	Description
verify	Allows you to confirm that a customer's bank account is in good standing, without actually transferring money out of that account.
charge	Allows you to transfer money from a customer's bank account to your merchant account. This transaction is completed in real time, though the banking network takes 3–5 days to transfer the funds.
credit	Allows you to transfer money from your merchant account directly to a customer's bank account. This transaction is completed in real time, though the banking network takes 3–5 days to transfer the funds.
updateShippingInfo	Allows you to send shipping information, including a tracking number, once you have shipped goods for which you previously processed a <i>charge</i> transaction.

Web Services supported for credit cards

Optimal Payments currently supports the following Web Services for credit cards:

Table 1-2: Credit Card Operations

Operation	Description
Authorization	Allows you to confirm that a credit card exists and has sufficient funds to cover a Purchase, but without settling the funds to your merchant account.
Purchase	Both authorizes and settles a requested amount against a credit card.
Verification	Allows you to run an AVS and/or CVD check on a credit card without processing a charge against that card.
Credit	Allows you to issue a credit for an amount that was previously settled.
Settlement	Allows you to Settle the amount of an existing Authorization, crediting the authorized amount from the credit card to your merchant account.
Stored Data Authorization	Allows you to authorize an amount on a customer's credit card using customer data that is stored in our database. You provide only a minimum of information, saving you time and effort.
Stored Data Purchase	Allows you to both authorize and settle an amount on a customer's credit card using customer data that is stored in our database. You provide only a minimum of information, saving you time and effort.
Cancel	Allows you to cancel a Credit, Settlement, or Payment transaction. You can cancel a Credit, Settlement, or Payment transaction as long as it is in a Pending state, which typically is before midnight of the day that it is requested. In some cases, you may find older Credit transactions in a Pending state.
Payment	Allows you to credit an amount to a customer's credit card. The Payment transaction is not associated with a previously existing authorization, so the amount of the transaction is not restricted in this way.
Authentication	Allows you to send an Authentication request, in order to validate a cardholder's password for credit cards enrolled in the 3D Secure program.

System requirements

The SOAP API has been tested with the following client environments:

Table 1-3: Client Environments

SOAP Client	Programming Environment	Operating Environment
Microsoft .NET 1.1 and 2.0 Framework	Microsoft Visual Studio .NET 2003	Microsoft Windows Server 2003 and Windows XP
Apache Axis 1.4	Java (J2SE 1.4.X and higher)	Linux and Microsoft Windows XP, Server 2003
Apache Axis 2.0	Java (J2SE 1.4.X and 1.5.X)	Linux and Microsoft Windows XP, Server 2003

For more information:

- J2SE or J2EE 1.3.1 or newer (1.4.X recommended) Sun Microsystems, Inc.
<http://java.sun.com/downloads/index.html>

- Apache Axis 1.4, The Apache Software Foundation
http://www.apache.org/dyn/closer.cgi/ws/axis/1_4
- Apache Axis2, 1.2, The Apache Software Foundation
http://ws.apache.org/axis2/1_2/contents.html
- Microsoft .NET Framework Version 1.1/2.0 Microsoft Corporation
<http://www.microsoft.com/net/default.mspix>



Whatever SOAP client you adopt, it must support document-style messaging.

Accessing the Optimal Payments WSDLs and links

Direct Debit

WSDL:

<https://webservices.optimalpayments.com/directdebitWS/DirectDebitService/v1?wsdl>

Web Service:

<https://webservices.optimalpayments.com/directdebitWS/DirectDebitService/v1>

HTTP Post:

<https://webservices.optimalpayments.com/directdebitWS/DirectDebitServlet/v1>

Credit card

WSDL:

<https://webservices.optimalpayments.com/creditcardWS/CreditCardService/v1?wsdl>

Web Service:

<https://webservices.optimalpayments.com/creditcardWS/CreditCardService/v1>

HTTP Post:

<https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1>

Testing Optimal Payments Web Services

Once you have configured your Web Services-enabled application, please contact our Technical Support team for instructions on how to test your API calls.

- Email support@optimalpayments.com
- Telephone 1-888-709-8753

Security features

For some transactions (e.g., Purchase) the Optimal Payments transaction processor provides additional features to protect the merchant from fraudulent card usage.

- Address verification system (AVS)
- Card validation data (CVD)
- Negative database
- 3D Secure

AVS

The Optimal Payments transaction processor supports address verification checks (AVS) wherever the issuing bank supports this feature. AVS verifies whether the address supplied by the customer using a card matches the billing address associated with that card at the issuing bank. This makes it more difficult to use the card fraudulently, since in order to use a stolen card someone must also know the billing address associated with it. In addition, if goods are to be shipped, the merchant can require that they be shipped to the billing address associated with the card.

Within Optimal Payments, each payment method is configured with the acceptable set of AVS return codes. If the bank returns a code that is not acceptable for the payment method, then the request is rejected with an Authorization Failed error. If you get an Authorization Failed error in response to a transaction request, and an Authorization number is returned in the response, then the failure was caused by the AVS check. You can look at the AVS code returned to determine exactly why the AVS check failed. The Optimal Payments transaction processor returns the following codes, in the *avsResponse* element, to the merchant application in response to a transaction request, with *A*, *N*, and *E* indicating AVS failure:

Table 1-4: AVS Codes

Code Letter	Explanation
A	Address matches, but zip code does not.
B	AVS not performed for international transaction. Either the postal code has invalid format or address information not provided.
E	AVS not supported for this industry.
M	For international transaction, address and postal code match.
N	No part of the address matches.
Q	Unknown response from issuer/banknet switch.
R	Retry. System unable to process.
S	AVS not supported.
U	Address information is unavailable.
W	Nine-digit zip code matches, but address does not.
X	Exact. Nine-digit zip code and address match.
Y	Yes. Five-digit zip code and address match.
Z	Five-digit zip code matches, but address does not.

When you registered with Optimal Payments, your account was set up to automatically apply AVS checks. The Optimal Payments transaction processor accepts only transactions for which the allowable AVS return codes are returned.

AVS has three limitations, which may affect the decisions you make with regard to failed AVS checks:

- AVS is not always reliable. Bad results can be returned if someone has moved, for instance, or because some people report five-digit zip codes and some report nine-digit zip codes.
- AVS does only limited support internationally. If you decide, therefore, to ship only to addresses that return good AVS results, you may exclude otherwise valid transactions.
- AVS is supported by many U.S. issuing banks, but not all. So even if you only serve U.S. customers, you may not always be able to depend on AVS being available.

CVD

The CVD value is a 3- or 4-digit number printed on the credit card, but which is not present on the magnetic strip. Therefore, the value is not printed on receipts or statements, reducing the probability of fraud from imprint information. The CVD feature, intended specifically for transactions where a card is not present, is a requirement of the Optimal Payments transaction processor. We support the CVD feature wherever the issuing banks do.

When customers enter their card and cardholder information, the CVD value is requested at the same time. One of four indicators flag the status of a CVD request:

- Not provided – The customer did not provide the CVD value.
- Provided – The customer provided the CVD value. When this indicator is selected, the CVD value is provided.
- Illegible – The customer claims the CVD value is illegible.
- Not present – The customer claims the CVD value is not on the card.

Negative database

Optimal Payments administers a negative database that provides additional protection against misuse of cards and inappropriate transaction requests. Card numbers and email addresses are entered into the negative database for the following reasons:

- A chargeback associated with the card number or email address has occurred.
- The card number or email address was involved in, or suspected to have been involved in, fraudulent activity.

Your account is configured to automatically implement this security feature, and any transaction request that attempts to use either a card number or email address that is in the negative database will not be processed.

3D Secure

Optimal Payments supports 3D Secure, an online cardholder authentication program designed to make Internet purchase transactions safer by authenticating a cardholder's identity at the time of purchase, before the merchant submits an authorization request. It is currently supported by several card brands, including Visa (Verified by Visa), MasterCard (SecureCode), and JCB (J/Secure). Authorizations processed using 3D Secure are guaranteed against most common types of chargeback disputes.

Using this guide

This user guide details major system functions. Each section provides an overview of functions, which are then broken down into procedures with steps to be followed.

Audience

This user's guide is intended for Optimal Payments merchants using our Web Services API to process transaction requests with Optimal Payments.



Functionality

This guide may document some features to which you do not have access. Access to such functionality is allotted on a merchant-by-merchant basis. If you have any questions, contact your account manager.

Symbols

This user guide uses the following symbols to bring important items to your attention:

Table 1-5: Symbols

Symbol	Description
	This note icon denotes a hint or tip to help you use the transaction processing application more efficiently.
	This warning icon alerts you about actions you might take that could have important consequences.

Direct Debit Transactions

Introduction

This chapter describes how to process Direct Debit transactions via the Transaction Processing Web Service. The following operations are supported:

Table 2-1: Supported Operations

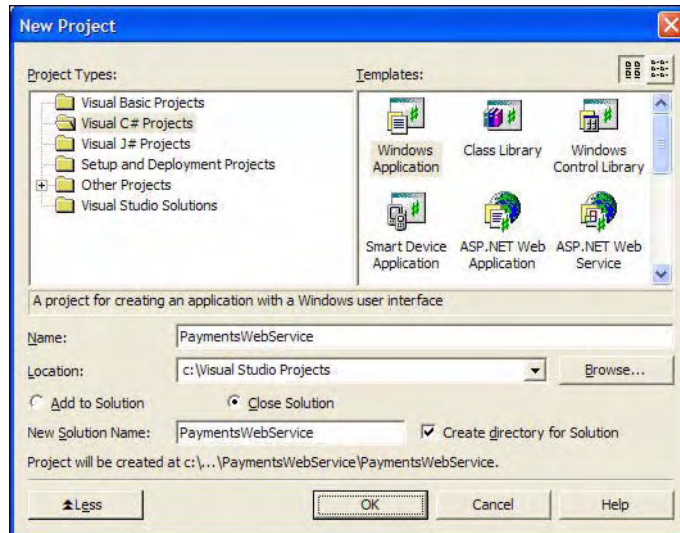
Operation	Description	Request Type
verify	Allows you to confirm that a customer's bank account is in good standing, without actually transferring money out of that account.	See <i>Building charge, verify, or credit requests</i> on page 2-3.
charge	Allows you to transfer money from a customer's bank account to your merchant account. This transaction is completed in real time, though the banking network takes 3–5 days to transfer the funds.	
credit	Allows you to transfer money from your merchant account directly to a customer's bank account. This transaction is completed in real time, though the banking network takes 3–5 days to transfer the funds.	
updateShippingInfo	Allows you to send shipping information, including a tracking number, once you have shipped goods for which you previously processed a <i>charge</i> transaction.	See <i>Building updateShippingInfo requests</i> on page 2-12.

- The *verify*, *charge*, and *credit* operations accept a *ddCheckRequestV1* document object.
- The *updateShippingInfo* operation accepts a *ddShippingRequestV1* document object.
- All operations return a *ddCheckResponseV1* response.

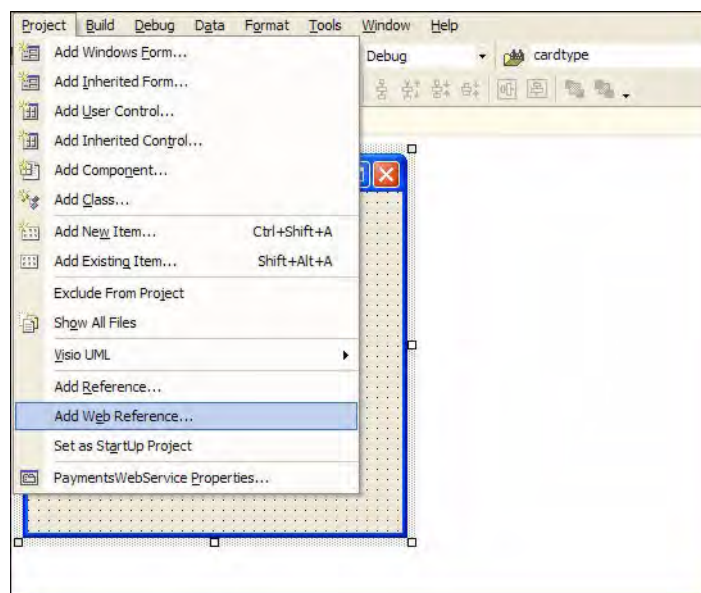
.NET example

To build the .NET example:

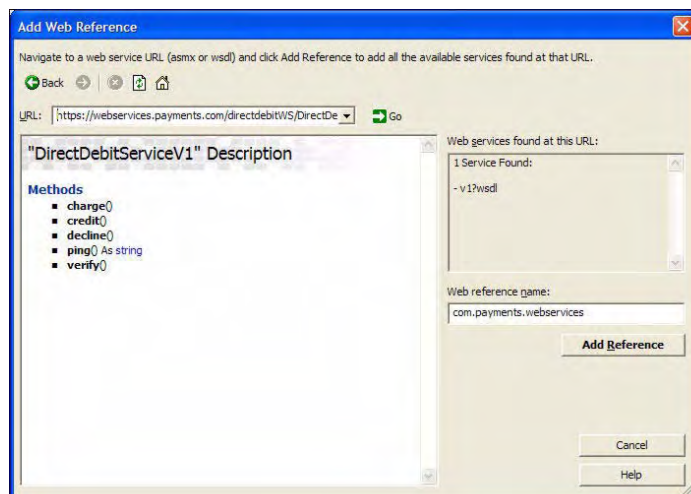
1. Create a new project.



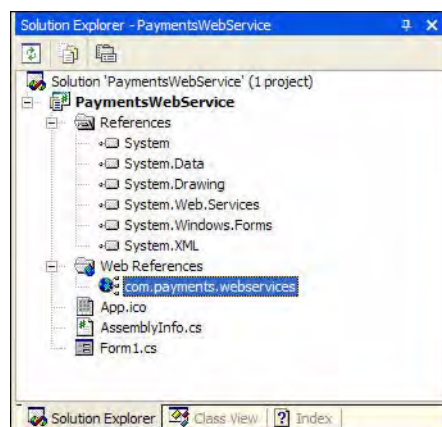
2. Add a Web Reference.



3. Enter the WSDL URL and click the Add Reference button.



The Web client is now built.



4. Build a Direct Debit request and process response.
 - See *Building charge, verify, or credit requests* on page 2-3
 - See *Building updateShippingInfo requests* on page 2-12
 - See *Processing the response* on page 2-16

Building charge, verify, or credit requests

Charge, verify, and credit requests require the *ddCheckRequestV1* document object. This section describes the structure of a *ddCheckRequestV1* and how to construct one. See Table 2-2: *ddCheckRequestV1 Elements* on page 2-6 for details on elements required.

charge example – C#

The following is a *charge* example in C#.



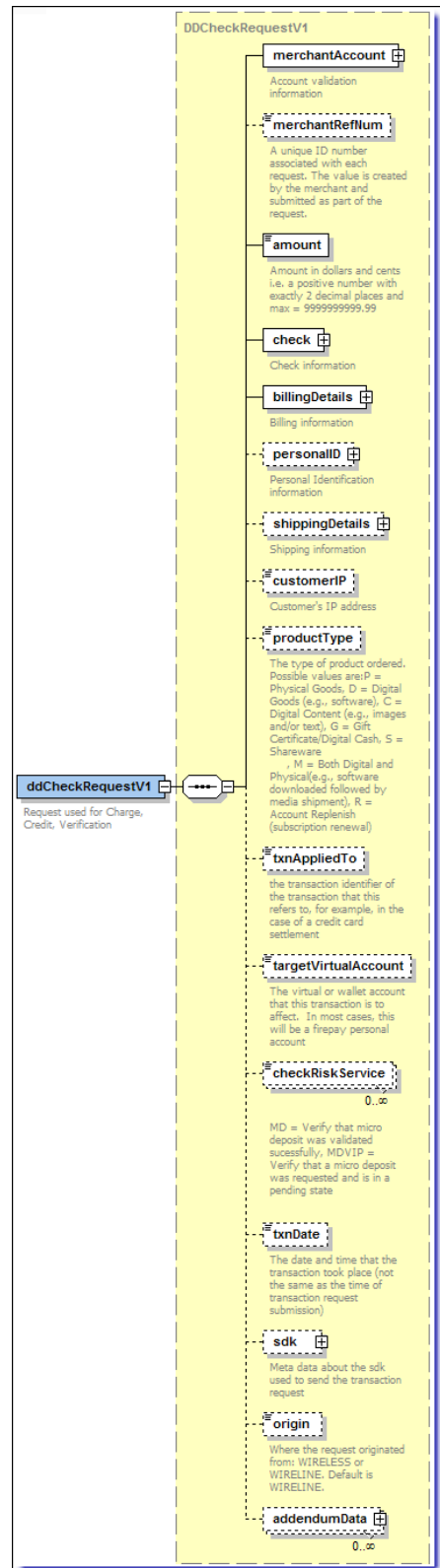
To make this a credit or verify request, just modify the value “charge” (underlined below) to “credit” or “verify”, respectively.

```
// Prepare the call to the Direct Debit Web Service
DDCheckRequestV1 ddCheckRequest = new DDCheckRequestV1();
ddCheckRequest.amount = "10.00";
MerchantAccountV1 merchantAccount = new MerchantAccountV1();
merchantAccount.accountNum = "12345678";
merchantAccount.storeID = "myStoreID";
merchantAccount.storePwd = "myStorePWD";
ddCheckRequest.merchantAccount = merchantAccount;
CheckV1 check = new CheckV1();
check.routingNum = "123456789";
check.accountNum = "987654321";
check.accountType = BankAccountTypeV1.PC;
check.bankName = "Chase";
check.checkNum = 12;
check.payee = "ACME Inc.";
ddCheckRequest.check = check;
BillingDetailsV1 billingDetails = new BillingDetailsV1();
billingDetails.firstName = "Jane";
billingDetails.lastName = "Jones";
billingDetails.street = "123 Main Street";
billingDetails.city = "LA";
billingDetails.country = CountryV1.US;
billingDetails.zip = "90210";
billingDetails.phone = "555-555-5555";
billingDetails.checkPayMethod = CheckPayMethodV1.WEB;
ddCheckRequest.billingDetails = billingDetails;
//Perform the Web Services call to process the charge
DirectDebitServiceV1 ddService = new DirectDebitServiceV1();
DDCheckResponseV1 ddTxnResponse = ddService.charge(ddCheckRequest);

// Print out the result
String responseTxt = ddTxnResponse.code + " - " + ddTxnResponse.decision +
    " - " + ddTxnResponse.description;
responseTxt += "Details:" + Environment.NewLine;
if (ddTxnResponse.detail != null)
{
    for (int i = 0; i < ddTxnResponse.detail.Length; i++)
    {
        responseTxt += " - " + ddTxnResponse.detail[i].tag + " - " +
            ddTxnResponse.detail[i].value + Environment.NewLine;
    }
}
responseTxt = responseTxt.Replace("\n", Environment.NewLine);
System.Console.WriteLine(responseTxt);
if (DecisionV1.ACCEPTED.Equals(ddTxnResponse.decision))
{
    System.Console.WriteLine("Transaction Successful.");
}
else
{
    System.Console.WriteLine("Transaction Failed with decision: " +
        ddTxnResponse.decision);
}
```


ddCheckRequestV1 schema

A *ddCheckRequestV1* document object has the following structure:



ddCheckRequestV1 elements

The *ddCheckRequestV1* document object may contain the following elements:

Table 2-2: ddCheckRequestV1 Elements

Element	Child Element	Required	Type	Description
merchantAccount	accountNum	Yes	String Max = 10	This is the merchant account number.
	storeID	Yes	String Max = 80	This is the transaction processing store identifier, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
	storePwd	Yes	String Max = 20	This is the transaction processing store password, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
merchantRefNum		Conditional	String Max = 40	This is a unique ID number associated with each request. The value is created by the merchant and submitted as part of the request.
amount		Yes	String Max = 999999999.99	This is amount of the transaction request.

