



Developers Manual

API Documentation

REDFIN NETWORK
PAYMENT GATEWAY
Document Version 2.07.0415-a

Copyright © 2001-08 Secured Financial Network, Inc. All Rights Reserved



Introduction

Visit us at: <http://www.redfinnet.com>.

Copyright © 2001-08 Secured Financial Network, Inc. All Rights Reserved

Congratulations on the selection of the **RedFin** Network Payment Gateway's Software Development Kit (SDK), the most advanced solution in the industry for processing credit cards, debit cards and check services. This SDK provides you with a fast, easy, reliable way to authorize credit card, ATM/debit card, and check transactions on your application.

Your opinion is important to us. If you have any suggestions feel free to [email us](#).

Thank you for choosing **RedFin Network!**

Virtual Payment Solutions.

1180 SW 36 Avenue Suite 204

Pompano Beach, FL 33069

Email: support@virtualpaymentsolutions.com

Phone: 866-685-4326

License Information

DISCLAIMER

The **RedFin** Network and supporting services are licensed on an “as is” basis. There are no warranties, expressed or implied, including, but not limited to, warranties of merchantability or of fitness for a particular purpose, and all such warranties are expressly and specifically disclaimed. **RedFin** Network shall have no liability or responsibility to you nor any other person or entity with respect to any liability, loss, or damage caused or alleged to be caused directly or indirectly by any **RedFin** Network software and supporting services. Use of the **RedFin** Network software and supporting services signifies agreement with this disclaimer and is subject to the license agreement provided with the installation of the software.

LICENSE

Use of the **RedFin** Network services requires a signed service agreement by the end user of the service or reseller of such services. **RedFin** Network grants you the right to use this software and supporting software for the purpose of connecting to the **RedFin** Network Payment Gateway. You may physically transfer each license, without cost, to a different computer.

RESTRICTED USE

Copying of the manual, interface specifications, or system files for use other than backing up files and/or to connect to systems other than the **RedFin** Network Payment Gateway is prohibited. You may not remove any product identification, copyright, or other proprietary notices from the software or documentation. Reverse engineering is strictly prohibited.

This client software is provided for use in connecting with the **RedFin** Network Payment Gateway only. Use of any components, code, or documentation to connect to other systems is strictly prohibited and a violation of this agreement and subject to all remedies available by law.

EULA

END-USER LICENSE AGREEMENT FOR **RedFin** NETWORKSOFTWARE

IMPORTANT READ CAREFULLY:

This **RedFin** Network End-User License Agreement (EULA) is a legal agreement between you (either an individual or a single entity) and **RedFin** Network for the **RedFin** Network product accompanying this EULA, which includes computer software and may include associated media, printed materials, and online or electronic documentation (SOFTWARE). By installing, copying, or otherwise using the SOFTWARE, you agree to be bound by the terms of this EULA. If you do not agree to the terms of this EULA, do not install, copy, or otherwise use the SOFTWARE.

SOFTWARE PRODUCT LICENSE

RedFin Network Software (SOFTWARE) is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE is licensed, not sold.

GRANT OF LICENSE

Secured Financial Network, Inc., hereby grants you ("Licensee") a limited, nonexclusive, non-transferable, royalty-free license to make and use a single copy an unlimited number of copies of the SOFTWARE accompanying this EULA to be installed on CPUs residing on Licensee's premises, solely for Licensee's internal use. **RedFin** Network and its suppliers shall retain title and all ownership rights to the product, and this Agreement shall not be construed in any manner as transferring any rights of ownership or license to the SOFTWARE or to the features or information therein, except as specifically stated herein.

USE RESTRICTION

Licensee acknowledges that the SOFTWARE acquired hereunder can only be used for reseller and/or end merchant evaluation and/or product's intended use of payment processing. Use of this product for any other purposes such as Competitor Evaluation, Reverse Engineering, Decompilation, and Disassembly is violation of this License and Licensee agrees that such acts are a blatant and flagrant violation of this License and will be subject to any and all penalties and remedies available by Law for violation of intellectual property laws and treaties.

DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS

- (a) Limitations on Modification, Reverse Engineering, Decompilation, and Disassembly. Licensee may not modify, reverse engineer, decompile, or disassemble the SOFTWARE, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.
- (b) No Rights to Sublicense or Rental. Licensee may not rent, lease or lend the SOFTWARE.
- (c) Termination. Without prejudice to any other rights, **RedFin** Network may terminate this EULA if Licensee fails to comply with the terms and conditions of this EULA. In such event, Licensee must destroy all copies of the SOFTWARE and all of its component parts.
- (d) Reservation of Rights. Licensee agrees that the SOFTWARE is owned **RedFin** Network and all rights not expressly granted herein are reserved by **RedFin** Network.

PRODUCT MAINTENANCE

Licensee understands and agrees that **RedFin** Network may provide updates to the SOFTWARE from time to time but **RedFin** Network shall have no obligation to provide maintenance or updates to Licensee for SOFTWARE licensed under this Agreement.

CONFIDENTIALITY

- (a) Licensee understands that the SOFTWARE contains confidential and proprietary trade secret information of **RedFin** Network that is not commercially available to the public. The Licensee agrees that, in partial consideration for **RedFin** Network's allowing Licensee access to and use of the SOFTWARE pursuant to this EULA:
 - (i) Licensee shall treat the SOFTWARE in the same manner that it treats its most confidential and proprietary trade secret materials, and
 - (ii) Licensee shall take all measures necessary to prevent the SOFTWARE from falling into the possession of persons not bound to maintain the confidentiality of the trade secrets contained within the SOFTWARE. As such, Licensee shall only permit employees and contractors who have a need to use the SOFTWARE for the purposes stated in the license grant (Section 1) to have access to and use such SOFTWARE, and Licensee's contractors shall not use the SOFTWARE unless and until they have entered into written non-disclosure agreements with the Licensee that require them to maintain the confidentiality of SOFTWARE. Licensee shall promptly advise **RedFin** Network, in writing, of any misappropriation or misuse of the SOFTWARE by any person which may come to the Licensee's attention.

(b) Licensee understands and agrees that disclosure or use of the SOFTWARE except as authorized above will result in irreparable harm to RedFin Network and that monetary damages may be inadequate to compensate RedFin Network for such breach. Accordingly, the Licensee agrees that RedFin Network will, in addition to any other remedies available to it at law or in equity, be entitled to injunctive relief to enforce the terms of this Agreement.

COPYRIGHT

All title and copyrights in and to the SOFTWARE (including but not limited to any images, photographs, animations, video, audio, music, text, and applets incorporated into the SOFTWARE), the accompanying printed materials, and any copies of the SOFTWARE are owned by RedFin Network or its suppliers. The SOFTWARE is protected by copyright laws and international treaty provisions. Therefore, Licensee must treat the SOFTWARE PRODUCT like any other copyrighted material except that Licensee may install the SOFTWARE on a single computer provided Licensee keep the original solely for backup or archival purposes. Licensee may not copy any printed materials accompanying the SOFTWARE without prior permission of RedFin Network.

U.S. GOVERNMENT RESTRICTED RIGHTS

The SOFTWARE and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of the Commercial Computer Software Restricted Rights at 48 CFR 52.227-19, as applicable. Manufacturer is Secured Financial Network, Inc./101 NE 3rd Ave., Suite 1500/Ft. Lauderdale, Fl 33301.

EXPORT RESTRICTIONS

Licensee acknowledges that the SOFTWARE acquired hereunder are subject to the export control laws and regulations of the U.S.A., and any amendments thereof. Licensee confirms that with respect to these SOFTWARE, it will not export or re-export them, directly or indirectly, either to (i) any countries that are subject to U.S.A export restrictions (currently including, but not necessarily limited to, Cuba, the Federal Republic of Yugoslavia (Serbia and Montenegro), Iran, Iraq, Libya, North Korea, South Africa (military and police entities), Syria, and Vietnam); (ii) any end user who Licensee knows or has reason to know will utilize them in the design, development or production of nuclear, chemical or biological weapons; or (iii) any end user who has been prohibited from participating in the U.S.A. export transactions by any federal agency of the U.S.A.

government. Licensee further acknowledges that the SOFTWARE may include technical data subject to export and re-export restrictions imposed by U.S.A. law.