Table 2-2: ddCheckRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
check	accountType	Yes	Enumeration	This is the type of checking account used for the transaction. Possible values are: <ul style="list-style-type: none"> • PC (Personal Checking) • PS (Personal Savings) • PL (Personal Loan) • BC (Business Checking) • BS (Business Savings) • BL (Business Loan)
	bankName	Yes	String Max = 40	This is the name of the customer's bank, to which this transaction is posted.
	checkNum	Yes	Long Max = 8	This is the check serial number, provided at the time of the transaction request. NOTE: The <i>checkNum</i> element is required for the <i>verify</i> operation, where it serves as a unique transaction ID, even though no money is transferred between accounts for this operation.
	accountNum	Yes	String Max = 17	This is the customer's bank account number.
	routingNum	Yes	String Min = 6 Max = 9	For USD accounts, this is the 9-digit routing number of the customer's bank. For British pound accounts, this is the 6-digit sort code of the customer's bank. For Canadian dollar accounts, this is a combination of the 3-digit institution ID and the 5-digit transit number of the customer's bank branch. Do not include spaces or dashes.
	payee	Conditional	String Max = 81	This is the descriptor that will appear on the customer's bank account. It is required only for <i>credit</i> and <i>verify</i> transactions. If you provide a value for this field, this value will be used as the statement descriptor. If you do not provide a value for this field, a default value configured for the merchant account will be used.

Table 2-2: ddCheckRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
billingDetails	checkPayMethod	Yes	Enumeration	This is the payment type. Possible values are: <ul style="list-style-type: none"> • WEB (Personal Check Only) • TEL (Personal Check Only) • PPD (Personal Check Only) • CCD (Business Check Only)
	firstName	Conditional	String Max = 40	This is the customer's first name. Required if <i>checkPayMethod</i> is set to WEB or TEL.
	lastName	Conditional	String Max = 40	This is the customer's last name. Required if <i>checkPayMethod</i> is set to WEB or TEL.
	companyName	Conditional	String Max = 50	This is the company's name. Required if <i>checkPayMethod</i> is a business check.
	street	Yes	String Max = 50	This is the first line of the customer's street address.
	street2	Optional	String Max = 50	This is the second line of the customer's street address.
	city	Yes	String Max = 40	This is the city in which the customer resides.
	state/region	Conditional	If state, then Enumeration If region, then string Max = 40	This is the state/province/region in which the customer resides. Provide <i>state</i> if within U.S./Canada. Provide <i>region</i> if outside of U.S./Canada. See <i>Appendix C: Geographical Codes</i> for correct codes to use.
	country	Yes	Enumeration	This is the country in which the customer resides. See <i>Country codes</i> on page C-3 for correct codes to use.
	zip	Optional	String Max = 10	This is the customer's ZIP code if in the U.S.; otherwise, this is the customer's postal code.
	phone	Yes	String Max = 40	This is the customer's telephone number.
	email	Optional	String Max = 100	This is the customer's email address.

Table 2-2: ddCheckRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
personalID	idNumber	Optional	String Max = 20	This is the number of the ID provided for the <i>idType</i> element.
	idType	Optional	Enumeration	This is the type of ID used to identify the owner of the bank account. Possible values are: <ul style="list-style-type: none"> • DL (Driver's License) • SS (Government ID) • MI (Military ID) • GN (Generic ID)
	state/region	Optional	If state, then Enumeration If region, then string Max = 40	This is the state/province/region in which the ID was issued. Provide <i>state</i> if within U.S./Canada. Provide <i>region</i> if outside of U.S./Canada. See <i>Appendix C: Geographical Codes</i> for correct codes to use.
	country	Optional	String Length = 2	This is the country in which the ID was issued. Possible values are: <ul style="list-style-type: none"> • CA (Canada) • US (United States)
	expiry	Optional		The <i>expiry</i> child element has three further child elements.
		Child Element of expiry		
	day	Optional	Int Length = 2	This is the day the ID expires.
	month	Optional	Int Length = 2	This is the month the ID expires.
	year	Conditional	Int Length = 4	This is the year the ID expires. This element is required if the <i>expiry</i> element is included.

Table 2-2: ddCheckRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
shippingDetails	carrier	Optional	Enumeration	This is the shipment carrier. Possible values are: <ul style="list-style-type: none"> • APC = APC Overnight • APS = AnPost • CAD = Canada Postal Service • DHL • FEX = Fedex • RML = Royal Mail • UPS = United Parcel Service • USPS = United States Postal Service • OTHER
	trackingNumber	Optional	String Max = 50	This is the shipping tracking number provided by the carrier.
	shipMethod	Optional	Enumeration	This is the method of shipment. Possible values are: <ul style="list-style-type: none"> • N = Next Day/Overnight • T = Two-Day Service • C = Lowest Cost • O = Other
	firstName	Optional	String Max = 40	This is the recipients's first name.
	lastName	Optional	String Max = 40	This is the recipient's last name.
	street	Optional	String Max = 50	This is the first line of the recipient's street address.
	street2	Optional	String Max = 50	This is the second line of the recipient's street address.
	city	Optional	String Max = 40	This is the city in which the recipient resides.
	state/region	Optional	If state, Enumeration If region, then string Max = 40	This is the state/province/region in which the recipient resides. Provide <i>state</i> if within U.S./Canada. Provide <i>region</i> if outside of U.S./Canada. See <i>Appendix C: Geographical Codes</i> for correct codes to use.
	country	Optional	Enumeration	This is the country in which the recipient resides.
	zip	Optional	String Max = 10	This is the recipient's ZIP code if in the U.S.; otherwise, this is the recipient's postal code.
	phone	Optional	String Max = 40	This is the recipient's phone number.
	email	Optional	String Max = 100	This is the recipient's email address.

Table 2-2: ddCheckRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
customerIP		Optional	String Max = 50	This is the customer's IP address.
productType		Optional	Enumeration	This is the type of product sold. Possible values are: <ul style="list-style-type: none"> • P (Physical Goods) • D (Digital Goods) • C (Digital Content) • G (Gift Certificate/Digital Cash) • S (Shareware) • M (Both Digital and Physical) • R (Account Replenish)
txnAppliedTo		No		This element is not applicable for Direct Debit transactions.
targetVirtualAccount		No		This element is not applicable for Direct Debit transactions.
checkRiskService		No		This element is not applicable for Direct Debit transactions.
txnDate		Optional	dateTime	This is the date and time that the transaction took place.
sdk	version	Conditional	String Max = 20	This is the version of the SDK used, if any. Required if <i>sdk</i> element is provided.
	platform	Conditional	String Max = 10	This is the integration language of the SDK used (e.g., Java, .NET). Required if <i>sdk</i> element is provided.
	provider	Conditional	String Max = 20	This is the author of the SDK used. Set to value "op" when the SDK is provided by Optimal Payments. Required if <i>sdk</i> element is provided.
origin		Optional	Enumeration	This is the origin of the request. Possible values are: <ul style="list-style-type: none"> • Wireless • Wireline
addendumData	tag	Optional	String Max = 30	This is additional data that the merchant can include with the transaction request.
	value	Optional	String Max = 1024	This is additional data that the merchant can include with the transaction request.

Building updateShippingInfo requests



All optional elements that take non-nullable data types (e.g., int or enum) must have their specified attribute set to true when setting values for those elements. See the shipMethod element in the example below.

The `updateShippingInfo` request requires the `ddShippingRequestV1` document object. This section describes the structure of a `ddShippingRequestV1` and how to construct one. See Table 2-3: `ddShippingRequestV1 Elements` on page 2-15 for details on elements required.

updateShippingInfo example – C#

The following is an `updateShippingInfo` example in C#.

```
// Prepare the call to the Direct Debit Web Service
DDShippingRequestV1 ddShippingRequest = new DDShippingRequestV1();
MerchantAccountV1 merchantAccount = new MerchantAccountV1();
merchantAccount.accountNum = "12345678";
merchantAccount.storeID = "myStoreID";
merchantAccount.storePwd = "myStorePwd";
ddShippingRequest.merchantAccount = merchantAccount;
ddShippingRequest.confirmationNumber = "123456"; // A valid confirmation number
from a previous settle auth or purchase
ddShippingRequest.carrier = CarrierV1.FEX; // Fedex
ddShippingRequest.shipMethod = ShipMethodV1.T;
ddShippingRequest.shipMethodSpecified = true;
ddShippingRequest.trackingNumber = "555666888999";
ddShippingRequest.firstName = "Jane";
ddShippingRequest.lastName = "Jones";
ddShippingRequest.street = "123 Main Street";
ddShippingRequest.city = "LA";
ddShippingRequest.country = CountryV1.US;
ddShippingRequest.countrySpecified = true;
ddShippingRequest.zip = "90210";
ddShippingRequest.email = "janejones@emailserver.com"; // optional email address

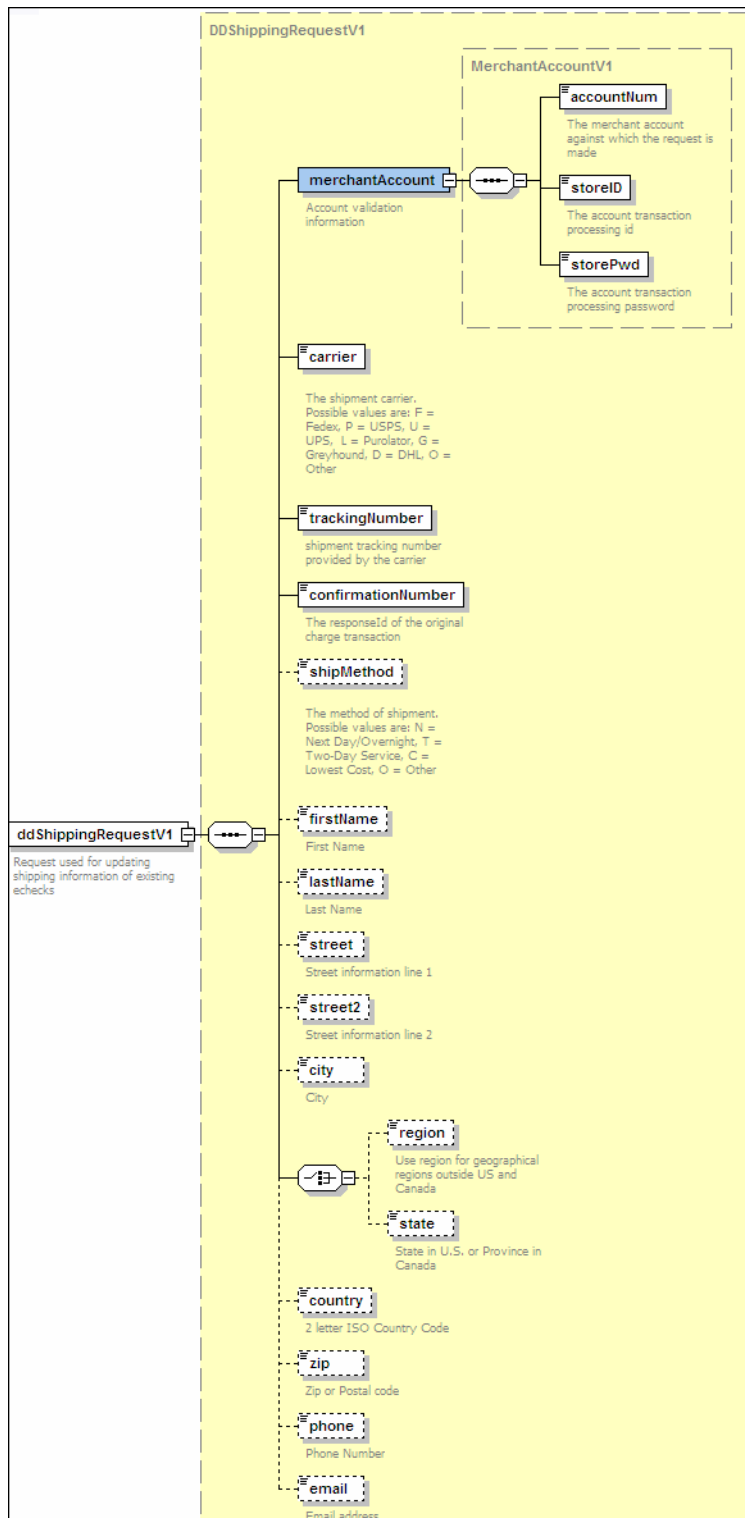
//Perform the Web Services call to update the shipping info
DirectDebitServiceV1 ddService = new DirectDebitServiceV1();
DDCheckResponseV1 ddTxnResponse =
ddService.updateShippingInfo(ddShippingRequest);
// Print out the result
String responseTxt = ddTxnResponse.code + " - " + ddTxnResponse.decision + " - " +
    ddTxnResponse.description;
responseTxt += "Details:" + Environment.NewLine;
if (ddTxnResponse.detail != null)
{
    for (int i = 0; i < ddTxnResponse.detail.Length; i++)
    {
        responseTxt += " - " + ddTxnResponse.detail[i].tag + " - " +
            ddTxnResponse.detail[i].value + Environment.NewLine;
    }
}
responseTxt = responseTxt.Replace("\n", Environment.NewLine);
System.Console.WriteLine(responseTxt);
if (DecisionV1.ACCEPTED.Equals(ddTxnResponse.decision))
{
    System.Console.WriteLine("Transaction Successful.");
}
```



```
    }  
    else  
    {  
        System.Console.WriteLine("Transaction Failed with decision: " +  
            ddTxnResponse.decision);  
    }
```

ddShippingRequestV1 schema

A *ddShippingRequestV1* document object has the following structure:



ddShippingRequestV1 elements

The *ddShippingRequestV1* document object may contain the following elements:

Table 2-3: ddShippingRequestV1 Elements

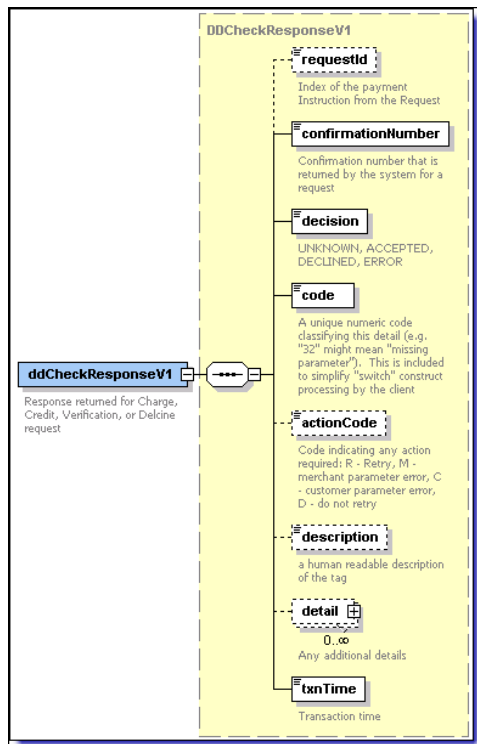
Element	Child Element	Required	Type	Description
merchantAccount	accountNum	Yes	String Max = 10	This is the merchant account number.
	storeID	Yes	String Max = 20	This is the transaction processing store identifier, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
	storePwd	Yes	String Max = 20	This is the transaction processing store password, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
carrier		Yes	Enumeration	This is the shipment carrier. Possible values are: <ul style="list-style-type: none"> • APC = APC Overnight • APS = AnPost • CAD = Canada Postal Service • DHL • FEX = Fedex • RML = Royal Mail • UPS = United Parcel Service • USPS = United States Postal Service • OTHER
trackingNumber		Yes	String Max = 50	This is the shipping tracking number provided by the carrier.
confirmationNumber		Yes	String Max = 20	This is the confirmation number returned by the payment processor.
shipMethod		Optional	Enumeration	This is the method of shipment. Possible values are: <ul style="list-style-type: none"> • N = Next Day/Overnight • T = Two-Day Service • C = Lowest Cost • O = Other
firstName		Optional	String Max = 40	This is the customer's first name.
lastName		Optional	String Max = 40	This is the customer's last name.
street		Optional	String Max = 50	This is the first line of the customer's street address.
street2		Optional	String Max = 50	This is the second line of the customer's street address.

Table 2-3: ddShippingRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
city		Optional	String Max = 40	This is the city in which the customer resides.
state/region		Optional	If state, then Enumeration If region, then string Max = 40	This is the state/province/region in which the customer resides. Provide <i>state</i> if within U.S./Canada. Provide <i>region</i> if outside of U.S./Canada. See <i>Appendix C: Geographical Codes</i> for correct codes to use.
country		Optional	Enumeration	This is the country in which the customer resides. See <i>Country codes</i> on page C-3 for correct codes to use.
zip		Optional	String Max = 10	This is the customer's ZIP code if in the U.S.; otherwise, this is the customer's postal code.
phone		Optional	String Max = 40	This is the customer's telephone number.
email		Optional	String Max = 100	This is the customer's email address.

Processing the response

A *ddCheckResponseV1* has the following structure:



The following elements are relevant for a *ddCheckResponseV1*:

Table 2-4: ddCheckResponseV1 Elements

Element	Child Element	Required	Type	Description
confirmationNumber		Yes	String Max = 20	This is the confirmation number returned by Optimal Payments.
decision		Yes	Enumeration	This is the status of the transaction. One of the following is returned: <ul style="list-style-type: none"> Accepted – the transaction was processed. Error – the transaction was attempted, but failed for some reason. Declined – the transaction was declined before it was sent for processing.
code		Yes	Int	This is a numeric code that categorizes the response. See <i>Response codes</i> on page B-1.
actionCode		Optional	Enumeration	This indicates what action to take. The following values are possible: <ul style="list-style-type: none"> C = Consumer Parameter Error. The consumer has provided incorrect information. Ask the customer to correct the information. D = Do Not Retry. Further attempts will fail. M = Merchant Parameter Error. Your application has provided incorrect information. Verify your information. R = Retry. The problem is temporary. Retrying the request will likely succeed.
description		Optional	String Max = 100	This is a human readable description of the <i>code</i> element.
detail	tag	Optional	String Max = 1024	This is the classification of the detail element.
	value	Optional	String Max = 30	This is the description of the detail.
txnTime		Yes	dateTime	This is the date and time the transaction was processed by the payment processor.

To process the response:

1. Get the response details, which are available via `get()` methods of the response.

```
String responseTxt = ddTxnResponse.code + " - " + ddTxnResponse.decision +
    " - " + ddTxnResponse.description;
responseTxt += "Details:" + Environment.NewLine;
if (ddTxnResponse.detail != null)
{
    for (int i = 0; i < ddTxnResponse.detail.Length; i++)
    {
        responseTxt += " - " + ddTxnResponse.detail[i].tag + " - " +
            ddTxnResponse.detail[i].value + Environment.NewLine;
    }
}
```

```
}
responseTxt = responseTxt.Replace("\n", Environment.NewLine);
System.Console.WriteLine(responseTxt);
if (DecisionV1.ACCEPTED.Equals(ddTxnResponse.decision))
{
    System.Console.WriteLine("Transaction Successful.");
}
else
{
    System.Console.WriteLine("Transaction Failed with decision: " +
        ddTxnResponse.decision);
}
```

2. Process based on the *decision* element. You would insert handling code appropriate to your application. You can also look at the *code* element to provide more fine-grained control for your application. See *Response codes* on page B-1 for more details.

Credit/Debit Card Transactions

Introduction

This chapter describes how to process credit and debit card transactions via the Transaction Processing Web Service. The following operations are supported:

Table 3-1: Supported Operations

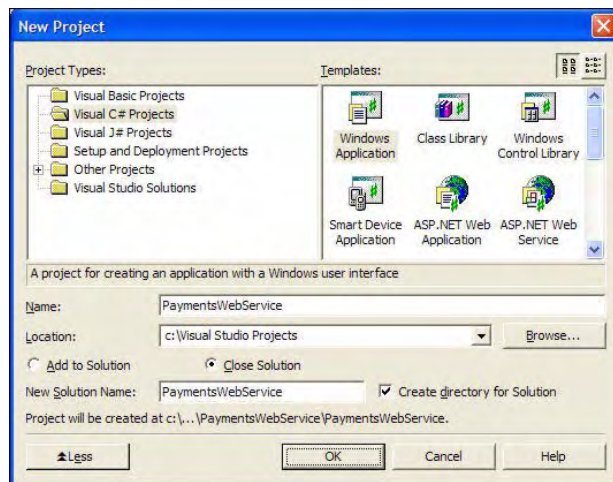
Operation	Description	Request Type
Authorization	Allows you to authorize an amount on a customer's credit/debit card. The authorized amount must be settled in a subsequent settlement transaction.	See <i>Building Purchase/Authorization/Verification requests</i> on page 3-4.
Purchase	Allows you to both authorize and settle an amount on a customer's credit/debit card in a single transaction.	
Verification	Allows you to run an AVS and/or CVD check on a customer's credit card without processing a charge against that card.	
Credit	Allows you to credit back to a customer's credit/debit card an amount that has previously been settled. You can credit all or part of an existing settlement.	See <i>Building Settlement/Credit requests</i> on page 3-14.
Settlement	Allows you to settle an amount that was previously authorized on a customer's credit/debit card. You can settle all or part of an existing authorization.	
Stored Data Authorization	Allows you to authorize an amount on a customer's credit/debit card using customer data that is stored in our database. You provide only a minimum of information, saving you time and effort.	See <i>Building Stored Data Requests</i> on page 3-17.
Stored Data Purchase	Allows you to both authorize and settle an amount on a customer's credit/debit card using customer data that is stored in our database. You provide only a minimum of information, saving you time and effort.	
Cancel	Allows you to cancel a Credit, Settlement, or Payment transaction. You can cancel a Credit, Settlement, or Payment transaction as long as it is in a Pending state, which typically is before midnight of the day that it is requested. In some cases, you may find older Credit transactions in a Pending state.	See <i>Building Cancel requests</i> on page 3-20.
Payment	Allows you to credit an amount to a customer's credit/debit card. The Payment transaction is not associated with a previously existing authorization, so the amount of the transaction is not restricted in this way.	See <i>Building Payment requests</i> on page 3-24.
Authentication	Allows you to send an Authentication request, in order to validate a cardholder's password for credit cards enrolled in the 3D Secure program.	See <i>Building Authentication requests</i> on page 3-30.

- The *Authorization*, *Purchase*, and *Verification* operations accept a *ccAuthRequestV1* document object.
- The *Settlement* and *Credit* operations accept a *ccPostAuthRequestV1* document object.
- The *Stored Data* operations accept a *ccStoredDataRequestV1* document object.
- The *Cancel Settle*, *Cancel Credit*, and *Cancel Payment* operations accept a *ccCancelRequestV1* document object.
- The *Payment* operation accepts a *ccPaymentRequestV1* document object.
- The *Authentication* operation accepts a *ccAuthenticateRequestV1* document object.
- All operations return a *ccTxnResponseV1* response.

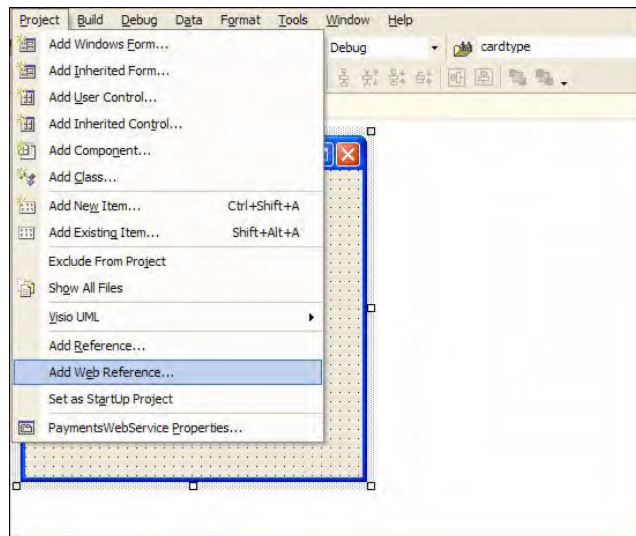
.NET example

To build the .NET example:

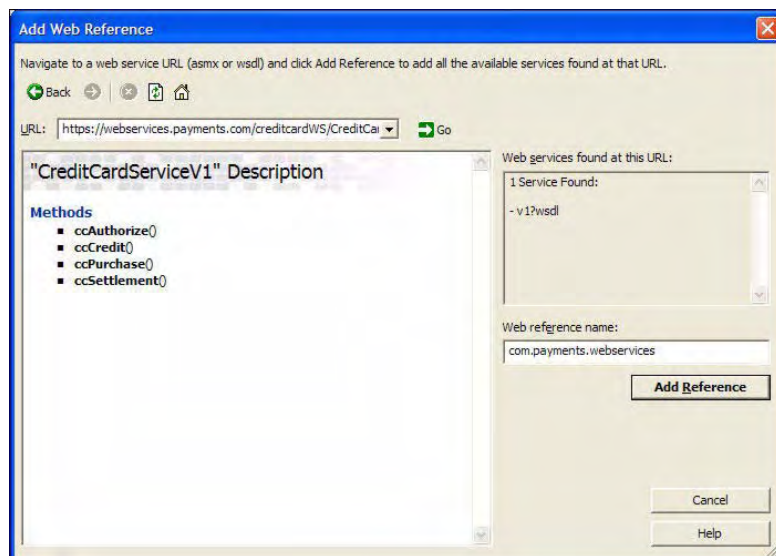
1. Create a new project.



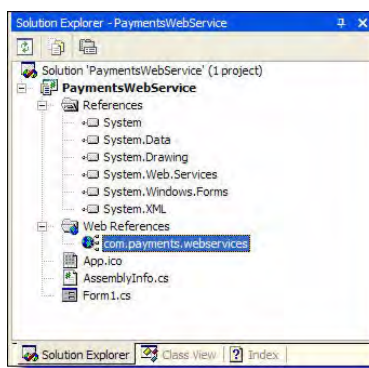
2. Add a Web Reference.



3. Enter the WSDL URL and click the Add Reference button.



The Web client is now built.



4. Build the request and process response:

- See *Building Purchase/Authorization/Verification requests* on page 3-4
- See *Building Settlement/Credit requests* on page 3-14
- See *Building Stored Data Requests* on page 3-17
- See *Building Cancel requests* on page 3-20
- See *Building Payment requests* on page 3-24
- See *Building Authentication requests* on page 3-30
- See *Processing the response* on page 3-33

Building Purchase/Authorization/Verification requests

Purchase, Authorization, and Verification requests require the `ccAuthRequestV1` document object. This section describes the structure of a `ccAuthRequestV1` and how to construct one. See Table 3-2: `ccAuthRequestV1 Elements` on page 3-8 for details on the elements required.

Purchase example – C#



All optional elements that take non-nullable data types (e.g., int or enum) must have their specified attribute set to true when setting values for those elements. See the `cardType` element in the example below.

The following is a Purchase example in C#.



To make this an Authorize or Verification request, just modify the value "`ccPurchase`" (underlined below) to "`ccAuthorize`" or "`ccVerification`", respectively.

```
//Prepare the call to the Credit Card Web Service
CCAuthRequestV1 ccAuthRequest = new CCAuthRequestV1();
MerchantAccountV1 merchantAccount = new MerchantAccountV1();
merchantAccount.accountNum = "12345678";
merchantAccount.storeID = "myStoreID";
merchantAccount.storePwd = "myStorePWD";
ccAuthRequest.merchantAccount = merchantAccount;
ccAuthRequest.merchantRefNum = "Ref-12345";
ccAuthRequest.amount = "10.00";
CardV1 card = new CardV1();
card.cardNum = "4653111111111111";
CardExpiryV1 cardExpiry = new CardExpiryV1();
cardExpiry.month = 11;
cardExpiry.year = 2006;
card.cardExpiry = cardExpiry;
card.cardType = CardTypeV1.VI;
card.cardTypeSpecified = true;
card.cvdIndicator = 1;
card.cvdIndicatorSpecified = true;
card.cvd = "111";
ccAuthRequest.card = card;
BillingDetailsV1 billingDetails = new BillingDetailsV1();
billingDetails.cardPayMethod = CardPayMethodV1.WEB; //WEB = Card Number Provided
billingDetails.cardPayMethodSpecified = true;
```

```

billingDetails.firstName = "Jane";
billingDetails.lastName = "Jones";
billingDetails.street = "123 Main Street";
billingDetails.city = "LA";
billingDetails.Item = (object) StateV1.CA; // California
billingDetails.country = CountryV1.US; // United States
billingDetails.countrySpecified = true;
billingDetails.zip = "90210";
billingDetails.phone = "555-555-5555";
billingDetails.email = "janejones@emailserver.com";
ccAuthRequest.billingDetails = billingDetails;
ccAuthRequest.previousCustomer = true;
ccAuthRequest.previousCustomerSpecified = true;
ccAuthRequest.customerIP = "127.0.0.1";
ccAuthRequest.productType = ProductTypeV1.M;
//M = Both Digital and Physical(e.g., software downloaded followed by media
shipment)
ccAuthRequest.productTypeSpecified = true;

//Request a 3D Secure Lookup
CardRiskServiceV1[] riskServices = { CardRiskServiceV1.TDS };
ccAuthRequest.cardRiskService = riskServices;

// Perform the Web Services call for the purchase
CreditCardServiceV1 ccService = new CreditCardServiceV1();
CCTxnResponseV1 ccTxnResponse = ccService.ccPurchase(ccAuthRequest);

// Print out the result
String responseTxt = ccTxnResponse.code + " - " + ccTxnResponse.decision + " - "
    + ccTxnResponse.description + Environment.NewLine;

// Print out the PAREq and ACSUrl

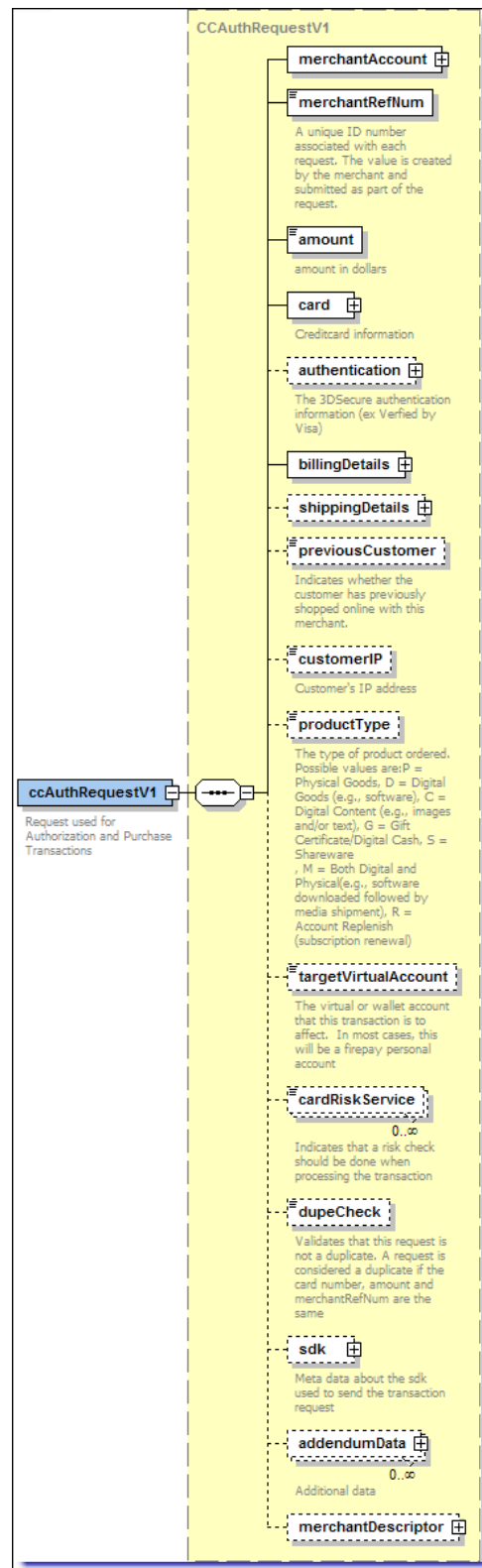
if ((ccTxnResponse.tdsResponse != null) &&
(ccTxnResponse.tdsResponse.paymentRequest != null) &&
(ccTxnResponse.tdsResponse.termURL != null))
{
    responseTxt += "PaReq: " + ccTxnResponse.tdsResponse.paymentRequest +
Environment.NewLine +
        "ACSUrl: " + ccTxnResponse.tdsResponse.acsURL +
Environment.NewLine;
}
responseTxt += "Details:" + Environment.NewLine;
if (ccTxnResponse.detail != null)
{
    for (int i = 0; i < ccTxnResponse.detail.Length; i++)
    {
        responseTxt += " - " + ccTxnResponse.detail[i].tag + " - " +
            ccTxnResponse.detail[i].value + Environment.NewLine;
    }
}
responseTxt = responseTxt.Replace("\n", Environment.NewLine);
System.Console.WriteLine(responseTxt);
if (DecisionV1.ACCEPTED.Equals(ccTxnResponse.decision))
{
    System.Console.WriteLine("Transaction Successful.");
}
else
{

```

```
        System.Console.WriteLine("Transaction Failed with decision: " +  
        ccTxnResponse.decision);  
    }
```

ccAuthRequestV1 schema

A *ccAuthRequestV1* has the following structure:



ccAuthRequestV1 elements

The *ccAuthRequestV1* document object may contain the following elements:

Table 3-2: ccAuthRequestV1 Elements

Element	Child Element	Required	Type	Description
merchantAccount	accountNum	Yes	String Max = 10	This is the merchant account number.
	storeID	Yes	String Max = 80	This is the transaction processing store identifier, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
	storePwd	Yes	String Max = 20	This is the transaction processing store password, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
merchantRefNum		Yes	String Max = 40	This is a unique ID number associated with each request. The value is created by the merchant and submitted as part of the request.
amount		Yes	String Max = 999999999.99	This is amount of the transaction request. NOTE: Though mandatory, this value will be ignored for the <i>ccVerification</i> transaction.

Table 3-2: ccAuthRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
card	cardNum	Yes	String Min = 8 Max = 20	This is the card number used for the transaction.
	cardExpiry	Yes		The <i>cardExpiry</i> child element has two further child elements – <i>month</i> and <i>year</i> .
		Child Element of cardExpiry		
	month	Yes	Int Max = 2	This is the month the credit card expires.
	year	Yes	Int Length = 4	This is the year the credit card expires.
	cardType	Optional	Enumeration	This is the type of card used for the transaction. Possible values are: <ul style="list-style-type: none"> • AM = American Express • CB = Carte Blanche • DC = Diners Club • DI = Discover • FP = FirePay • JC = JCB • LA = Laser • MC = MasterCard • MD = Maestro • N = Novus • SO = Solo • VD = Visa Delta • VE = Visa Electron • VI = Visa
	issueNum	Optional	Integer Max = 2	The 1- or 2-digit number located on the front of the card, following the card number. NOTE: The <i>issueNum</i> element can be used only when the <i>cardType</i> is SO (Solo).
	cvdIndicator	Optional	Integer Length = 1	This is the status of CVD value information. Possible values are: <ul style="list-style-type: none"> • 0 – The customer did not provide a value. • 1 – The customer provided a value. • 2 – The value is illegible. • 3 – The value is not on the card. NOTE: The <i>cvdIndicator</i> element is mandatory for the <i>ccVerification</i> transaction. Also note that even though this element is optional, it is required for several risk-related checks and we strongly advise you to include it to avoid failed transactions.
	cvd	Conditional	String Length = 3 or 4	The 3- or 4-digit security code that appears on the card following the card number. This code does not appear on imprints. NOTE: The <i>cvd</i> element is mandatory when the <i>cvdIndicator</i> element value = 1.

Table 3-2: ccAuthRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
authentication NOTE: Do not include this element if you are using the <i>cardRiskService</i> element to perform a 3D Secure security check.	indicator	Optional	Integer Value = 1	This is the Electronic Commerce Indicator code, a 1-digit value that gets returned by the card issuer indicating whether the cardholder was successfully authenticated.
	cavv	Optional	String Max = 80	This is the Cardholder Authentication Verification Value returned in the Authentication Response by the card issuer.
	xid	Optional	String Max = 80	This is the transaction identifier returned in the Authentication Response by the card issuer.

Table 3-2: ccAuthRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
billingDetails	cardPayMethod	Optional	Enumeration	This is the method used to make the payment. Possible values are: <ul style="list-style-type: none"> • WEB – Card number provided online • P – Card present • TEL – Card number provided by phone
	firstName	Optional	String Max = 40	This is the customer's first name.
	lastName	Optional	String Max = 40	This is the customer's last name.
	street	Optional	String Max = 50	This is the first line of the customer's street address. NOTE: the <i>street</i> element is mandatory if you are processing a <i>ccVerification</i> transaction.
	street2	Optional	String Max = 50	This is the second line of the customer's street address.
	city	Optional	String Max = 40	This is the city in which the customer resides.
	state/region	Optional	If state, Enumeration If region, then string Max = 40	This is the state/province/region in which the customer resides. Provide <i>state</i> if within U.S./Canada. Provide <i>region</i> if outside of U.S./Canada. See <i>Appendix C: Geographical Codes</i> for correct codes to use.
	country	Optional	Enumeration	This is the country in which the customer resides. See <i>Country codes</i> on page C-3 for correct codes to use.
	zip	Mandatory	String Max = 10	This is the customer's ZIP code if in the U.S.; otherwise, this is the customer's postal code.
	phone	Optional	String Max = 40	This is the customer's telephone number.
	email	Optional	String Max = 100	This is the customer's email address.

Table 3-2: ccAuthRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
shippingDetails	carrier	Optional	Enumeration	This is the shipment carrier. Possible values are: <ul style="list-style-type: none"> • APC = APC Overnight • APS = AnPost • CAD = Canada Postal Service • DHL • FEX = Fedex • RML = Royal Mail • UPS = United Parcel Service • USPS = United States Postal Service • OTHER
	shipMethod	Optional	Enumeration	The method of shipment. Possible values are: <ul style="list-style-type: none"> • N = Next Day/Overnight • T = Two-Day Service • C = Lowest Cost • O = Other
	firstName	Optional	String Max = 40	This is the recipients's first name.
	lastName	Optional	String Max = 40	This is the recipient's last name.
	street	Optional	String Max = 50	This is the first line of the recipient's street address.
	street2	Optional	String Max = 50	This is the second line of the recipient's street address.
	city	Optional	String Max = 40	This is the city in which the recipient resides.
	state/region	Optional	If state, Enumeration If region, then string Max = 40	This is the state/province/region in which the recipient resides. Provide <i>state</i> if within U.S./Canada. Provide <i>region</i> if outside of U.S./Canada. See <i>Appendix C: Geographical Codes</i> for correct codes to use.
	country	Optional	Enumeration	This is the country in which the recipient resides.
	zip	Optional	String Max = 10	This is the recipient's ZIP code if in the U.S.; otherwise, this is the recipient's postal code.
	phone	Optional	String Max = 40	This is the recipient's phone number.
	email	Optional	String Max = 100	This is the recipient's email address.
previousCustomer		Optional	boolean	Indicates whether the customer has previously shopped online with this merchant.

Table 3-2: ccAuthRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
customerIP		Optional	String Max = 50	This is the customer's IP address.
productType		Optional	Enumeration	This is the type of product sold. Possible values are: <ul style="list-style-type: none"> • P (Physical Goods) • D (Digital Goods) • C (Digital Content) • G (Gift Certificate/Digital Cash) • S (Shareware) • M (Both Digital and Physical) • R (Account Replenish)
targetVirtualAccount		No		This element is not applicable for credit card transactions.
cardRiskService NOTE: Do not use this element if you are processing <i>Verification</i> transactions. Use this element only for <i>Authorization</i> and <i>Purchase</i> transactions.		Optional		Include this element if you want to use 3D Secure security check on a credit card transaction. Currently, Visa, MasterCard, Maestro, and JCB support the 3D Secure security check. Use the following value: <ul style="list-style-type: none"> • TDS
dupeCheck		Optional	Boolean	This validates that this request is not a duplicate. A request is considered a duplicate if the <i>cardNum</i> , <i>amount</i> , and <i>merchantRefNum</i> are the same.
sdk	version	Conditional	String Max = 20	This is the version of the SDK used, if any. Required if <i>sdk</i> element is provided.
	platform	Conditional	String Max = 10	This is the integration language of the SDK used (e.g., Java, .NET). Required if <i>sdk</i> element is provided.
	provider	Conditional	String Max = 20	This is the author of the SDK used. Set to value "op" when the SDK is provided by Optimal Payments. Required if <i>sdk</i> element is provided.
addendumData	tag	Optional	String Max = 30	This is additional data that the merchant can include with the transaction request.
	value	Optional	String Max = 1024	This is additional data that the merchant can include with the transaction request.