DISCLAIMER OF WARRANTY

THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, RED FIN NETWORK FURTHER DISCLAIMS ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. THE ENTIRE RISK ARISING OUT OF THE USE OR PERFORMANCE OF THE PRODUCT AND DOCUMENTATION REMAINS WITH LICENSEE.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL RED FIN NETWORK BE LIABLE FOR ANY CONSEQUENTIAL, INCIDENTAL, DIRECT, INDIRECT, SPECIAL, PUNITIVE, OR OTHER DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE PRODUCT OR DOCUMENTATION, EVEN IF RED FIN NETWORK HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES/JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO LICENSEE.

GOVERNING LAW

This EULA shall be governed by the laws of the State of Washington.

Should you have any questions concerning this EULA, or if you desire to contact **RedFin** Network for any reason, please write: Secured Financial Network, Inc., 1180 SW 36th Avenue, Pompano Beach FL, 33069.

Overview

Testing

You can request a test account on <http://www.redfinnet.com/contact.html> or email to support@virtualpaymentsolutions.com.

Please include the following information on your request:

- Company name;
- Contact name;
- Phone number;
- Email address associated with the test account;
- Type of account: If you are a Merchant or a Developer/Integrator;
- Name of Sales Agent/Company who introduced you to **RedFin** Network.

Testing can be performed with the following test cards:

Card Type	Number
MasterCard	5000300020003003
Visa	4005550000000019
Discover	6001111111111117
Diners	36999999999999
AMEX	374255312721002

Web Services

The **RedFin** Network was built using XML and Web Services allowing for easy integration into any platform/system.

The following operations are supported:

- [GetCardTrx](#) – Retrieves card transaction details for a merchant
- [GetCardTrxSummary](#) – Retrieves card transaction summary for a merchant
- [GetCheckTrx](#) – Retrieves check transaction details for a merchant
- [GetCardType](#) – Returns the name of the card issuer: Visa, MasterCard, etc.
- [GetInfo](#) – Retrieves information from the web service
- [GetOpenBatchSummary](#) – Retrieves payment type transaction summary of the current open batch for a merchant
- [IsCommercialCard](#) – Returns (T/F) if the card is a known commercial card
- [ProcessCheck](#) – Processes check transactions for a merchant
- [ProcessCreditCard](#) – Processes credit card transactions for a merchant
- [ProcessDebitCard](#) – Processes debit card transactions for a merchant
- [ProcessEBTCard](#) – Processes EBT card transactions for a merchant
- [ProcessGiftCard](#) – Processes gift card transactions for a merchant
- [ProcessSignature](#) – Sends a signature to apply to a receipt transaction
- [ValidCard](#) – Validates the credit card by checking the card length based on the card type, performing a mod 10 checksum, and validating the expiration date
- [ValidCardLength](#) – Validates the credit card length
- [VaildExpDate](#) – Validates the expiration date of the credit card
- [ValidMod10](#) – Validates the credit card by performing a mod 10 checksum on the card number; returns (T/F)
- [AddMerchant](#) – Adds a merchant to the account
- [DeleteMerchant](#) – Deletes a merchant from the account
- [AddRecurringCreditCard](#) -Allows customer information to be programmatically stored through web services for recurring billing.
- [AddRecurringCheck](#)- Allows check information to be programmatically stored through web services for recurring billing.

- [* **ProcessCreditCard**](#) -Allows for processing credit card transactions in recurring billing .
- [* **ProcessCheck**](#)- Allows for processing ACH /check transactions for recurring billing.
- [**ManageCheckInfo**](#) – Allows for programmatic management of existing check information for recurring billing.
- [**ManageCreditCardInfo**](#)- Allows for programmatic management of credit card information for customers specific to recurring billing.
- [**ManageContract**](#) – Allows for managing existing contracts for updates and modifications.
- [**ManageCustomer**](#) – Allows for management of existing customers in the recurring billing web service.
- [**ManageContractAddDaysToNextBillDt**](#) – Allows for modification of next billing date for recurring billing contracts.
- [**GetNetworkID**](#)- This web service allows for returning the debit network ID if the debit card number matches any of these network's bin ranges

*** = This is for recurring.asmx webservice. Not to be confused with Transact.asmx**

GetCardTrx

This Web service operation retrieves card transaction details for a merchant:

<https://secure.redfinnet.com/admin/ws/trxdetail.aspx?op=GetCardTrx>.

Parameter	Description
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
RPNum	Required. The merchant's RP Number in order to uniquely identify merchant's account for the query
PNRef	Optional. The unique payment reference number assigned to the transaction. <i>If this field is provided, all other query fields will be ignored when using PNRef parameter to query the system.</i>
BeginDt	Required, except when PNRef is provided. The begin date of the date range in MM/DD/YYYY (or YYYY-MM-DD or YYYY-MM-DDThh:mm:ss) format. This date will be converted to YYYY-MM-DDThh:mm:ss (time is in 24-hour format). If the passed-in BeginDt does not contain time information, BeginDt will default to midnight on the given date. For example: 2005-08-19T12:00:12 is kept as is 2005-08-19 becomes 2005-08-19T00:00:00 2005/08/19 becomes 2005-08-19T00:00:00 08/19/2005 becomes 2005-08-19T00:00:00 The query to obtain transactions in the date range is constructed as follows: (Date DT >=BeginDt) AND (Date DT<EndDt) where Date DT is the transaction timestamp
EndDt	Required, except when PNRef is provided. The end date of the date range in MM/DD/YYYY (or YYYY-MM-DD or YYYY-MM-DDThh:mm:ss) format. This date will be converted to YYYY-MM-DDThh:mm:ss (time is in 24-hour format). If the passed-in EndDt does not contain time information, EndDt will be incremented to the <i>next day</i> at midnight such that no transaction on the desired end date will be excluded based on its time. For example: 2005-08-19T12:00:12 is kept as is 2005-08-19 becomes 2005-08-20T00:00:00 08/19/2005 becomes 2005-08-20T00:00:00
PaymentType	Optional. If provided, only those transactions matching the PaymentType will be included. Valid values are: <ul style="list-style-type: none"> • 'AMEX' American Express card • 'CARTBLANCH' Carte Blanch card • 'DEBIT' Debit card • 'DINERS' Diners Club card • 'DISCOVER' Novus Discover card • 'EBT' Electronic Benefit Transfer • 'JAL' JAL card

	<ul style="list-style-type: none"> • 'JCB' Japanese Commercial Bank card • 'MASTERCARD' Master card • 'VISA' Visa card • 'EGC' Gift card • 'PAYRECEIPT' to retrieve receipt images that were uploaded to the payment server • 'SETTLE' to retrieve requests to settle transactions <p>Or any permutation of the above values, e.g. "PAYRECEIPT','SETTLE" will pull all transactions with either PayReceipt or Settle payment types.</p>
<p>ExcludePaymentType</p>	<p>Optional. If provided, any transaction matching the ExcludePaymentType will be excluded</p>
<p>TransType</p>	<p>Optional. If provided, only those transactions matching the TransType will be included. Valid values are</p> <ul style="list-style-type: none"> • 'Authorization' to retrieve previously-authorized (pre-auth) transactions • 'Capture' to retrieve captured transactions • 'Credit' to retrieve return transactions • 'ForceCapture' to retrieve force-auth transactions • 'GetStatus' to make an inquiry to the EBT or gift card's balance • 'PostAuth' to retrieve post-auth transactions • 'Purged' to remove a transaction from the current batch due to an error • 'Receipt' to retrieve receipt images that were uploaded to the payment server • 'RepeatSale' to retrieve repeat-sale transactions • 'Sale' to retrieve sale transactions • 'Void' to retrieve void transactions <p>Or any permutation of the above values, e.g. "Credit','Sale" will pull all transactions with either Credit or Sale transaction types.</p>
<p>ExcludeTransType</p>	<p>Optional. If provided, any transaction matching the ExcludeTransType will be excluded</p>
<p>ApprovalCode</p>	<p>Optional. If provided, only those transactions matching the ApprovalCode parameter will be included</p>
<p>Result</p>	<p>Optional. If provided, only those transactions matching the Result will be included. Valid values are:</p> <ul style="list-style-type: none"> • 0 is Approved • Anything else is declined; if you want all the declined transactions, you should leave this field empty and use the ExcludeResult=0 instead.