Building Settlement/Credit requests

Settlement and Credit requests require the *ccPostAuthRequestV1* document object. This section describes the structure of a *ccPostAuthRequestV1* and how to construct one. See Table 3-3: *ccPostAuthRequestV1 Elements* on page 3-16 for details on the elements required.

Settlement example – C#

The following is a Settlement example in C#.



To make this a Credit request, just modify the value “ccSettlement” (underlined below) to “ccCredit”.

```
//Prepare the call to the Credit Card Web Service
CCPostAuthRequestV1 ccPostAuthRequest = new CCPostAuthRequestV1();
ccPostAuthRequest.confirmationNumber = "123456";
MerchantAccountV1 merchantAccount = new MerchantAccountV1();
merchantAccount.accountNum = "12345678";
merchantAccount.storeID= "myStoreID";
merchantAccount.storePwd = "myStorePWD";
ccPostAuthRequest.merchantAccount = merchantAccount;
ccPostAuthRequest.merchantRefNum = "Ref-12345";
ccPostAuthRequest.amount = "10.00";

// Perform the Web Service call for the Settlement
CreditCardServiceV1 ccService = new CreditCardServiceV1();
CCTxnResponseV1 ccTxnResponse = ccService.ccSettlement(ccPostAuthRequest);

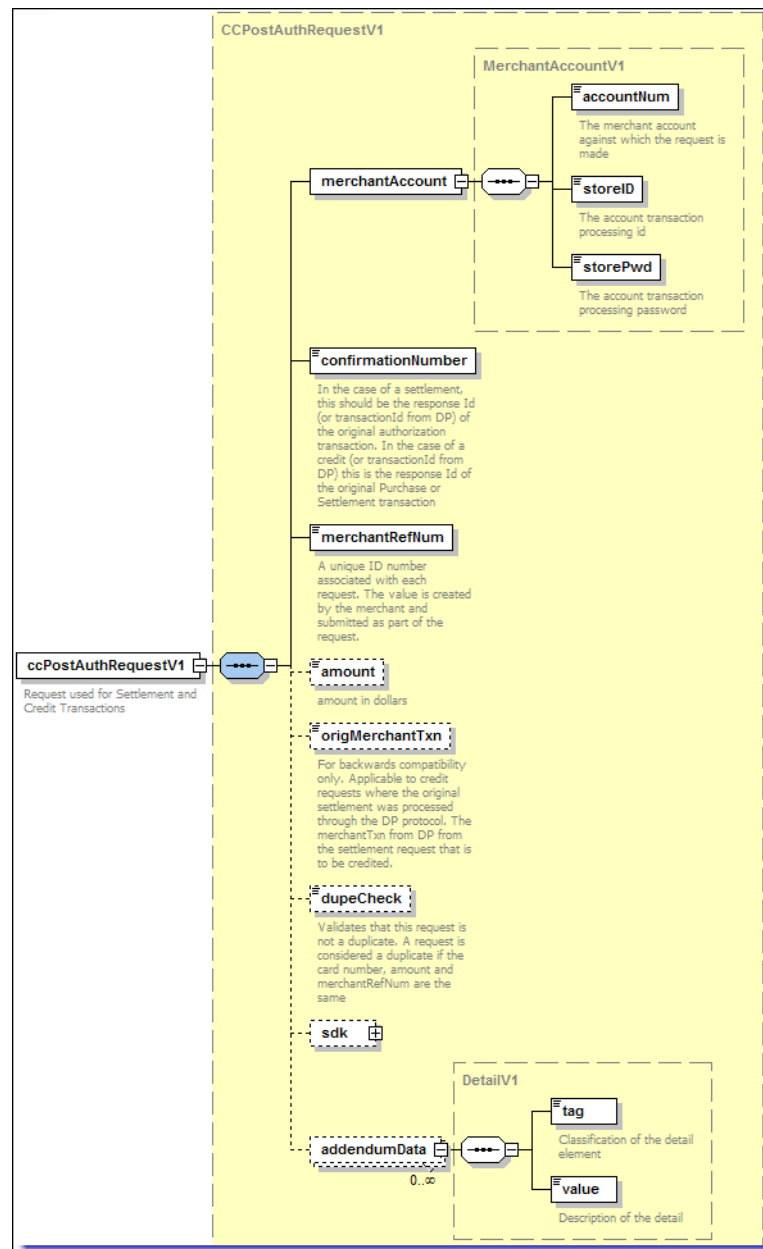
// Print out the result
String responseTxt = ccTxnResponse.code + " - " + ccTxnResponse.decision +
                    " - " + ccTxnResponse.description ;
responseTxt += "Details:" + Environment.NewLine;

if (ccTxnResponse.detail != null)
{
    for (int i = 0; i < ccTxnResponse.detail.Length; i++)
    {
        responseTxt += " - " + ccTxnResponse.detail[i].tag + " - " +
            ccTxnResponse.detail[i].value + Environment.NewLine;
    }
}
responseTxt = responseTxt.Replace("\n", Environment.NewLine);
System.Console.WriteLine(responseTxt);

if (DecisionV1.ACCEPTED.Equals(ccTxnResponse.decision))
{
    System.Console.WriteLine("Transaction Successful.");
}
else
{
    System.Console.WriteLine("Transaction Failed with decision: " +
        ccTxnResponse.decision);
}
```

ccPostAuthRequestV1 schema

A *ccPostAuthRequestV1* document object has the following structure:



ccPostAuthRequestV1 elements

The *ccPostAuthRequestV1* document object may contain the following elements:

Table 3-3: ccPostAuthRequestV1 Elements

Element	Child Element	Required	Type	Description
merchantAccount	accountNum	Yes	String Max = 10	This is the merchant account number.
	storeID	Yes	String Max = 20	This is the transaction processing store identifier, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
	storePwd	Yes	String Max = 20	This is the transaction processing store password, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
confirmationNumber		Yes	String Max = 20	This is the confirmation number returned by the payment processor for the original request.
merchantRefNum		Yes	String Max = 40	This is a unique ID number associated with each request. The value is created by the merchant and submitted as part of the request.
amount		Optional	String Max= 999999999.99	This is amount of the transaction request. You can Settle all or part of an Authorization. You can Credit all or part of a Settlement.
origMerchantTxn		Conditional	String Max = 255	This is the merchant transaction ID from a Settlement that was processed via the Direct Payment API and that is now being credited via the Web Services API.
dupeCheck		Optional	Boolean	This validates that this request is not a duplicate. A request is considered a duplicate if the <i>cardNum</i> , <i>amount</i> , and <i>merchantRefNum</i> are the same.
sdk	version	Conditional	String Max = 20	This is the version of the SDK used, if any. Required if <i>sdk</i> element is provided.
	platform	Conditional	String Max = 10	This is the integration language of the SDK used (e.g., Java, .NET). Required if <i>sdk</i> element is provided.
	provider	Conditional	String Max = 20	This is the author of the SDK used. Set to value "op" when the SDK is provided by Optimal Payments. Required if <i>sdk</i> element is provided.

Table 3-3: ccPostAuthRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
addendumData	tag	Optional	String Max = 30	This is additional data that the merchant can include with the transaction request.
	value	Optional	String Max = 1024	This is additional data that the merchant can include with the transaction request.

Building Stored Data Requests

Stored Data Authorization/Purchase requests require the *ccStoredDataRequestV1* document object. This section describes the structure of a *ccStoredDataRequestV1* and how to construct one. See Table 3-4: *ccStoredDataRequestV1 Elements* on page 3-19 for details on the elements required.

Stored Data requests allow you to perform credit card Authorizations and Purchases by providing a minimum of customer information. The Stored Data request requires a Confirmation Number from a previous Authorization or Purchase. This Confirmation Number allows Optimal Payments to access from its database most of the data required for the transaction.

Stored Data Authorization example – C#

The following is a Stored Data Authorization example in C#.



To make this a Stored Data Purchase request, just modify the value “ccStoredDataAuthorize” (underlined below) to “ccStoredDataPurchase”.

```
//Prepare the call to the Credit Card Web Service
MerchantAccountV1 merchantAccount = new MerchantAccountV1();
merchantAccount.accountNum = "1000000127";
merchantAccount.storeID = "test";
merchantAccount.storePwd = "test";

CCStoredDataRequestV1 ccStoredDataRequest = new CCStoredDataRequestV1();
ccStoredDataRequest.confirmationNumber = "111374429";
ccStoredDataRequest.amount = "97.97";
ccStoredDataRequest.merchantRefNum = "jim55";
ccStoredDataRequest.merchantAccount = merchantAccount;

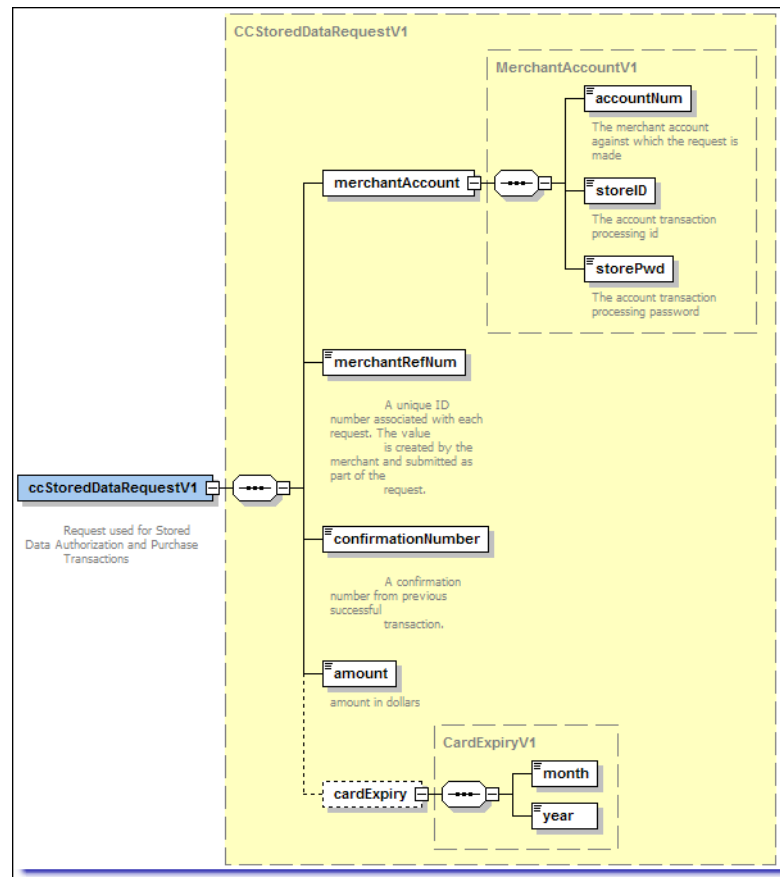
// Perform the Web Service call for the Stored Data Transaction
CreditCardServiceV1 ccService = new CreditCardServiceV1();
CCTxnResponseV1 ccTxnResponse =
ccService.ccStoredDataAuthorize(ccStoredDataRequest);
String responseTxt = "";

// Print out the result
if (ccTxnResponse.detail != null)
{
    for (int i = 0; i < ccTxnResponse.detail.Length; i++)
    {
        responseTxt += " - " + ccTxnResponse.detail[i].tag + " - " +
ccTxnResponse.detail[i].value + Environment.NewLine;
    }
}
```

```
}
responseTxt = responseTxt.Replace("\n", Environment.NewLine);
System.Console.WriteLine(responseTxt);
if (DecisionV1.ACCEPTED.Equals(ccTxnResponse.decision))
{
    System.Console.WriteLine("Transaction Successful.");
}
else
{
    System.Console.WriteLine("Transaction Failed with decision: " +
ccTxnResponse.decision);
}
```


ccStoredDataRequestV1 schema

A *ccStoredDataRequestV1* document object has the following structure:



ccStoredDataRequestV1 elements

The *ccStoredDataRequestV1* document object may contain the following elements:

Table 3-4: ccStoredDataRequestV1 Elements

Element	Child Element	Required	Type	Description
merchantAccount	accountNum	Required	String Max = 10	This is the merchant account number.
	storeID	Required	String Max = 20	This is the transaction processing store identifier, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
	storePwd	Required	String Max = 20	This is the transaction processing store password, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.

Table 3-4: ccStoredDataRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
merchantRefNum		Required	String Max = 40	This is a unique ID number associated with each request. The value is created by the merchant and submitted as part of the request.
confirmationNumber		Required	String Max = 20	This is the confirmation number returned by the payment processor for the original request.
amount		Required	String Max=999999999.99	This is amount of the transaction request.
cardExpiry	month	Optional	Int Max = 2	This is the month the credit card expires. Use this element to include updated card expiry date information, if required, to be included with the rest of the transaction data.
	year	Optional	Int Length = 4	This is the year the credit card expires. Use this element to include updated card expiry date information, if required, to be included with the rest of the transaction data.

Building Cancel requests

Use the *ccCancelRequestV1* document object to cancel *Settle*, *Credit*, and *Payment* transactions. This section describes the structure of a *ccCancelRequestV1* and how to construct one. See Table 3-5: *ccCancelRequestV1 Elements* on page 3-23 for details on the elements required.

Cancel Settle example – C#

The following is a Cancel Settle example in C#.



To make this a request to cancel a Credit or a Payment, just modify the value “ccCancelSettle” (underlined below) to “ccCancelCredit” or “ccCancelPayment”, respectively.

```
//Prepare the call to the Credit Card Web Service
CCCancelRequestV1 ccCancelRequest = new CCCancelRequestV1();
MerchantAccountV1 merchantAccount = new MerchantAccountV1();
merchantAccount.accountNum = "12345678";
merchantAccount.storeID= "myStoreID";
merchantAccount.storePwd = "myStorePWD";
ccCancelRequest.merchantAccount = merchantAccount;
ccCancelRequest.confirmationNumber = "123456";

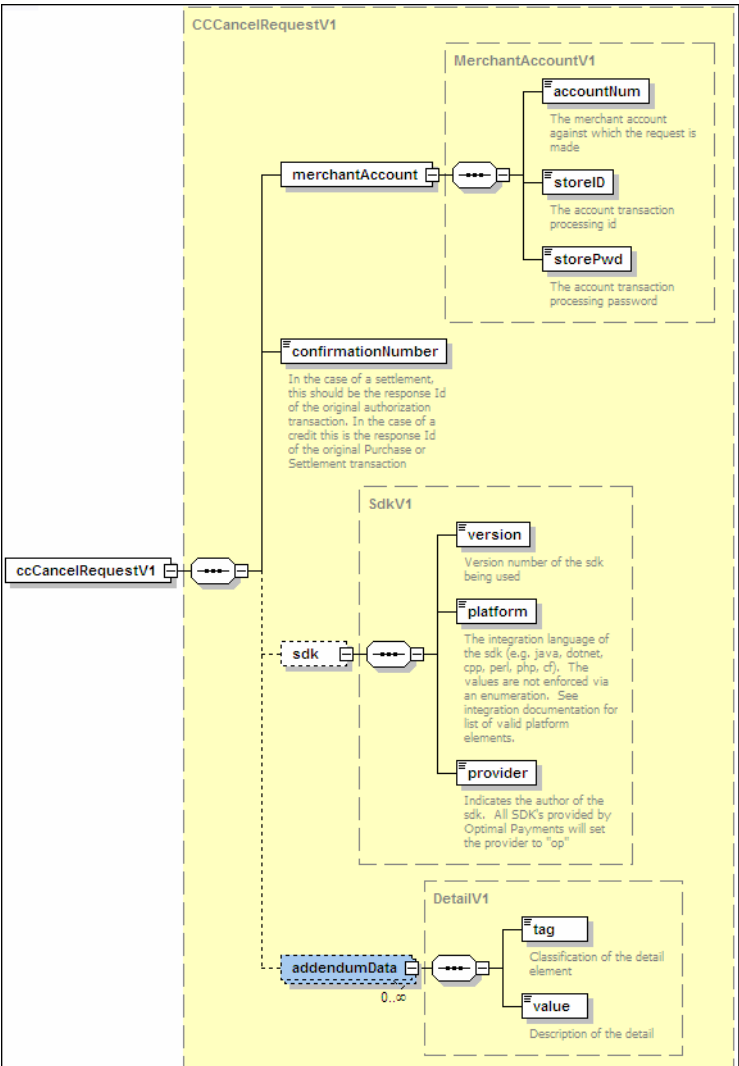
// Perform the Web Services call for the cancel settle
CreditCardServiceV1 ccService = new CreditCardServiceV1();
CCTxnResponseV1 ccTxnResponse = ccService.ccCancelSettle(ccCancelRequest);
```

```
// Print out the result
String responseTxt = ccTxnResponse.code + " - " + ccTxnResponse.decision +
    " - " + ccTxnResponse.description ;
responseTxt += "Details:" + Environment.NewLine;
if (ccTxnResponse.detail != null)
{
    for (int i = 0; i < ccTxnResponse.detail.Length; i++)
    {
        responseTxt += " - " + ccTxnResponse.detail[i].tag + " - " +
            ccTxnResponse.detail[i].value + Environment.NewLine;
    }
}
responseTxt = responseTxt.Replace("\n", Environment.NewLine);
System.Console.WriteLine(responseTxt);

if (DecisionV1.ACCEPTED.Equals(ccTxnResponse.decision))
{
    System.Console.WriteLine("Transaction Successful.");
}
else
{
    System.Console.WriteLine("Transaction Failed with decision: " +
        ccTxnResponse.decision);
}
```

ccCancelRequestV1 schema

A *ccCancelRequestV1* document object has the following structure:



ccCancelRequestV1 elements

The *ccCancelRequestV1* document object may contain the following elements:

Table 3-5: ccCancelRequestV1 Elements

Element	Child Element	Required	Type	Description
merchantAccount	accountNum	Yes	String Max = 10	This is the merchant account number.
	storeID	Yes	String Max = 20	This is the transaction processing store identifier, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
	storePwd	Yes	String Max = 20	This is the transaction processing store password, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
confirmationNumber		Yes	String Max = 20	This is the confirmation number returned by the payment processor for the original Settlement or Credit request.
sdk	version	Conditional	String Max = 20	This is the version of the SDK used, if any. Required if <i>sdk</i> element is provided.
	platform	Conditional	String Max = 10	This is the integration language of the SDK used (e.g., Java, .NET). Required if <i>sdk</i> element is provided.
	provider	Conditional	String Max = 20	This is the author of the SDK used. Set to value "op" when the SDK is provided by Optimal Payments. Required if <i>sdk</i> element is provided.
addendumData	tag	Optional	String Max = 30	This is additional data that the merchant can include with the transaction request.
	value	Optional	String Max = 1024	This is additional data that the merchant can include with the transaction request.

Building Payment requests

Payments require the *ccPaymentRequestV1* document object. This section describes the structure of a *ccPaymentRequestV1* and how to construct one. See Table 3-6: *ccPaymentRequestV1 Elements* on page 3-26 for details on the elements required.

Payment example – C#

The following is a Payment example in C#.

```
//Prepare the call to the Credit Card Web Service
CCPaymentRequestV1 ccPaymentRequest = new CCPaymentRequestV1();
MerchantAccountV1 merchantAccount = new MerchantAccountV1();
merchantAccount.accountNum = "12345678";
merchantAccount.storeID = "myStoreID";
merchantAccount.storePwd = "myStorePWD";
ccPaymentRequest.merchantAccount = merchantAccount;
ccPaymentRequest.merchantRefNum = "Ref-12345";
ccPaymentRequest.amount = "10.00";
CardV1 card = new CardV1();
card.cardNum = "4653111111111111";
CardExpiryV1 cardExpiry = new CardExpiryV1();
cardExpiry.month = 11;
cardExpiry.year = 2006;
card.cardExpiry = cardExpiry;
card.cardType = CardTypeV1.VI;
card.cardTypeSpecified = true;
card.cvdIndicator = 1;
card.cvdIndicatorSpecified = true;
card.cvd = "111";
ccPaymentRequest.card = card;
BillingDetailsV1 billingDetails = new BillingDetailsV1();
billingDetails.cardPayMethod = CardPayMethodV1.WEB; //WEB = Card Number Provided
billingDetails.cardPayMethodSpecified = true;
billingDetails.firstName = "Jane";
billingDetails.lastName = "Jones";
billingDetails.street = "123 Main Street";
billingDetails.city = "LA";
billingDetails.Item = (object)StateV1.CA; // California
billingDetails.country = CountryV1.US; // United States
billingDetails.countrySpecified = true;
billingDetails.zip = "90210";
billingDetails.phone = "555-555-5555";
billingDetails.email = "janejones@emailserver.com";
ccPaymentRequest.billingDetails = billingDetails;

// Perform the Web Services call for the payment request
CreditCardServiceV1 ccService = new CreditCardServiceV1();
CCTxnResponseV1 ccTxnResponse = ccService.ccPayment(ccPaymentRequest);

// Print out the result
String responseTxt ccTxnResponse.code + " - " + ccTxnResponse.decision + " - "
    + ccTxnResponse.description ;

responseTxt += "Details:" + Environment.NewLine;

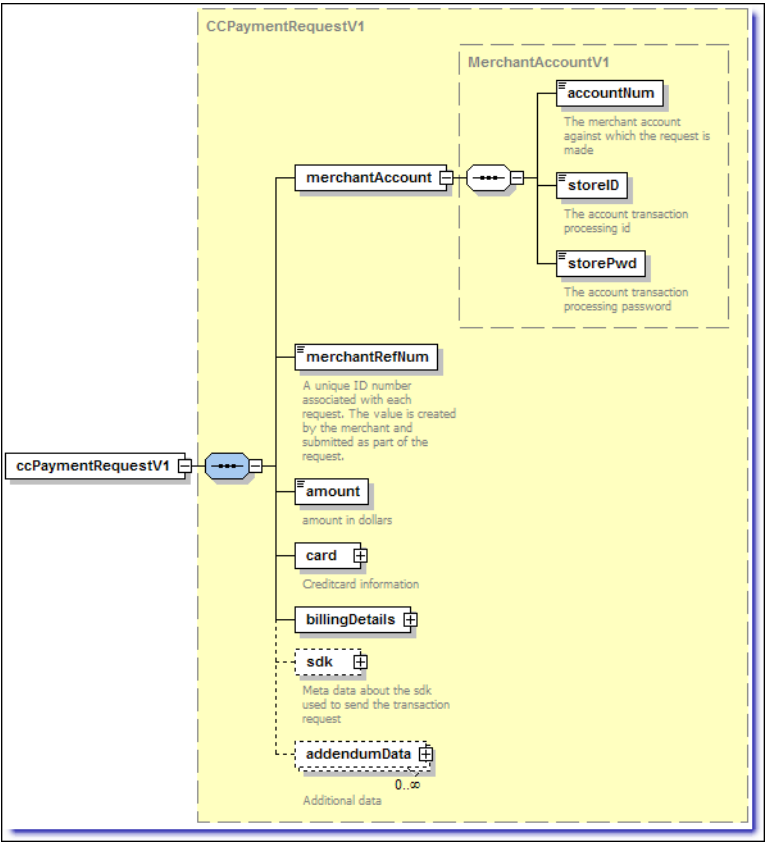
if (ccTxnResponse.detail != null)
{
```

```
        for (int i = 0; i < ccTxnResponse.detail.Length; i++)
        {
            responseTxt += " - " + ccTxnResponse.detail[i].tag + " - " +
                ccTxnResponse.detail[i].value + Environment.NewLine;
        }

    }
    responseTxt = responseTxt.Replace("\n", Environment.NewLine);
    System.Console.WriteLine(responseTxt);
    if (DecisionV1.ACCEPTED.Equals(ccTxnResponse.decision))
    {
        System.Console.WriteLine("Transaction Successful.");
    }
    else
    {
        System.Console.WriteLine("Transaction Failed with decision: " +
            ccTxnResponse.decision);
    }
}
catch (WebException we)
{
    consoleTextBox.Text += we.Message.ToString();
    consoleTextBox.Refresh();
}
```

ccPaymentRequestV1 schema

A *ccPaymentRequestV1* has the following structure:



ccPaymentRequestV1 elements

The *ccPaymentRequestV1* document object may contain the following elements:

Table 3-6: ccPaymentRequestV1 Elements

Element	Child Element	Required	Type	Description
merchantAccount	accountNum	Yes	String Max = 10	This is the merchant account number.
	storeID	Yes	String Max = 80	This is the transaction processing store identifier, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
	storePwd	Yes	String Max = 20	This is the transaction processing store password, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.

Table 3-6: ccPaymentRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
merchantRefNum		Yes	String Max = 40	This is a unique ID number associated with each request. The value is created by the merchant and submitted as part of the request.
amount		Yes	String	This is amount of the transaction request. The amount maximum is configured on a merchant-by-merchant basis. It applies to each transaction and to the daily maximum per credit card. Contact your account manager for details.

Table 3-6: ccPaymentRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
card	cardNum	Yes	String Min = 8 Max = 20	This is the card number used for the transaction.
	cardExpiry	Yes		The <i>cardExpiry</i> child element has two further child elements – <i>month</i> and <i>year</i> .
	Child Element of cardExpiry			
	month	Yes	Int Max = 2	This is the month the credit card expires.
	year	Yes	Int Length = 4	This is the year the credit card expires.
	cardType	Optional	Enumeration	This is the type of card used for the transaction. Possible values are: <ul style="list-style-type: none"> • AM = American Express • CB = Carte Blanche • DC = Diners Club • DI = Discover • FP = FirePay • JC = JCB • LA = Laser • MC = MasterCard • MD = Maestro • N = Novus • SO = Solo • VD = Visa Delta • VE = Visa Electron • VI = Visa
	issueNum	Optional	Integer Max = 2	The 1- or 2-digit number located on the front of the card, following the card number. NOTE: The <i>issueNum</i> element can be used only when the <i>cardType</i> is SO (Solo).
	cvdIndicator	Optional	Integer Length = 1	This is the status of CVD value information. Possible values are: <ul style="list-style-type: none"> • 0 – The customer did not provide a value. • 1 – The customer provided a value. • 2 – The value is illegible. • 3 – The value is not on the card. NOTE: Even though this element is optional, it is required for several risk-related checks and we strongly advise you to include it to avoid failed transactions.
	cvd	Conditional	String Length = 3 or 4	The 3- or 4-digit security code that appears on the card following the card number. This code does not appear on imprints. NOTE: The <i>cvd</i> element is mandatory when the <i>cvdIndicator</i> element value = 1.

Table 3-6: ccPaymentRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
billingDetails	cardPayMethod	Optional	Enumeration	This is the method used to make the payment. Possible values are: <ul style="list-style-type: none"> • WEB – Card number provided online • P – Card present • TEL – Card number provided by phone
	firstName	Optional	String Max = 40	This is the customer's first name.
	lastName	Optional	String Max = 40	This is the customer's last name.
	street	Optional	String Max = 50	This is the first line of the customer's street address.
	street2	Optional	String Max = 50	This is the second line of the customer's street address.
	city	Optional	String Max = 40	This is the city in which the customer resides.
	state/region	Optional	If state, Enumeration If region, then string Max = 40	This is the state/province/region in which the customer resides. Provide <i>state</i> if within U.S./Canada. Provide <i>region</i> if outside of U.S./Canada. See <i>Appendix C: Geographical Codes</i> for correct codes to use.
	country	Optional	Enumeration	This is the country in which the customer resides. See <i>Country codes</i> on page C-3 for correct codes to use.
	zip	Mandatory	String Max = 10	This is the customer's ZIP code if in the U.S.; otherwise, this is the customer's postal code.
	phone	Optional	String Max = 40	This is the customer's telephone number.
	email	Optional	String Max = 100	This is the customer's email address.
sdk	version	Conditional	String Max = 20	This is the version of the SDK used, if any. Required if <i>sdk</i> element is provided.
	platform	Conditional	String Max = 10	This is the integration language of the SDK used (e.g., Java, .NET). Required if <i>sdk</i> element is provided.
	provider	Conditional	String Max = 20	This is the author of the SDK used. Set to value "op" when the SDK is provided by Optimal Payments. Required if <i>sdk</i> element is provided.

Table 3-6: ccPaymentRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
addendumData	tag	Optional	String Max = 30	This is additional data that the merchant can include with the transaction request.
	value	Optional	String Max = 1024	This is additional data that the merchant can include with the transaction request.

Building Authentication requests

Authentication requests require the *ccAuthenticateRequestV1* document object. This section describes the structure of a *ccAuthenticateRequestV1* and how to construct one. See Table 3-7: *ccAuthenticateRequestV1 Elements* on page 3-31 for details on the elements required.

Authentication example – C#

The following is an Authentication example in C#.

```
//Prepare the call to the Credit Card Web Service
CCAuthenticateRequestV1 authenticateRequest = new CCAuthenticateRequestV1();
MerchantAccountV1 merchantAccount = new MerchantAccountV1();
merchantAccount.accountNum = "12345678";
merchantAccount.storeID = "myStoreID";
merchantAccount.storePwd = "myStorePWD";
authenticateRequest.merchantAccount = merchantAccount;
authenticateRequest.confirmationNumber = "myConfirmationNumber";
authenticateRequest.paymentResponse = "myPaymentResponse";

// Perform the Web Services call for the authentication
CreditCardServiceV1 ccService = new CreditCardServiceV1();
CCTxnResponseV1 ccTxnResponse = ccService.ccAuthenticate(authenticateRequest);

// Print out the result
String responseTxt = ccTxnResponse.confirmationNumber + " - " +
ccTxnResponse.code + " - " + ccTxnResponse.decision + " - " +
    ccTxnResponse.description;
responseTxt += Environment.NewLine;
responseTxt += "Details:" + Environment.NewLine;
if (ccTxnResponse.detail != null)
{
    for (int i = 0; i < ccTxnResponse.detail.Length; i++)
    {
        responseTxt += " - " + ccTxnResponse.detail[i].tag + " - " +
            ccTxnResponse.detail[i].value + Environment.NewLine;
    }
}
responseTxt = responseTxt.Replace("\n", Environment.NewLine);
System.Console.WriteLine(responseTxt);
consoleTextBox.Text = responseTxt;

if (DecisionV1.ACCEPTED.Equals(ccTxnResponse.decision))
{
    System.Console.WriteLine("Transaction Successfull.");
}
```

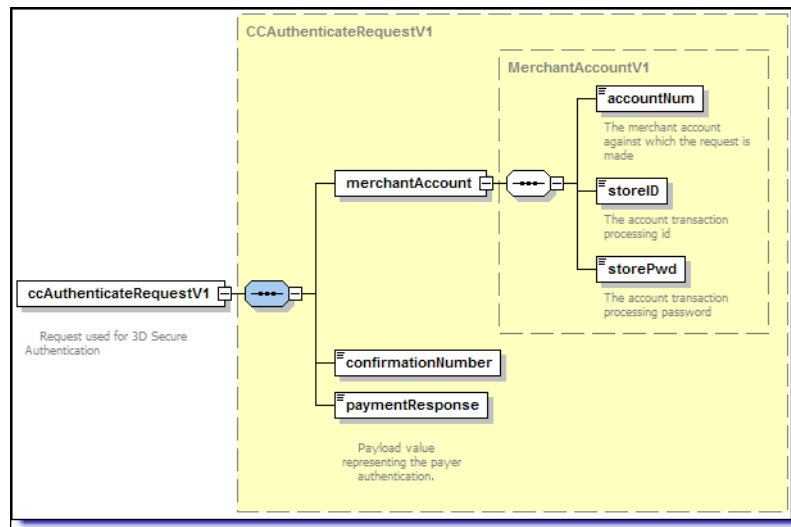
```

else
{
    System.Console.WriteLine("Transaction Failed with decision: " +
        ccTxnResponse.decision);
}

```

ccAuthenticateRequestV1 schema

A *ccAuthenticateRequestV1* has the following structure:



ccAuthenticateRequestV1 elements

The *ccAuthenticateRequestV1* document object may contain the following elements:

Table 3-7: ccAuthenticateRequestV1 Elements

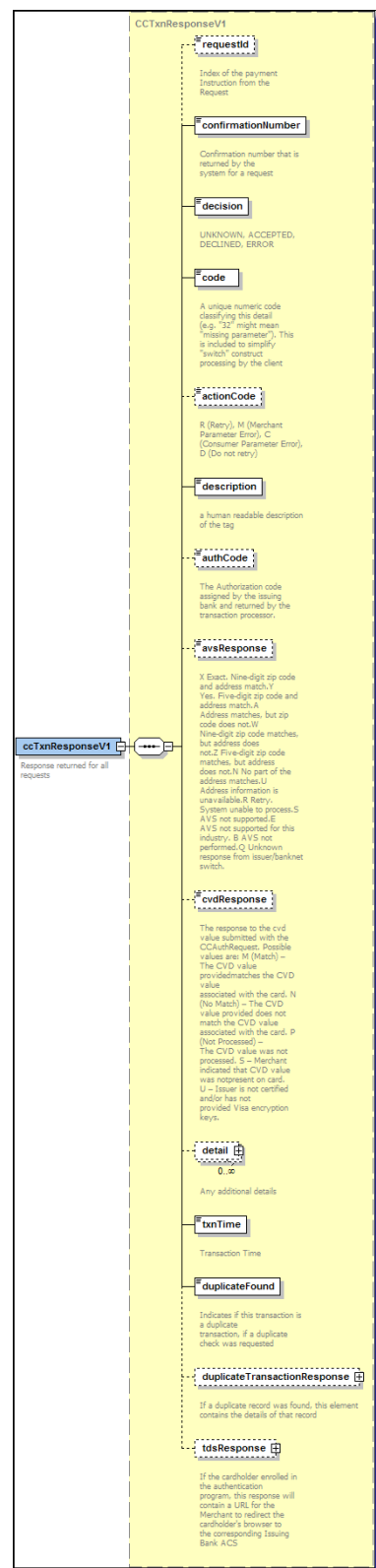
Element	Child Element	Required	Type	Description
merchantAccount	accountNum	Yes	String Max = 10	This is the merchant account number.
	storeID	Yes	String Max = 20	This is the transaction processing store identifier, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
	storePwd	Yes	String Max = 20	This is the transaction processing store password, used to authenticate the request. It is defined by Optimal Payments and provided to the merchant as part of the integration process.
confirmationNumber		Yes	String Max = 20	This is the confirmation number in the <i>ccTxnResponseV1</i> returned by the payment processor in response to the cardholder enrollment lookup.

Table 3-7: ccAuthenticateRequestV1 Elements (Continued)

Element	Child Element	Required	Type	Description
paymentResponse		Yes	String	This is the Payment Authentication Response that is returned from the issuing bank via your customer's Web browser once your customer has provided their authentication information. It is an encoded response generated by the Issuer ACS software. Its digital signature will be verified through Optimal Payments to ensure it was generated by a legitimate Issuer.

Processing the response

A *ccTxnResponseV1* has the following structure:



The following elements are relevant for a *ccTxnResponseV1*:

Table 3-8: ccTxnResponseV1 Elements

Element	Child Element	Required	Type	Description
confirmationNumber		Yes	String Max = 20	This is the confirmation number returned by Optimal Payments.
decision		Yes	Enumeration	This is the status of the transaction. One of the following is returned: <ul style="list-style-type: none"> Accepted – the transaction was processed. Error – the transaction was attempted, but failed for some reason. Declined – the transaction was declined before it was sent for processing.
code		Yes	Int	This is a numeric code that categorizes the response. See <i>Response codes</i> on page B-1.
actionCode		Optional	Enumeration	This indicates what action the caller should take. Possible values are: <ul style="list-style-type: none"> C = Consumer Parameter Error. The consumer has provided incorrect information. Ask the customer to correct the information. D = Do Not Retry. Further attempts will fail. M = Merchant Parameter Error. Your application has provided incorrect information. Verify your information. R = Retry. The problem is temporary. Retrying the request will likely succeed.
description		Yes	String Max = 1024	This is a human readable description of the <i>code</i> element.
authCode		Optional	String Max = 20	This is the Authorization code assigned by the issuing bank and returned by the transaction processor.

Table 3-8: ccTxnResponseV1 Elements (Continued)

Element	Child Element	Required	Type	Description
avsResponse		Optional	Enumeration	<p>This is the AVS response from the card issuer. Possible values are:</p> <ul style="list-style-type: none"> • X – Exact. Nine-digit zip code and address match. • Y – Yes. Five-digit zip code and address match. • A – Address matches, but zip code does not. • W – Nine-digit zip code matches, but address does not. • Z – Five-digit zip code matches, but address does not. • N – No part of the address matches. • U – Address information is unavailable. • R – Retry. System unable to process. • S – AVS not supported. • E – AVS not supported for this industry. • B – AVS not performed. • Q – Unknown response from issuer/banknet switch.
cvdResponse		Optional	Enumeration	<p>This is the response to the <i>cvdValue</i> submitted with the transaction request. Possible values are:</p> <ul style="list-style-type: none"> • M (Match) – The CVD value provided matches the CVD value associated with the card. • N (No Match) – The CVD value provided does not match the CVD value associated with the card. • P (Not Processed) – The CVD value was not processed. • Q (Unknown Response) – No results were received concerning the CVD value. • S (Not Present) – CVD should be on the card. However, the cardholder indicated it was not present. • U – Issuer is not certified and/or has not provided Visa encryption keys.
detail	tag	Optional	String Max = 30	This is the classification of the <i>detail</i> element.
	value	Optional	String Max = 1024	This is the description of the detail.
txnTime		Yes	dateTime	This is the date and time the transaction was processed by the payment processor.

Table 3-8: ccTxnResponseV1 Elements (Continued)

Element	Child Element	Required	Type	Description
duplicateFound		Yes	boolean	This indicates if this transaction is a duplicate transaction, if a duplicate check was requested.
duplicateTransactionResponse		Optional	CCTxnResponse V1	If a duplicate record was found, this element contains the details of that record.
tdsResponse	acsURL	Optional	String Max = 255	This is a fully qualified URL to redirect the consumer to complete the Payment Authentication Request transaction. It is returned only when the <i>cardRiskService</i> element is set to the value of <i>tds</i> .
	paymentRequest	Optional	String	This is an encoded Payment Authentication Request generated by the merchant authentication processing system (MAPS). It is returned only when the <i>cardRiskService</i> element is included with the value of <i>tds</i> .

To process the response:

1. Get the response details, which are available via `get()` methods of the response.

```
String responseTxt = ccTxnResponse.code + " - " + ccTxnResponse.decision +
    " - " + ccTxnResponse.description ;
responseTxt += "Details:" + Environment.NewLine;
if (ccTxnResponse.detail != null)
{
    for (int i = 0; i < ccTxnResponse.detail.Length; i++)
    {
        responseTxt += " - " + ccTxnResponse.detail[i].tag + " - " +
            ccTxnResponse.detail[i].value + Environment.NewLine;
    }
}
responseTxt = responseTxt.Replace("\n", Environment.NewLine);
System.Console.WriteLine(responseTxt);

if (DecisionV1.ACCEPTED.Equals(ccTxnResponse.decision))
{
    System.Console.WriteLine("Transaction Successful.");
}
else
{
    System.Console.WriteLine("Transaction Failed with decision: " +
        ccTxnResponse.decision);
}
```

2. Process based on the *decision* element. You would insert handling code appropriate to your application. You can also look at the *code* element to provide more fine-grained control for your application. See *Response codes* on page B-1 for more details.

Using the HTTP Post Method

Introduction

In addition to a standard Web Service–based call, you can also use the HTTP Post method to post Direct Debit and credit card transaction requests to Optimal Payments.

Note that the URLs in the examples below point to the Optimal Payments Production environment. In order to test your integration, please contact Technical Support to obtain Test URLs and Test merchant account parameters.

- Email support@optimalpayments.com
- Telephone 1-888-709-8753

Direct Debit requests

charge/verify/credit

To send a charge, verify, or credit request via HTTP Post:

1. Create a transaction request like the following example:

```
<ddCheckRequestV1 xmlns="http://www.optimalpayments.com/directdebit/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <merchantRefNum>Ref-12345</merchantRefNum>
  <amount>10.00</amount>
  <check>
    <accountType>PC</accountType>
    <bankName>Chase</bankName>
    <checkNum>12</checkNum>
    <accountNum>987654321</accountNum>
    <routingNum>123456789</routingNum>
  </check>
  <billingDetails>
    <checkPayMethod>WEB</checkPayMethod>
    <firstName>Jane</firstName>
    <lastName>Jones</lastName>
    <street>123 Main Street</street>
    <city>LA</city>
    <state>CA</state>
    <country>US</country>
    <zip>90210</zip>
    <phone>555-555-5555</phone>
    <email>janejones@emailserver.com</email>
  </billingDetails>
</ddCheckRequestV1>
```

```

<sdk>
  <version>1.0</version>
  platform>http</platform>
  <provider>Merchant</provider>
</sdk>
</ddCheckRequestV1>

```



See Table 2-2: ddCheckRequestV1 Elements on page 2-6 for a list and description of parameters to include in this request.