ExcludeResult	Optional. If provided, any transactions matching the ExcludeResult will be excluded.
NameOnCard	Optional. Cardholder's name as it appears on the card. If provided, only those transactions with cardholder's name matching NameOnCard will be included. Matching is done using wild cards: e.g. "test" will match "test", "1test" and "1test234"
CardNum	Optional. A card number. If provided, only those transactions with the cardholder's name matching CardNum will be included. Matching is done using wild cards
CardType	<p>If provided, only those transactions matching the CardType will be included. Valid values are:</p> <ul style="list-style-type: none"> • 'AMEX' American Express card • 'CARTBLANCH' Carte Blanch card • 'DEBIT' Debit card • 'DINERS' Diners Club card • 'DISCOVER' Novus Discover card • 'EBT' Electronic Benefit Transfer • 'JAL' JAL card • 'JCB' Japanese Commercial Bank card • 'MASTERCARD' Master card • 'VISA' Visa card • 'EGC' Gift card <p>Or any permutation of the above values, "'VISA','MASTER','DISCOVER'" will pull all transactions with either VISA, MASTER and DISCOVER card type</p> <p>There are cases when Cardtype needs to be set to ALL, for example CardType=ALL</p>
ExcludeCardType	Optional. If provided, any transaction matching the ExcludeCardType will be excluded
ExcludeVoid	Required, except when PNRef is provided. An option to exclude voided transactions or not; must either be TRUE or FALSE
User	Optional. The user who originated the transactions. If provided, only those transactions created by the matching User will be included. Matching is done using wild cards
InvoiceId	Optional. The invoice ID that was included in the original transaction. If provided, only those transactions with matching invoiceId will be included. Matching is done using wild cards
SettleFlag	Optional. An option to retrieve the settled transactions or unsettled transactions; must either be 1 for true or 0 for false
SettleMsg	Optional. The settlement ID or message returned from the host
SettleDt	Optional. The date of the settlement

TransformType	<p>Optional. The type of format to transform the data into. Leave the field blank to default to XML</p> <ul style="list-style-type: none"> • XML will output the plain XML string • XSL will use XSL to transform the XML output • DELIM uses ColDelim and RowDelim to format the output
Xsl	<p>Optional. This field is used only if the TransformType is XSL. The XSL to transform the resulting dataset. If provided, the resulting dataset will be transformed using this XSL. You may pass in a URL to the XSL file, or the XSL string itself. If this field is not empty, the Web Services will try to locate the file from the URL. If that also fails, it will treat it as an XSL string. In any case, the final XSL string will be loaded and validated against the XSL schema; if it passes, then that XSL will be used for transformation. A sample predefined XSL is included with this Web Services:</p> <ul style="list-style-type: none"> • https://secure.redfinnet.com/admin/ws/TabDelim.xsl for a tab delimited transformation
ColDelim	<p>Optional. This field is used only if the TransformType is DELIM. This defines the string that separates each column</p>
RowDelim	<p>Optional. This field is used only if the TransformType is DELIM. This defines the string that separates each row</p>
IncludeHeader	<p>Optional. This field is used only if the TransformType is DELIM. If TRUE, then field headers will be included in the first row using the same delimiter strings; must either be TRUE or FALSE</p>
ExtData	<p>Optional. Extended data in XML format. Valid values are:</p> <ul style="list-style-type: none"> • <IMAGE_TYPE>NO_IMAGE</IMAGE_TYPE> for no image • <IMAGE_TYPE>ONLY_IMAGE</IMAGE_TYPE> for only the image • <IMAGE_TYPE>ALL</IMAGE_TYPE> for all images • <CustomerID>CustomerID</CustomerID> for customer ID • <Amount>Amount</Amount> Total amount to search transactions for in DDDD.CC format. • <RegisterNum>RegisterNum</RegisterNum> Register number, originally passed with the transaction, to search transactions for.

Example

The following table contains sample data you can use to test this Web service. The **UserName**, **Password**, and **RPNum** parameters should be changed when testing this example yourself. The example data shown will create a list of sale card transactions that were processed between 1/1/2000 and 1/1/3000. The format of the output will show descriptive headers and will be similar to a Comma Separate Values (CSV) format because of the use of the "," character as the column delimiter.

Parameter	Value
-----------	-------

UserName	test
Password	123
RPNum	1
BeginDt	1/1/2000
EndDt	1/1/3000
TransType	'Sale'
ExcludeVoid	TRUE
TransformType	DELIM
ColDelim	,
RowDelim	
IncludeHeader	TRUE

Result: The following is the result of the using the Web Services with the sample data from the table above.