2. Include this transaction request in an HTTP Post (see *HTML example – charge* on page A-2).

This HTTP Post must include two parameters:

- **txnMode** – charge, credit, or verify
- **txnRequest** – the transaction request above

3. Send the HTTP Post to the following URL:

<https://webservices.optimalpayments.com/directdebitWS/DirectDebitServlet/v1>

HTML example – charge

This example shows a form version of a *charge/verify/credit* request – containing the example above – that you could post to Optimal Payments.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<title>Optimal Payments - Direct Debit Test</title>
<body>
<form NAME="Direct Debit" METHOD="post"

ACTION="https://webservices.optimalpayments.com/directdebitWS/DirectDebitServlet/v1"
>
<b>Transaction Mode: </b>
<br>
<select name="txnMode">
<option value="charge">Charge</option>
<option value="credit">Credit</option>
<option value="verify">Verify</option>
</select>
<br>
<br>
<b>XML Message body:</b>
<TEXTAREA class="xmlbox" name="txnRequest" COLS=100 ROWS=10>
<ddCheckRequestV1 xmlns="http://www.optimalpayments.com/directdebit/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <merchantRefNum>Ref-12345</merchantRefNum>
  <amount>10.00</amount>
  <check>
    <accountType>PC</accountType>
    <bankName>Chase</bankName>
    <checkNum>12</checkNum>
    <accountNum>987654321</accountNum>
  </check>
</ddCheckRequestV1>
</TEXTAREA>

```

```

        <routingNum>123456789</routingNum>
    </check>
    <billingDetails>
        <checkPayMethod>WEB</checkPayMethod>
        <firstName>Jane</firstName>
        <lastName>Jones</lastName>
        <street>123 Main Street</street>
        <city>LA</city>
        <state>CA</state>
        <country>US</country>
        <zip>90210</zip>
        <phone>555-555-5555</phone>
        <email>janejones@emailserver.com</email>
    </billingDetails>
    <sdk>
        <version>1.0</version>
        <platform>http</platform>
        <provider>Merchant</provider>
    </sdk>
</ddCheckRequestV1>
</TEXTAREA> <br>
<input TYPE=submit class=input VALUE="Send Request"></form>
</body>
</html>

```



The maximum file size supported is 100K. If your HTTP Post exceeds 100K it will fail.

See *Sample HTTP Post* on page A-16 to see what this would look like viewed with a browser. Clicking the Send Request button sends the transaction via HTTP Post to Optimal Payments.

updateShippingInfo

To send an updateShippingInfo request via HTTP Post:

1. Create a transaction request like the following example:

```
<ddShippingRequestV1
xmlns="http://www.optimalpayments.com/directdebit/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <carrier>FEX</carrier>
  <trackingNumber>555666888999</trackingNumber>
  <confirmationNumber>123456</confirmationNumber>
  <shipMethod>T</shipMethod>
  <firstName>Jane</firstName>
  <lastName>Jones</lastName>
  <street>123 Main Street</street>
  <city>LA</city>
  <state>CA</state>
  <country>US</country>
  <zip>90210</zip>
  <phone>555-555-5555</phone>
  <email>janejones@emailserver.com</email>
</ddShippingRequestV1>
```



See Table 2-3: ddShippingRequestV1 Elements on page 2-15 for a list and description of parameters to include in this request.

2. Include this transaction request in an HTTP Post (see *HTML example – updateShippingInfo* on page A-4).

This HTTP Post must include two parameters:

- **txnMode** – updateShippingInfo
- **txnRequest** – the transaction request above

3. Send the HTTP Post to the following URL:

<https://webservices.optimalpayments.com/directdebitWS/DirectDebitServlet/v1>

HTML example – updateShippingInfo

This example shows a form version of an *updateShippingInfo* request – containing the example above – that you could post to Optimal Payments.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<title>Direct Debit Test</title>
<body>
<form NAME="Direct Debit" METHOD="post"

ACTION="https://webservices.optimalpayments.com/directdebitWS/DirectDebitServlet/v1"
>
<input type=hidden name="txnMode" value="updateShippingInfo" >
```

```

<br>
<b>XML Message body:</b>
<TEXTAREA class="xmlbox" name="txnRequest" COLS=100 ROWS=10>

<ddShippingRequestV1
xmlns="http://www.optimalpayments.com/directdebit/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <carrier>FEX</carrier>
  <trackingNumber>555666888999</trackingNumber>
  <confirmationNumber>123456</confirmationNumber>
  <shipMethod>T</shipMethod>
  <firstName>Jane</firstName>
  <lastName>Jones</lastName>
  <street>123 Main Street</street>
  <city>LA</city>
  <state>CA</state>
  <country>US</country>
  <zip>90210</zip>
  <phone>555-555-5555</phone>
  <email>janejones@emailserver.com</email>
</ddShippingRequestV1>
</TEXTAREA> <br>
<input TYPE=submit class=input VALUE="Send Request"></form>
</body>
</html>

```



The maximum file size supported is 100K. If your HTTP Post exceeds 100K it will fail.

See *Sample HTTP Post* on page A-16 to see what this would look like viewed with a browser. Clicking the Send Request button sends the transaction via HTTP Post to Optimal Payments.

Credit card requests

Purchase/Authorization/Verification

To send a credit card Purchase/Authorization/Verification transaction request via HTTP Post:

1. Create a transaction request like the following example:

```

<ccAuthRequestV1 xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <merchantRefNum>Ref-12345</merchantRefNum>
  <amount>10.00</amount>

```

```

<card>
  <cardNum>4653111111111111</cardNum>
  <cardExpiry>
    <month>11</month>
    <year>2008</year>
  </cardExpiry>
  <cardType>VI</cardType>
  <cvdIndicator>1</cvdIndicator>
  <cvd>111</cvd>
</card>
<billingDetails>
  <cardPayMethod>WEB</cardPayMethod>
  <firstName>Jane</firstName>
  <lastName>Jones</lastName>
  <street>123 Main Street</street>
  <city>LA</city>
  <state>CA</state>
  <country>US</country>
  <zip>90210</zip>
  <phone>555-555-5555</phone>
  <email>janejones@emailserver.com</email>
</billingDetails>
<shippingDetails>
  <carrier>FEX</carrier>
  <shipMethod>T</shipMethod>
  <firstName>Jane</firstName>
  <lastName>Jones</lastName>
  <street>44 Main Street</street>
  <city>LA</city>
  <state>CA</state>
  <country>US</country>
  <zip>90210</zip>
  <phone>555-555-5555</phone>
  <email>janejones@emailserver.com</email>
</shippingDetails>
<previousCustomer>true</previousCustomer>
<customerIP>127.0.0.1</customerIP>
<productType>M</productType>
<cardRiskService>TDS</cardRiskService>
<addendumData>
  <tag>CUST_ACCT_OPEN_DATE</tag>
  <value>20041012</value>
</addendumData>
<addendumData>
  <tag>MERCHANT_COUNTRY_CODE</tag>
  <value>US</value>
</addendumData>
<addendumData>
  <tag>MERCHANT_ZIP_CODE</tag>
  <value>90210</value>
</addendumData>
</ccAuthRequestV1>

```



See Table 3-2: ccAuthRequestV1 Elements on page 3-8 for a list and description of parameters to include in this request.

2. Include this transaction request in an HTTP Post (see *HTML example – Authorization* on page A-7).

This HTTP Post must include two parameters:

- **txnMode** – ccAuthorize, ccPurchase, or ccVerification
- **txnRequest** – the transaction request above

3. Send the HTTP Post to the following URL:

<https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1>

HTML example – Authorization

This example shows a form version of a credit card *Authorization* request – containing the example above – that you could post to Optimal Payments.



To make this a Purchase or Verification request, just modify the txnMode value “ccAuthorize” (underlined below) to “ccPurchase” or “ccVerification”, respectively.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<title>Credit Card Test</title>
<body>
<form NAME="Credit Card" METHOD="post"
ACTION="https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1">
<input type="hidden" name="txnMode" value="ccAuthorize" >
<b>XML Message body:</b>
<TEXTAREA class="xmlbox" name="txnRequest" COLS=100 ROWS=10 >
<ccAuthRequestV1 xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <merchantRefNum>Ref-12345</merchantRefNum>
  <amount>10.00</amount>
  <card>
    <cardNum>4653111111111111</cardNum>
    <cardExpiry>
      <month>11</month>
      <year>2008</year>
    </cardExpiry>
    <cardType>VI</cardType>
    <cvdIndicator>1</cvdIndicator>
    <cvd>111</cvd>
  </card>
  <billingDetails>
    <cardPayMethod>WEB</cardPayMethod>
    <firstName>Jane</firstName>
    <lastName>Jones</lastName>
    <street>123 Main Street</street>
    <city>LA</city>
    <state>CA</state>
    <country>US</country>
```

```

    <zip>90210</zip>
    <phone>555-555-5555</phone>
    <email>janejones@emailserver.com</email>
  </billingDetails>
  <shippingDetails>
    <carrier>FEX</carrier>
    <shipMethod>T</shipMethod>
    <firstName>Jane</firstName>
    <lastName>Jones</lastName>
    <street>44 Main Street</street>
    <city>LA</city>
    <state>CA</state>
    <country>US</country>
    <zip>90210</zip>
    <phone>555-555-5555</phone>
    <email>janejones@emailserver.com</email>
  </shippingDetails>
  <previousCustomer>true</previousCustomer>
  <customerIP>127.0.0.1</customerIP>
  <productType>M</productType>
  <cardRiskService>TDS</cardRiskService>
  <addendumData>
    <tag>CUST_ACCT_OPEN_DATE</tag>
    <value>20041012</value>
  </addendumData>
  <addendumData>
    <tag>MERCHANT_COUNTRY_CODE</tag>
    <value>US</value>
  </addendumData>
  <addendumData>
    <tag>MERCHANT_ZIP_CODE</tag>
    <value>90210</value>
  </addendumData>
</ccAuthRequestV1>
</TEXTAREA>
<br>
<input TYPE=submit class=input VALUE="Send Request">
</form>
</body>
</html>

```



The maximum file size supported is 100K. If your HTTP Post exceeds 100K it will fail.

See *Sample HTTP Post* on page A-16 to see what this would look like viewed with a browser. Clicking the Send Request button sends the transaction via HTTP Post to Optimal Payments.

Settlement/Credit

To send a credit card Settlement/Credit transaction request via HTTP Post:

1. Create a transaction request like the following example:

```

<ccPostAuthRequestV1 xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>

```

```

    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <confirmationNumber>123456</confirmationNumber>
  <merchantRefNum>Ref-12345</merchantRefNum>
  <amount>10.00</amount>
  <addendumData>
    <tag>MERCHANT_DATA</tag>
    <value>MERCHANT_DATA</value>
  </addendumData>
</ccPostAuthRequestV1>

```



See Table 3-3: ccPostAuthRequestV1 Elements on page 3-16 for a list and description of parameters to include in this request.

2. Include this transaction request in an HTTP Post (see *HTML example – Settlement* on page A-9).

This HTTP Post must include two parameters:

- **txnMode** – ccSettlement or ccCredit
- **txnRequest** – the transaction request above

3. Send the HTTP Post to the following URL:

<https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1>

HTML example – Settlement

This example shows a form version of a credit card *Settlement* request – containing the example above – that you could post to Optimal Payments.



To make this a Credit request, just modify the txnMode value “ccSettlement” (underlined below) to “ccCredit”.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<title>Credit Card Test</title>
<body>
<form NAME="Credit Card" METHOD="post"
ACTION="https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1">
<input type="hidden" name="txnMode" value="ccSettlement" >
<b>XML Message body:</b>
<TEXTAREA class="xmlbox" name="txnRequest" COLS=100 ROWS=10 >
<ccPostAuthRequestV1 xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <confirmationNumber>123456</confirmationNumber>
  <merchantRefNum>Ref-12345</merchantRefNum>
  <amount>10.00</amount>

```

```

    <addendumData>
      <tag>MERCHANT_DATA</tag>
      <value>MERCHANT_DATA</value>
    </addendumData>
  </ccPostAuthRequestV1>
</TEXTAREA>
<br>
<input TYPE=submit class=input VALUE="Send Request">
</form>
</body>
</html>

```



The maximum file size supported is 100K. If your HTTP Post exceeds 100K it will fail.

See *Sample HTTP Post* on page A-16 to see what this would look like viewed with a browser. Clicking the Send Request button sends the transaction via HTTP Post to Optimal Payments.

Stored Data Authorization/Purchase

To send a Stored Data Authorization/Purchase transaction request via HTTP Post:

1. Create a transaction request like the following example:

```

<ccStoredDataRequestV1
xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <merchantRefNum>VALUE</merchantRefNum>
  <confirmationNumber>123456</confirmationNumber>
  <amount>18.00</amount>
</ccStoredDataRequestV1>

```



See Table 3-4: ccStoredDataRequestV1 Elements on page 3-19 or a list and description of parameters to include in this request.

2. Include this transaction request in an HTTP Post (see *HTML example – Stored Data Authorization* on page A-11).

This HTTP Post must include two parameters:

- **txnMode** – ccStoredDataAuthorize or ccStoredDataPurchase
- **txnRequest** – the transaction request above

3. Send the HTTP Post to the following URL:

<https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1>

HTML example – Stored Data Authorization

This example shows a form version of a credit card *Stored Data Authorization* request – containing the example above – that you could post to Optimal Payments.



To make this a *Stored Data Purchase* request, just modify the *txnMode* value "ccStoredDataAuthorize" (underlined below) to "ccStoredDataPurchase".

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<title>Credit Card Test</title>
<body>
<form NAME="Credit Card" METHOD="post"
ACTION="https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1">
<input type=hidden name="txnMode" value="ccStoredDataAuthorize" >
<b>XML Message body:</b>
<TEXTAREA class="xmlbox" name="txnRequest" COLS=100 ROWS=10 >
<ccStoredDataRequestV1
xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <merchantRefNum>VALUE</merchantRefNum>
  <confirmationNumber>123456</confirmationNumber>
  <amount>18.00</amount>
</ccStoredDataRequestV1>
</TEXTAREA>
<br>
<input TYPE=submit class=input VALUE="Send Request">
</form>
</body>
</html>
```



The maximum file size supported is 100K. If your HTTP Post exceeds 100K it will fail.

See *Sample HTTP Post* on page A-16 to see what this would look like viewed with a browser. Clicking the Send Request button sends the transaction via HTTP Post to Optimal Payments.

Cancel Settle/Credit/Payment

To cancel a credit card Settle/Credit/Payment request via HTTP Post:

1. Create a transaction request like the following example:

```
<ccCancelRequestV1 xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
```

```

    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <confirmationNumber>123456</confirmationNumber>
  <addendumData>
    <tag>MERCHANT_DATA</tag>
    <value>MERCHANT_DATA</value>
  </addendumData>
</ccCancelRequestV1>

```



See Table 3-5: ccCancelRequestV1 Elements on page 3-23 for a list and description of parameters to include in this request.

2. Include this transaction request in an HTTP Post (see *HTML example – Cancel* on page A-12).

This HTTP Post must include two parameters:

- **txnMode** – ccCancelSettle, ccCancelCredit, or ccCancelPayment
- **txnRequest** – the transaction request above

3. Send the HTTP Post to the following URL:

<https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1>

HTML example – Cancel

This example shows a form version of a credit card *Cancel* request – containing the example above – that you could post to Optimal Payments to cancel a *Settle* transaction.



To make this a *Cancel Credit* or *Cancel Payment* request, just modify the *txnMode* value “ccCancelSettle” (underlined below) to “ccCancelCredit” or “ccCancelPayment”, respectively.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<title>Credit Card Test</title>
<body>
<form NAME="Credit Card" METHOD="post"
ACTION="https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1">
<input type="hidden" name="txnMode" value="ccCancelSettle" >
<b>XML Message body:</b>
<TEXTAREA class="xmlbox" name="txnRequest" COLS=100 ROWS=10 >
<ccCancelRequestV1 xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <confirmationNumber>123456</confirmationNumber>
  <addendumData>
    <tag>MERCHANT_DATA</tag>
    <value>MERCHANT_DATA</value>
  </addendumData>
</ccCancelRequestV1>
</TEXTAREA>
<br>

```

```
<input TYPE=submit class=input VALUE="Send Request">
</form>
</body>
</html>
```



The maximum file size supported is 100K. If your HTTP Post exceeds 100K it will fail.

See *Sample HTTP Post* on page A-16 to see what this would look like viewed with a browser. Clicking the Send Request button sends the transaction via HTTP Post to Optimal Payments.

Payment request

To make a Payment request via HTTP Post:

1. Create a transaction request like the following example:

```
<ccPaymentRequestV1 xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <merchantRefNum>Ref-12345</merchantRefNum>
  <amount>10.00</amount>
  <card>
    <cardNum>4653111111111111</cardNum>
    <cardExpiry>
      <month>11</month>
      <year>2008</year>
    </cardExpiry>
    <cardType>VI</cardType>
    <cvdIndicator>1</cvdIndicator>
    <cvd>111</cvd>
  </card>
  <billingDetails>
    <cardPayMethod>WEB</cardPayMethod>
    <firstName>jane</firstName>
    <lastName>jones</lastName>
    <street>123 Main Street</street>
    <city>LA</city>
    <state>CA</state>
    <country>US</country>
    <zip>90210</zip>
    <phone>555-555-5555</phone>
    <email>janejones@emailserver.com</email>
  </billingDetails>
</ccPaymentRequestV1>
```



See Table 3-6: ccPaymentRequestV1 Elements on page 3-26 for a list and description of parameters to include in this request.

2. Include this transaction request in an HTTP Post (see *HTML example – Payment* on page A-14).

This HTTP Post must include two parameters:

- **txnMode** – ccPayment
- **txnRequest** – the transaction request above

3. Send the HTTP Post to the following URL:

<https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1>

HTML example – Payment

This example shows a form version of a credit card *Payment* request – containing the example above – that you could post to Optimal Payments.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>CC Payment</title>
</head>

<body>
<h1> Credit Card Webservice Tester </h1>
<form NAME="creditcard" id="creditcard" METHOD="post"
ACTION="https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1">
<input type="hidden" name="txnMode" value="ccPayment" >
<b>XML Message body:</b>
<TEXTAREA class="xmlbox" name="txnRequest" COLS=100 ROWS=10 >
<ccPaymentRequestV1 xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <merchantRefNum>Ref-12345</merchantRefNum>
  <amount>10.00</amount>
  <card>
    <cardNum>4653111111111111</cardNum>
    <cardExpiry>
      <month>11</month>
      <year>2008</year>
    </cardExpiry>
    <cardType>VI</cardType>
    <cvdIndicator>1</cvdIndicator>
    <cvd>111</cvd>
  </card>
  <billingDetails>
    <cardPayMethod>WEB</cardPayMethod>
    <firstName>jane</firstName>
    <lastName>jones</lastName>
    <street>123 Main Street</street>
    <city>LA</city>
    <state>CA</state>
    <country>US</country>
    <zip>90210</zip>
    <phone>555-555-5555</phone>
    <email>janejones@emailserver.com</email>
  </billingDetails>
</ccPaymentRequestV1>
</TEXTAREA>
</form>
</body>
</html>
```



```

</ccPaymentRequestV1>
</TEXTAREA>
<br>
<input TYPE=submit class=input VALUE="Send Request">
</form>
</body>
</html>

```



The maximum file size supported is 100K. If your HTTP Post exceeds 100K it will fail.

See *Sample HTTP Post* on page A-16 to see what this would look like viewed with a browser. Clicking the Send Request button sends the transaction via HTTP Post to Optimal Payments.

Authentication

To send a credit card Authentication transaction request via HTTP Post:

1. Create a transaction request like the following example:

```

<ccAuthenticateRequestV1
xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <confirmationNumber>jonathan-test</confirmationNumber>
  <paymentResponse>myPaymentResponse</paymentResponse>
</ccAuthenticateRequestV1>

```



See Table 3-7: ccAuthenticateRequestV1 Elements on page 3-31 for a list and description of parameters to include in this request.

2. Include this transaction request in an HTTP Post (see *HTML example – Authentication* on page A-15).

This HTTP Post must include two parameters:

- **txnMode** – ccAuthenticate
- **txnRequest** – the transaction request above

3. Send the HTTP Post to the following URL:

<https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1>

HTML example – Authentication

This example shows a form version of a credit card *Authentication* request – containing the example above – that you could post to Optimal Payments.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<title>Credit Card Test</title>

```

```

<body>
<form NAME="Credit Card" METHOD="post"
ACTION="https://webservices.optimalpayments.com/creditcardWS/CreditCardServlet/v1">
<input type=hidden name="txnMode" value="ccAuthenticate" >
<b>XML Message body:</b>
<TEXTAREA class="xmlbox" name="txnRequest" COLS=100 ROWS=10 >
<ccAuthenticateRequestV1
xmlns="http://www.optimalpayments.com/creditcard/xmlschema/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.optimalpayments.com/creditcard/xmlschema/v1">
  <merchantAccount>
    <accountNum>12345678</accountNum>
    <storeID>myStoreID</storeID>
    <storePwd>myStorePWD</storePwd>
  </merchantAccount>
  <confirmationNumber>jonathan-test</confirmationNumber>
  <paymentResponse>myPaymentResponse</paymentResponse>
</ccAuthenticateRequestV1>
</TEXTAREA>
<br>
<input TYPE=submit class=input VALUE="Send Request">
</form>
</body>
</html>

```

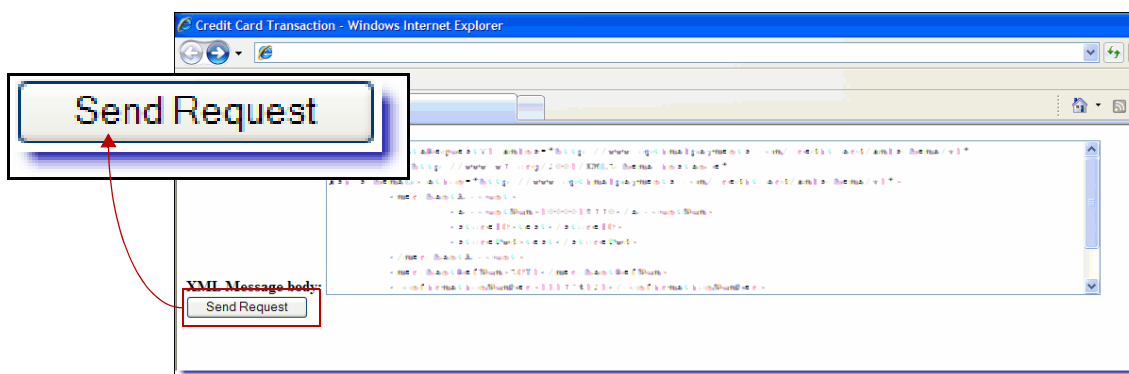


The maximum file size supported is 100K. If your HTTP Post exceeds 100K it will fail.

See *Sample HTTP Post* on page A-16 to see what this would look like viewed with a browser. Clicking the Send Request button sends the transaction via HTTP Post to Optimal Payments.

Sample HTTP Post

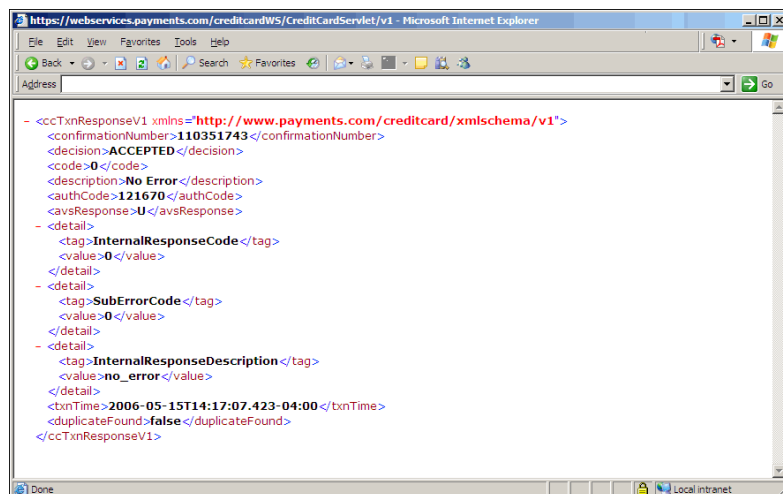
This is what the HTTP Post examples of the transaction types described in this chapter would look like viewed in a browser:



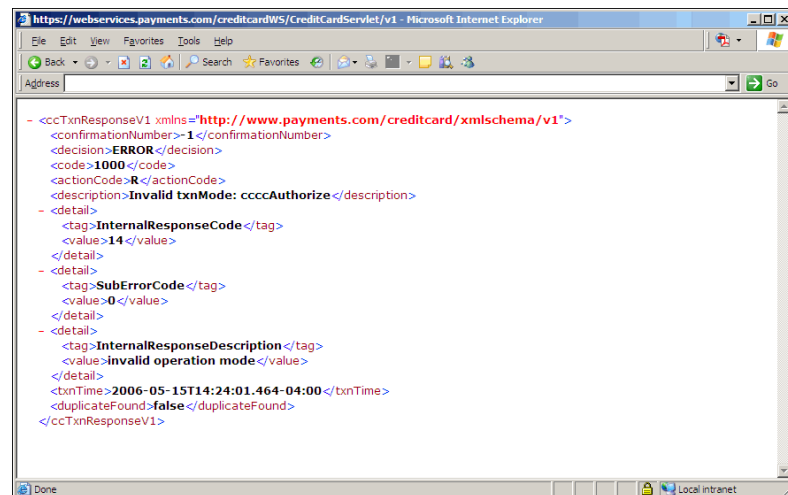
Clicking the Send Request button sends the transaction via HTTP Post to Optimal Payments.

Sample HTTP Post responses

Optimal Payments returns a response like the following for a successful transaction request:



Optimal Payments returns a response like the following for a failed transaction request:



For an explanation of the elements in the response, see *Processing the response* on page 3-33.
For an explanation of error codes, see *Response codes* on page B-1.

Response Codes

Overview

Optimal Payments Web Services can return different types of code if a transaction attempt fails:

- Response codes
- Action codes
- Return codes

Response codes

The following table describes the response codes that could be returned by Optimal Payments Web Services.

Table B-1: Response Codes

Response Code	Action	Description
1000	R	An internal error occurred. Please retry the transaction.
1001	R	An error occurred with the external processing gateway. Please retry the transaction.
1002	R	An internal error occurred. Please retry the transaction.
1003	D	An error occurred with the external processing gateway. Do not retry the transaction. Contact Technical Support for more information.
1004	M	Your account is not enabled for this transaction type. Please verify your parameters and retry the transaction.
1005	M	You submitted an invalid FirePay account number with your request. Please verify this parameter and retry the transaction.
1006	D	An error occurred with the external processing gateway. Do not retry the transaction. Contact Technical Support for more information.
1007	R	An internal error occurred. Please retry the transaction.
2000	D	The echeck status cannot be found.
2001	C	You submitted a request containing a check number already used within the last 24 hours. Please use another Check Number value and retry the transaction.
2002	M	The echeck request cannot be found. Please verify your parameters and retry.
2003	M	The payment type you provided conflicts with the bank account type you provided. Please verify your parameters and retry the transaction.
2004	D	The echeck transaction cannot be found.
2005	M	An internal error occurred. Please verify your parameters and retry the transaction.
2006	C	You have submitted a Decline transaction in response to a Settlement attempt.
2007	M	The payment type included with your request cannot be used in Credit transaction mode. Please verify your parameters and retry the request.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
2008	C	You have submitted an invalid routing number. Please verify your parameter and retry the transaction.
2009	C	You have submitted an invalid bank account number. Please verify your parameter and retry the transaction.
2010	C	You have submitted an invalid check number. Please verify your parameter and retry the transaction.
3001	C	You submitted an unsupported card type with your request. Please verify this parameter and retry the transaction.
3002	C	You submitted an invalid card number, brand, or combination of card number and brand with your request. Please verify these parameters and retry the transaction.
3003	C	You submitted an incorrect value for the cvdIndicator parameter with your request. Please verify this parameter and retry the transaction.
3004	C	You have requested an AVS check. Please ensure that the zip/postal code is provided.
3005	C	You submitted an incorrect value for the cvd parameter with your request. Please verify this parameter and retry the transaction.
3006	C	You submitted an expired credit card number with your request. Please verify this parameter and retry the request.
3007	D	Your request has failed the AVS check. Note that the amount has still been reserved on the customer's card and will be released in 3-5 business days. Please ensure the billing address is accurate before retrying the transaction.
3008	D	You submitted a card type for which the merchant account is not configured.
3009	D	Your request has been declined by the issuing bank.
3010	D	The consumer is attempting a purchase for more than the balance available in their FirePay account.
3011	D	You submitted an inactive FirePay account number with your request.
3012	D	Your request has been declined by the issuing bank because the credit card expiry date submitted is invalid.
3013	D	Your request has been declined by the issuing bank due to problems with the credit card account.
3014	D	Your request has been declined – the issuing bank has returned an unknown response. Contact the cardholder's credit card company for further investigation.
3015	D	The bank has requested that you process the transaction manually by calling the cardholder's credit card company.
3016	D	The bank has requested that you retrieve the card from the cardholder – it may be a lost or stolen card.
3017	C	You submitted an invalid credit card number with your request. Please verify this parameter and retry the transaction.
3018	R	The bank has requested that you retry the transaction.
3019	C	Your request has failed the CVD check. Please note that the amount has still been reserved on the customer's card and will be released in 3–5 business days. Please ensure the CVD value is accurate before retrying the transaction.
3020	R	The bank has requested that you retry the transaction.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
3021	M	The confirmation number included in this request could not be found. Please verify this parameter and retry the transaction.
3022	D	The card has been declined due to insufficient funds.
3023	D	Your request has been declined by the issuing bank due to its proprietary card activity regulations.
3024	D	Your request has been declined because the issuing bank does not permit the transaction for this card.
3200	M	You have submitted an invalidly formatted authorization ID for this settlement. Please verify this parameter and retry the transaction.
3201	M	The authorization ID included in this settlement request could not be found. Please verify this parameter and retry the transaction.
3202	M	You have exceeded the maximum number of settlements allowed.
3203	M	The authorization is either fully settled or cancelled.
3204	M	The requested settlement amount exceeds the remaining authorization amount.
3402	M	The requested credit amount exceeds the remaining settlement amount.
3403	M	You have already processed the maximum number of credits allowed for this settlement.
3404	M	The settlement has already been fully credited.
3405	M	The settlement you are attempting to credit has expired.
3406	M	The settlement you are attempting to credit has not been batched yet. There are no settled funds available to credit.
3407	M	The settlement referred to by the transaction response ID you provided cannot be found. Please verify this parameter and retry the transaction.
3408	M	You have submitted an invalidly formatted response ID for the original purchase or settlement. Please verify this parameter and retry the transaction.
3601	D	The 3D Secure authentication of this cardholder by the card issuer failed.
3602	M	The confirmation number included in the 3D Secure authentication request could not be found. The confirmation number must be the one returned by the payment processor in response to the original authorization or purchase.
3603	M	You submitted a request that is not available for 3D Secure authentication.
3604	D	The original authorization or purchase request has expired. The 3D Secure authentication request must be completed within 60 minutes of the original authorization or purchase.
4001	D	The card number or email address associated with this transaction is in our negative database.
4002	D	The transaction was declined by our Risk Management department.
5000	M	Your merchant account authentication failed. Either your store ID/password are invalid or the IP address from which you are sending the transaction has not been authorized. Please verify these parameters and retry the transaction. If the error persists, please contact Technical Support.
5001	M	You submitted an invalid currency code with your request. Please verify this parameter and retry the transaction.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
5002	M	You submitted an invalid payment method with your request. Please verify this parameter and retry the transaction.
5003	C	You submitted an invalid amount with your request. Please verify this parameter and retry the transaction.
5004	M	You submitted an invalid account type with your request. Please verify this parameter and retry the transaction.
5005	M	You submitted an invalid operation type with your request. Please verify this parameter and retry the transaction.
5006	M	You submitted an invalid personal ID type with your request. Please verify this parameter and retry the transaction.
5007	M	You submitted an invalid product type with your request. Please verify this parameter and retry the transaction.
5008	M	You submitted an invalid carrier with your request. Please verify this parameter and retry the transaction.
5009	M	You submitted an invalid ship method with your request. Please verify this parameter and retry the transaction.
5010	M	You submitted an invalid ID country with your request. Please verify this parameter and retry the transaction.
5011	M	You submitted an invalid order date & time with your request. Please verify this parameter and retry the transaction.
5012	M	You submitted an invalid ship country parameter with your request. Please verify this parameter and retry the transaction.
5013	M	You submitted an invalid ship state parameter with your request. Please verify this parameter and retry the transaction.
5014	M	You submitted an invalid transaction type with your request. Please verify this parameter and retry the transaction.
5015	M	One or more of the parameters in your request exceeds the maximum permissible field length. Please verify your parameters and retry the transaction.
5016	M	Either you submitted an invalid merchant account or the merchant account could not be found. Please verify this parameter and retry the transaction. If the error persists, please contact Technical Support.
5017	D	The merchant account submitted with your request is not enabled. Do not retry the transaction. Contact Technical Support for more information.
5018	M	You submitted a request that is missing a mandatory field. Please verify your parameters and retry the transaction.
5019	D	There is no processor set up for the merchant account submitted with your request. Do not retry the transaction. Contact Technical Support for more information.
5020	M	The currency type included with your request does not match the currency type of your merchant account. Please verify your parameters and retry the request.
5021	D	Your transaction request has been declined. Please verify the transaction details before you attempt this transaction again. Should you require more information, please contact Technical Support.
5022	M	You submitted a request that is missing search criteria. Please verify your parameters and retry the transaction.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
5023	M	You submitted an invalid XML request. Please verify your request and retry the transaction.
5024	M	You submitted an invalid ID expiry date for this request. Please verify this parameter and retry the transaction.
5025	M	The search criteria you submitted is currently not supported. Please verify your search criteria and retry.
5026	M	The Web Services API does not currently support credit card transactions. Please verify your parameters and retry your transaction.
5027	M	You submitted an invalid risk service name with your transaction request. Please verify your parameters and retry your transaction.
5028	M	You submitted a batch file with the same file name, file size, and number of entries within the last 24 hours.
5029	M	You submitted an improperly formatted batch file.
5030	M	You submitted a batch file that has exceeded the maximum number of rows allowed.
5031	M	The transaction you have submitted has already been processed.
5032	C	You submitted an invalid city with your request. Please verify this parameter and retry the transaction.
5033	C	You submitted an invalid country with your request. Please verify this parameter and retry the transaction.
5034	C	You submitted an invalid email address with your request. Please verify this parameter and retry the transaction.
5035	C	You submitted an invalid name with your request. Please verify this parameter and retry the transaction.
5036	C	You submitted an invalid phone number with your request. Please verify this parameter and retry the transaction.
5037	C	You submitted an invalid zip/postal code with your request. Please verify this parameter and retry the transaction.
5038	C	You submitted an invalid state/province with your request. Please verify this parameter and retry the transaction.
5039	C	You submitted an invalid street with your request. Please verify this parameter and retry the transaction.
5040	D	Your merchant account is not configured for the transaction you attempted. Please contact Technical Support for more information.
5041	M	You submitted an invalid merchantData parameter. Please verify this parameter and retry the transaction.
5042	M	The merchant reference number is missing or invalid, or it exceeds the maximum permissible length. Please verify this parameter and retry the transaction.
5043	M	You submitted an invalid account open date with your request. Please verify this parameter and retry the transaction.
5044	M	You submitted an invalid customer ID with your request. Please verify this parameter and retry the transaction.
5045	M	You submitted an invalid customer IP address with your request. Please verify this parameter and retry the transaction.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
5046	M	You submitted an invalid merchant SIC code with your request. Please verify this parameter and retry the transaction.
5047	M	You submitted an invalid value for the previous customer parameter with your request. Please verify this parameter and retry the transaction.
5048	M	You submitted an invalid product code with your request. Please verify this parameter and retry the transaction.
5049	M	You submitted an invalid value for user data with your request. Please verify this parameter and retry the transaction.
5050	D	An error occurred with your merchant account configuration. Please contact Technical Support.
5051	C	You have submitted an invalid confirmation number. Please verify your parameter and retry.
5052	C	You have submitted an invalid Zip/Postal code. Please verify your parameter and retry the transaction.
5053	C	You have submitted invalid personal ID information. Please verify your parameters and retry the transaction.
5054	M	You cannot submit recurring billing elements with this operation type. Please verify that your request includes the correct operation type and retry the transaction.
5055	M	You submitted an invalid ECI value. Please verify this parameter and retry the transaction.
5056	M	An ECI value must be provided for 3D Secure authentication. Please verify this parameter and retry the transaction.
5057	M	You submitted a credit card brand that does not support 3D Secure authentication. Please verify this parameter and retry the transaction.
5058	M	You submitted an ECI value without including a CAVV value, which must also be provided for a successful 3D Secure authentication. Please verify this parameter and retry the transaction.
6000	M	Could not find op-config.xml file in the classpath. Please verify that the configuration file is included in the classpath.
6001	M	Your merchant account number cannot be found in op-config.xml. Please verify that your merchant account number is included in the configuration file.
6003	M	Cannot connect to server. Please verify that the proxy server is correctly configured. If the error persists, contact Technical Support.
6004	M	A value in op-config.xml is invalid. Please verify that all values included in the configuration file are valid.
6005	M	The server URL value in op-config.xml is invalid. Please ensure that a valid server URL value is included in the configuration file.
6006	M	The server URL value in op-config.xml is missing or unreadable. Please ensure that the correct server URL is included in the configuration file.
6007	M	The timeout value in op-config.xml is missing or unreadable. Please ensure that the correct timeout value is included in the configuration file.
6011	M	You have not selected a valid path to the op-config.xml file.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
6012	M	The file you have selected as the op-config.xml is either not an XML file or not a valid XML file.
6013	M	An error occurred while saving the op-config.xml file.
6014	M	The merchant account number you submitted does not match the merchant account number in the op-config.xml file. Please ensure that the merchant account number submitted with transaction requests matches the merchant account number in the configuration file.
6016	M	The MerchantAccount object in the OPTeller class either is null or contains an invalid attribute.
6017	M	The MerchantAccount object in the OPTeller class either is null or contains an invalid attribute.
6018	M	The MerchantAccount object in the OPTeller class either is null or contains an invalid attribute.
6019	M	The MerchantAccount object in the OPTeller class either is null or contains an invalid attribute.
6022	M	A SOAP exception occurred while invoking the service “charge”.
6023	M	You have received a SOAP fault response while processing a “charge” request.
6024	M	A SOAP exception occurred while invoking the service “verify”.
6025	M	You have received a SOAP fault response while processing a “verify” request.
6026	M	A SOAP exception occurred while invoking the service “ping”.
6027	M	You have received a SOAP fault response while processing a “ping” request.
6028	M	A SOAP exception occurred while invoking the service “decline”.
6029	M	You have received a SOAP fault response while processing a “decline” request.
6030	M	A SOAP exception occurred while invoking the service “credit”.
6031	M	You have received a SOAP fault response while processing a “credit” request.
6032	M	Unable to open the op-config.xml file. Either it is in use by another process or you do not have the permission required to open it.
6035	M	Could not find the environment variable OPTIMAL_DIR. Please verify that the OPTIMAL_DIR variable is set and targets the directory in which you installed the application.
7000	D	The address you are trying to create already exists.
7001	M	Unable to create address. Try again or contact Technical Support.
7002	M	Unable to update address. Try again or contact Technical Support.
7003	M	Unable to locate the specified address.
7004	M	Unable to load address. Try again or contact Technical Support.
7005	M	You have provided an invalid address type.
7006	M	Unable to locate addresses for consumer.
7007	M	Unable to load addresses. Try again or contact Technical Support.
7010	D	The consumer you are trying to create already exists.
7011	M	Unable to create consumer. Try again or contact Technical Support.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
7012	M	Unable to update consumer. Try again or contact Technical Support.
7013	M	Unable to locate the specified consumer.
7014	M	Unable to load consumer. Try again or contact Technical Support.
7015	M	You have specified an invalid consumer creation type.
7016	M	You have specified an invalid consumer title.
7020	D	The contact method you are trying to create already exists.
7021	M	Unable to create contact method. Try again or contact Technical Support.
7022	M	Unable to update contact method. Try again or contact Technical Support.
7023	M	Unable to locate the specified contact method.
7024	M	Unable to load contact method. Try again or contact Technical Support.
7025	M	You have provided an invalid contact method type.
7026	M	Unable to locate contact methods for consumer.
7027	M	Unable to load contact methods. Try again or contact Technical Support.
7040	D	The payment method you are trying to create already exists.
7041	M	Unable to create payment method. Try again or contact Technical Support.
7042	M	Unable to update payment method. Try again or contact Technical Support.
7043	M	Unable to locate the specified payment method.
7044	M	Unable to load payment method. Try again or contact Technical Support.
7045	M	You have provided an invalid payment method type.
7046	M	Unable to locate payment methods for consumer.
7047	M	Unable to load payment methods. Try again or contact Technical Support.
7050	D	The billing schedule you are trying to create already exists.
7051	M	Unable to create billing schedule. Try again or contact Technical Support.
7052	M	Unable to update billing schedule. Try again or contact Technical Support.
7053	M	Unable to locate the specified billing schedule.
7054	M	Unable to load billing schedule. Try again or contact Technical Support.
7055	M	You have provided an invalid payment interval type.
7056	M	Unable to locate billing schedules for consumer.
7057	M	Unable to load billing schedules. Try again or contact Technical Support.
7058	M	Billing amount must be greater than zero.
7059	M	Unable to update billing schedule status. Try again or contact Technical Support.
7070	M	Unable to execute the search operation. Try again or contact Technical Support.
7080	M	Unable to update records. Try again or contact Technical Support.
7090	D	The consumer status you are trying to create already exists.
7091	M	Unable to create consumer status. Try again or contact Technical Support.
7092	M	Unable to update consumer status. Try again or contact Technical Support.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
7093	M	Unable to locate the specified consumer status.
7094	M	Unable to load consumer status. Try again or contact Technical Support.
7095	M	Unable to locate the merchant legal entity.
7096	M	Unable to retrieve merchant legal entity. Try again or contact Technical Support.
8000	M	Unable to perform operation. Try again or contact Technical Support.
8001	M	You have submitted an invalid request. Please verify your parameters and retry.
8002	M	Your request is missing the <i>consumerInfo</i> element.
8003	M	Your request is missing a value for the <i>firstName</i> element.
8004	M	Your request is missing a value for the <i>lastName</i> element.
8005	M	Your request is missing the <i>billingAddress/shippingAddress</i> element.
8006	M	Your request is missing <i>billingAddress/shippingAddress</i> information. There is no value for the <i>street</i> element.
8007	M	Your request is missing <i>billingAddress/shippingAddress</i> information. There is no value for the <i>city</i> element.
8008	M	Your request is missing <i>billingAddress/shippingAddress</i> information. There is no value for the <i>zip</i> element.
8009	M	Your request is missing <i>billingAddress/shippingAddress</i> information. There is no value for the <i>country</i> element.
8010	M	Your request is missing <i>billingAddress/shippingAddress</i> information. There is no value for the <i>state</i> or <i>region</i> element.
8011	M	Your request is missing the <i>contactMethod</i> element.
8012	M	Your request is missing a home telephone number, cell phone number, or email address for the <i>type</i> element.
8013	M	Your request is missing the <i>paymentMethod</i> element.
8014	M	Your request is missing a value for the <i>id</i> child element of the <i>billingAddress</i> element.
8015	M	Your request is missing either the <i>card</i> or <i>check</i> child element of the <i>paymentMethod</i> element.
8016	M	Your request is missing the <i>card</i> element.
8017	M	Your request is missing a value for the <i>cardNum</i> element.
8018	M	Your request is missing a value for the <i>cardType</i> element.
8019	M	Your request is missing the <i>cardExpiry</i> element.
8020	M	Your request is missing a value for the <i>month</i> element.
8021	M	Your request is missing a value for the <i>year</i> element.
8022	M	Your request is missing the <i>check</i> element.
8023	M	Your request is missing a value for the <i>accountNum</i> element.
8024	M	Your request is missing a value for the <i>accountType</i> element.
8025	M	Your request is missing a value for the <i>routingNum</i> element.
8026	M	Your request is missing a value for the <i>bankName</i> element.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
8027	M	Your request is missing a value for the <i>checkNum</i> element.
8028	M	Your request is missing the <i>billingSchedule</i> element.
8029	M	Your request is missing a value for the <i>intervalCode</i> element.
8030	M	Your request is missing a value for the <i>nextBillingDate</i> element.
8031	M	Your request is missing a value for the <i>startDate</i> element.
8032	M	Your request is missing a value for the <i>amount</i> element.
8033	M	Your request is missing the <i>merchantAccount</i> element.
8034	M	Your request is missing a value for the <i>accountNum</i> element.
8035	M	Your request is missing a value for the <i>storeID</i> element.
8036	M	Your request is missing a value for the <i>storePwd</i> element.
8037	M	The credentials supplied with the <i>merchantAccount</i> element could not be validated.
8038	M	An internal error occurred. Please retry the request.
8039	M	Your request is missing a value for the <i>consumerId</i> element.
8040	M	Your request is missing a value for the <i>title</i> element.
8041	M	Your request is missing a value for the <i>merchantRefNum</i> child element of the <i>consumerInfo</i> element.
8042	M	Your request is missing a value for the <i>street2</i> element.
8043	M	You have submitted an invalid <i>address id</i> element. It does not match the <i>consumerId</i> submitted with the request.
8044	M	You have submitted an invalid <i>contactMethod id</i> element. It does not match the <i>consumerId</i> submitted with the request.
8045	M	Your request is missing a value for the <i>paymentMethodId</i> element.
8046	M	Your request is missing a value for the <i>ccHolderName</i> element.
8047	M	Your request is missing a value for the <i>merchantRefNum</i> child element of the <i>paymentMethod</i> element.
8048	M	Your request is missing a value for the <i>id</i> child element of the <i>billingSchedule</i> element.
8049	M	Your request is missing a value for the <i>serviceName</i> element.
8050	M	Your request is missing a value for the <i>statusCode</i> element.
8051	M	Your request is missing a value for the <i>lastDayOfTheMonth</i> element.
8052	M	Your request is missing a value for the <i>merchantRefNum</i> child element of the <i>billingSchedule</i> element.
8053	M	Your request contains two <i>consumerId</i> elements that don't match.
8054	M	You attempted an unsupported operation.
8055	M	You have submitted an invalid <i>endDate</i> element.
8056	M	You have submitted an invalid <i>issueNumber</i> element.
8057	M	You have submitted an invalid <i>statusCode</i> element.
8058	M	You have submitted an invalid <i>intervalCode</i> element.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
8059	M	You have submitted an invalid <i>title</i> element.
8060	M	You have submitted an invalid <i>cardType</i> element.
8061	M	You have submitted an invalid <i>accountType</i> element.
8062	M	You have submitted an invalid <i>state</i> element.
8063	M	You have submitted an invalid <i>country</i> element.
8064	M	You have submitted an invalid <i>startDate</i> element.
8065	M	You have submitted an invalid <i>endDate</i> element.
8066	M	You have submitted an invalid <i>nextBillingDate</i> element.
8067	M	The <i>endDate</i> cannot be before the <i>startDate</i> .
8068	M	The <i>nextBillingDate</i> cannot be after the <i>endDate</i> .
8069	M	The <i>nextBillingDate</i> cannot be before the <i>startDate</i> .
8070	M	You have submitted an invalid <i>cardExpiry</i> element.
8071	M	You have submitted an invalid paymentMethod <i>billingAddressId</i> . Either it does not match the <i>consumerId</i> submitted with the request or it is the wrong address type.
8072	M	You have submitted an invalid paymentMethod <i>shippingAddressId</i> . Either it does not match the <i>consumerId</i> submitted with the request or it is the wrong address type.
8073	M	You have submitted an invalid contactMethod <i>type</i> element.
8074	M	You have submitted an invalid paymentMethod <i>id</i> element. It does not match the <i>consumerId</i> submitted with the request.
8075	M	You have submitted an invalid billingSchedule <i>id</i> element. It does not match the <i>consumerId</i> submitted with the request.
8076	M	You have submitted an invalid <i>consumerId</i> element. The consumer specified is not registered with this merchant.
8077	M	You have submitted an invalid billingSchedule <i>paymentMethodId</i> element. The payment method identified is not registered with this consumer.
8078	D	You have submitted a duplicate request within the last 24 hours.
8079	D	You have submitted a file that has already been processed within the last 24 hours.
8080	M	You have submitted an invalid record type. It is missing the <i>recurringBillingDetails</i> element.
8081	D	You have submitted an empty file.
8082	M	You have submitted an invalid request.
8083	M	Your request is missing a value for the <i>cvv</i> element.
8084	M	You have submitted an invalid <i>cvv</i> element.
8085	M	Your request is missing a value for the Merchant Transaction ID element.
8086	M	Your request is missing a value for the Transaction ID element.
8087	M	You have submitted an invalid <i>accountNum</i> element.
8088	M	You have submitted a record with an invalid number of fields.
8089	D	You have submitted a file that exceeds the file size limit.