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

```
<string xmlns="https://secure.redfinnet.com/admin/ws ">
  TRX_HD_Key,Invoice_ID,Seq_Num_CH,Date_DT,Merchant_Key,User_Name
  _VC,Register_Number_CH,Reseller_Key,Payment_Type_ID,Trans_Type_ID,
  Processor_ID,TRX_Settle_Key,TRX_Settle_Msg_VC,Void_Flag_CH,Settle_Fla
  g_CH,Ref_Number_CH,Settle_Date_DT,Last_Update_DT,TRX_Card_Key,Card
  _Info_Key,Auth_Amt_MN,Tip_Amt_MN,Total_Amt_MN,Cash_Back_Amt_MN,
  SureCharge_Amt_MN,Account_Type_CH,Result_CH,Result_Txt_VC,Approval
  _Code_CH,Host_Ref_Num_CH,AVS_Resp_CH,AVS_Resp_Txt_VC,CV_Resp_C
  H,CV_Resp_Txt_VC,Host_Date_CH,Host_Time_CH,Acct_Num_CH,Exp_CH,Ty
  pe_CH,Name_on_Card_VC,Street_CH,Zip_CH,Track_VC,Pin_Block_CH,TRX_
  Receipt_key,Create_Date_DT,Receipt_Type_ID,IP_VC|14,,,12/3/2003
  10:47:48 AM,1,test , ,100,VISA ,Sale ,VITAL ,14,GB00004
  ACCEPTED,,1,,12/3/2003 10:46:34 AM,12/3/2003 10:48:13
  AM,14,14,2.5000,0,2.5000,0,0,VISA ,0 ,APPROVAL VITAL1 ,VITAL1 , ,
  0,, ,,,4266503700000247,1009 ,VISA ,VINCE VISA , , ,1, ,,,127.0.0.1|
  13,,,12/3/2003 10:04:29 AM,1,test , ,100,MASTERCARD,Sale ,VITAL ,
  13,GB00004 ACCEPTED,,1,,12/3/2003 10:46:34 AM,12/3/2003 10:48:13
  AM,13,13,2.8500,0,2.8500,0,0,MASTERCARD,0 ,APPROVAL VITAL9 ,VITAL9 ,
  ,0,, ,,,5424000000000015,0905 ,MASTERCARD, , , , ,,,127.0.0.1|
  12,,,12/3/2003 9:45:59 AM,1,test , ,100,VISA ,Sale ,VITAL ,12,GB00003
  ACCEPTED,,1,,12/3/2003 9:46:16 AM,12/3/2003 9:46:16
  AM,12,12,1.6800,0,1.6800,0,0,VISA ,0 ,APPROVAL VITAL3 ,VITAL3 , ,
  0,, ,,,4126196901499,0905 ,VISA , , , , ,,,127.0.0.1|10,,,12/3/2003
  9:35:19 AM,1,test , ,100,VISA ,Sale ,VITAL ,10,GB00003
  ACCEPTED,,1,,12/3/2003 9:46:16 AM,12/3/2003 9:46:16
  AM,10,10,2.3500,0,2.3500,0,0,VISA ,0 ,APPROVAL VITAL8 ,VITAL8 , ,
  0,, ,,,4055016727870315,0905 ,VISA , , , , ,,,127.0.0.1|7,,,12/3/2003
  9:27:57 AM,1,test , ,100,VISA ,Sale ,VITAL ,7,GB00002
```