Table B-1: Response Codes (Continued)

Response Code	Action	Description
8090	M	You have submitted an invalid Transaction Date element.
8091	M	You have submitted an invalid ID Expiration element.
8092	M	You have submitted an unsupported transaction type for the Transaction Code element.
8093	M	You have submitted an invalid <i>billingSchedule</i> element. Your request is either missing a <i>paymentMethodId</i> or missing a new <i>paymentMethod</i> element.
8094	M	You have submitted an invalid <i>paymentMethod</i> element. Your request is either missing a <i>billingAddressId</i> or missing a new <i>billingAddress</i> element.
8095	M	You have submitted a file that exceeds the limit of 1000 records.
8096	M	You have submitted an invalid <i>transactionMode</i> element. Please verify this parameter and retry the transaction.

Action codes

The transaction processor returns an action along with each response code. The meanings for the action codes are as follows:

- **C** = Consumer Parameter Error. The consumer has provided incorrect information. Ask the customer to correct the information.
- **D** = Do Not Retry
- **M** = Merchant Parameter Error. Your application has provided incorrect information. Verify your information.
- **R** = Retry

Return codes

These are the codes used by the ACH system for identifying the various reasons a check or payment is returned. These codes are generated by customers' banks (RDFI) to return items.

Table B-2: Return Codes

Code	Description	Code	Description
R01	Insufficient Funds	R02	Account Closed
R03	No Account/Unable to Locate Account	R04	Invalid Account Number
R06	Returned per ODFI's Request	R07	Authorization Revoked by Customer *
R08	Payment Stopped or Stop Payment	R09	Uncollected Funds
R10	Customer Advises Not Authorized *	R11	Check Truncation Entry Return
R12	Branch Sold to Another RDFI	R13	RDFI Not Qualified to Participate
R14	Representative Payee Deceased or Unable to Continue in that Capacity	R15	Beneficiary or Account Holder Deceased
R16	Account Frozen	R17	File Record Edit Criteria
R18	Improper Effective Entry Date	R19	Amount Field Error
R20	Non-Transaction Account	R21	Invalid Company Identification

Table B-2: Return Codes (Continued)

Code	Description	Code	Description
R22	Invalid Individual ID Number	R23	Credit Entry Refused by Receiver
R24	Duplicate Entry	R25	Addenda Error
R26	Mandatory Field Error	R27	Trace Number Error
R28	Routing Number Check Digit Error	R29	Corporate Customer Advises Not Authorized
R30	RDFI Not Participant in CK Truncation Program	R31	Permissible Return Entry
R32	RDFI Non-Settlement	R33	Return of XCX Entry
R34	Limited Participation DFI	R35	Return of Improper Debit Entry
R36	Return of Improper Credit Entry	R51	Item is Ineligible, Notice Not Provided, Signature Not Genuine, or Item Altered
R52	Stop Payment on Item		

*An initially cleared item can be returned as "Unauthorized", "Authorized", or "Authorization Revoked" for up to 60 days following the date of Presentment.

Geographical Codes

Province codes

Table C-1: Province Codes

Province	Code
Alberta	AB
British Columbia	BC
Manitoba	MB
New Brunswick	NB
Newfoundland	NL
Nova Scotia	NS
Northwest Territories	NT
Nunavut	NU
Ontario	ON
Prince Edward Island	PE
Quebec	QC
Saskatchewan	SK
Yukon	YT

State codes

Table C-2: State Codes

State	Code	State	Code
Alabama	AL	Alaska	AK
Arizona	AZ	Arkansas	AR
California	CA	Colorado	CO
Connecticut	CT	Delaware	DE
District of Columbia	DC	Florida	FL
Georgia	GA	Hawaii	HI
Idaho	ID	Illinois	IL
Indiana	IN	Iowa	IA
Kansas	KS	Kentucky	KY
Louisiana	LA	Maine	ME
Maryland	MD	Massachusetts	MA
Michigan	MI	Minnesota	MN
Mississippi	MS	Missouri	MO
Montana	MT	Nebraska	NE
Nevada	NV	New Hampshire	NH
New Jersey	NJ	New Mexico	NM
New York	NY	North Carolina	NC
North Dakota	ND	Ohio	OH
Oklahoma	OK	Oregon	OR
Pennsylvania	PA	Rhode Island	RI
South Carolina	SC	South Dakota	SD
Tennessee	TN	Texas	TX
Utah	UT	Vermont	VT
Virginia	VA	Washington	WA
West Virginia	WV	Wisconsin	WI
Wyoming	WY	Canada	CN
International	IT	Puerto Rico	PR
U.S. Virgin Islands	VI	Northern Mariana Is.	MP
Guam	GU	American Samoa	AS
Palau	PW	Armed Forces Americas	AA
Armed Forces Europe	AE	Armed Forces Pacific	AP
United States Federal	US		

Country codes

Table C-3: Country Codes

Country	Code	Country	Code
Afghanistan	AF	Åland Islands	AX
Albania	AL	Algeria	DZ
American Samoa	AS	Andorra	AD
Angola	AO	Anguilla	AI
Antarctica	AQ	Antigua and Barbuda	AG
Argentina	AR	Armenia	AM
Aruba	AW	Australia	AU
Austria	AT	Azerbaijan	AZ
Bahamas	BS	Bahrain	BH
Bangladesh	BD	Barbados	BB
Belarus	BY	Belgium	BE
Belize	BZ	Benin	BJ
Bermuda	BM	Bhutan	BT
Bolivia	BO	Bosnia and Herzegovina	BA
Botswana	BW	Bouvet Island	BV
Brazil	BR	British Indian Ocean Territory	IO
Brunei Darussalam	BN	Bulgaria	BG
Burkina Faso	BF	Burundi	BI
Cambodia	KH	Cameroon	CM
Canada	CA	Cape Verde	CV
Cayman Islands	KY	Central African Republic	CF
Chad	TD	Chile	CL
China	CN	Christmas Island	CX
Cocos (Keeling) Islands	CC	Colombia	CO
Comoros	KM	Congo	CG
Congo, Democratic Republic of	CD	Cook Islands	CK
Costa Rica	CR	Côte D'Ivoire	CI
Croatia	HR	Cuba	CU
Cyprus	CY	Czech Republic	CZ
Denmark	DK	Djibouti	DJ
Dominica	DM	Dominican Republic	DO
Ecuador	EC	Egypt	EG

Table C-3: Country Codes (Continued)

Country	Code	Country	Code
El Salvador	SV	Equatorial Guinea	GQ
Eritrea	ER	Estonia	EE
Ethiopia	ET	Falkland Islands	FK
Faroe Islands	FO	Fiji	FJ
Finland	FI	France	FR
Metropolitan France	FX	French Guiana	GF
French Polynesia	PF	French Southern Territories	TF
Gabon	GA	Gambia	GM
Georgia	GE	Germany	DE
Ghana	GH	Gibraltar	GI
Greece	GR	Greenland	GL
Grenada	GD	Guadeloupe	GP
Guam	GU	Guatemala	GT
Guernsey	GG	Guinea	GN
Guinea-Bissau	GW	Guyana	GY
Haiti	HT	Heard and McDonald Islands	HM
Honduras	HN	Hong Kong	HK
Hungary	HU	Iceland	IS
India	IN	Indonesia	ID
Iran (Islamic Republic of)	IR	Iraq	IQ
Ireland	IE	Isle of Man	IM
Israel	IL	Italy	IT
Jamaica	JM	Japan	JP
Jersey	JE	Jordan	JO
Kazakhstan	KZ	Kenya	KE
Kiribati	KI	Korea, Democratic People's Republic	KP
Korea, Republic of	KR	Kuwait	KW
Kyrgyzstan	KG	Lao People's Democratic Republic	LA
Latvia	LV	Lebanon	LB
Lesotho	LS	Liberia	LR
Libyan Arab Jamahiriya	LY	Liechtenstein	LI
Lithuania	LT	Luxembourg	LU
Macau	MO	Macedonia	MK
Madagascar	MG	Malawi	MW

Table C-3: Country Codes (Continued)

Country	Code	Country	Code
Malaysia	MY	Maldives	MV
Mali	ML	Malta	MT
Marshall Islands	MH	Martinique	MQ
Mauritania	MR	Mauritius	MU
Mayotte	YT	Mexico	MX
Micronesia, Federated States of	FM	Moldova, Republic of	MD
Monaco	MC	Mongolia	MN
Montenegro	ME	Montserrat	MS
Morocco	MA	Mozambique	MZ
Myanmar	MM	Namibia	NA
Nauru	NR	Nepal	NP
The Netherlands	NL	Netherlands Antilles	AN
New Caledonia	NC	New Zealand	NZ
Nicaragua	NI	Niger	NE
Nigeria	NG	Niue	NU
Norfolk Island	NF	Northern Mariana Islands	MP
Norway	NO	Oman	OM
Pakistan	PK	Palau	PW
Palestinian Territory, Occupied	PS	Panama	PA
Papua New Guinea	PG	Paraguay	PY
Peru	PE	Philippines	PH
Pitcairn	PN	Poland	PL
Portugal	PT	Puerto Rico	PR
Qatar	QA	Reunion	RE
Romania	RO	Russian Federation	RU
Rwanda	RW	Saint Helena	SH
Saint Kitts and Nevis	KN	Saint Lucia	LC
Saint Vincent and the Grenadines	VC	Samoa	WS
San Marino	SM	Sao Tome and Principe	ST
Saudi Arabia	SA	Senegal	SN
Serbia	RS	Seychelles	SC
Sierra Leone	SL	Singapore	SG
Slovakia (Slovak Republic)	SK	Slovenia	SI
Solomon Islands	SB	Somalia	SO

Table C-3: Country Codes (Continued)

Country	Code	Country	Code
South Africa	ZA	South Georgia and the South Sandwich Islands	GS
Spain	ES	Sri Lanka	LK
St. Helena	SH	St. Pierre and Miquelon	PM
Sudan	SD	Suriname	SR
Svalbard and Jan Mayen Islands	SJ	Swaziland	SZ
Sweden	SE	Switzerland	CH
Syrian Arab Republic	SY	Taiwan	TW
Tajikistan	TJ	Tanzania, United Republic of	TZ
Timor-Leste	TL	Thailand	TH
Togo	TG	Tokelau	TK
Tonga	TO	Trinidad and Tobago	TT
Tunisia	TN	Turkey	TR
Turkmenistan	TM	Turks and Caicos Islands	TC
Tuvalu	TV	Uganda	UG
Ukraine	UA	United Arab Emirates	AE
United Kingdom	GB	United States	US
United States Minor Outlying Islands	UM	Uruguay	UY
Uzbekistan	UZ	Vanuatu	VU
Vatican City State (Holy See)	VA	Venezuela	VE
Vietnam	VN	Virgin Islands (British)	VG
Virgin Islands (U.S.)	VI	Wallis and Futuna Islands	WF
Western Sahara	EH	Yemen	YE
Zambia	ZM	Zimbabwe	ZW