ACCEPTED,,1,,12/3/2003 9:26:35 AM,12/3/2003 9:28:16
 AM,7,7,1.2500,0,1.2500,0,0,VISA ,0 ,APPROVAL VITAL4 ,VITAL4 , ,
 0,, ,,,4007000000027,0905 ,VISA , , , , ,,,127.0.0.1|6,,,12/3/2003 9:27:44
 AM,1,test , ,100,VISA ,Sale ,VITAL ,6,GB00002 ACCEPTED,,1,,12/3/2003
 9:26:35 AM,12/3/2003 9:28:16 AM,6,6,2.0000,0,2.0000,0,0,VISA ,0
 ,APPROVAL VITAL9 ,VITAL9 , ,0,, ,,,4007000000027,0905
 ,VISA , , , , ,,,127.0.0.1|5,,,12/3/2003 9:27:15 AM,1,test , ,
 100,MASTERCARD,Sale ,VITAL ,5,,,,,12/3/2003 9:27:17
 AM,5,5,1.2500,0,1.2500,0,0,MASTERCARD,12 , ACCT LENGTH ERR,EA , ,
 0,, ,,,5405001478615777532,0905 ,MASTERCARD, , , , ,,,127.0.0.1|
 2,,,12/3/2003 9:03:16 AM,1,test , ,100,MASTERCARD,Sale ,VITAL ,
 2,GB00001 ACCEPTED,,1,,12/3/2003 9:03:40 AM,12/3/2003 9:03:40
 AM,2,2,2.4500,0,2.4500,0,0,MASTERCARD,0 ,APPROVAL VITAL7 ,VITAL7 , ,
 0,, ,,,5424000000000015,0905 ,MASTERCARD, , , , ,,,127.0.0.1|
 1,,,12/3/2003 8:58:02 AM,1,test , ,100,VISA ,Sale ,VITAL ,1,GB00001
 ACCEPTED,,1,,12/3/2003 9:03:40 AM,12/3/2003 9:03:40
 AM,1,1,1.0000,0,1.0000,0,0,VISA ,0 ,APPROVAL VITAL3 ,VITAL3 , ,
 0,, ,,,4111111111111111,0905 ,VISA , , , , ,,,127.0.0.1

</string>

GetCardTrxSummary

This Web service operation retrieves a card transaction summary for a merchant:

<https://secure.redfinnet.com/admin/ws/trxdetail.aspx?op=GetCardTrxSummary>.

Parameter	Value
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
RPNum	Required. The merchant's RP Number in order to uniquely identify the merchant's account for the query
BeginDt	Required. The begin date of the date range in MM/DD/YYYY format. This date will be converted to MM/DD/YYYYT00:00:00:0000AM
EndDt	Required. The end date of the date range in MM/DD/YYYY format. This date will be converted to MM/DD/YYYYT12:59:59:9999PM
ApprovalCode	Optional. If provided, only those transactions matching the ApprovalCode parameter will be included
Register	Optional. The register that originated the transaction. If provided, only those transactions with the matching register will be included
NameOnCard	Optional. Cardholder's name as it appears on the card. If provided, only those transactions with cardholder's name matching NameOnCard will be included. Matching is done using wild cards: e.g. "test" will match "test", "1test" and "1test234"
CardNum	Optional. A card number. If provided, only those transactions with the cardholder's name matching CardNum will be included. Matching is done using wild cards
CardType	Optional. If provided, only those transactions matching the CardType will be included. Valid values are: <ul style="list-style-type: none"> • 'AMEX' American Express card • 'CARTBLANCH' Carte Blanch card • 'DEBIT' Debit card • 'DINERS' Diners Club card • 'DISCOVER' Novus Discover card • 'EBT' Electronic Benefit Transfer • 'JAL' JAL card • 'JCB' Japanese Commercial Bank card • 'MASTERCARD' Master card • 'VISA' Visa card • 'EGC' Gift card

ExcludeVoid	Required. Whether to exclude voided transactions; must either be TRUE or FALSE . Default is TRUE
User	Optional. The user who originated the transactions. If provided, only those transactions created by the matching User will be included. Matching is done using wild cards
SettleFlag	Optional. An option to retrieve the settled transactions or unsettled transactions; must either be TRUE or FALSE
SettleMsg	Optional. The settlement ID or message returned from the host
SettleDt	Optional. The settlement timestamp
TransformType	Optional. The type of format to transform the data into. Leave the field blank to default to XML <ul style="list-style-type: none"> • XML will output the plain XML string • XSL will use XSL to transform the XML output • DELIM uses ColDelim and RowDelim to format the output
Xsl	Optional. This field is used only if the TransformType is XSL . The XSL to transform the resulting dataset; if provided, the resulting dataset will be transformed using this XSL. You may pass in a URL to the XSL file, or the XSL string itself. If this field is not empty, the Web Services will try to locate the file from the URL. If that also fails, it will treat it as a XSL string. In any case, the final XSL string will be loaded and validated against the XSL schema; if it passes, then that XSL will be used for transformation. A sample predefined XSL is included with this Web Services: <ul style="list-style-type: none"> • https://secure.redfinnet.com/admin/ws/TabDelim.xsl for a tab delimited transformation
ColDelim	Optional. This field is used only if the TransformType is DELIM . This defines the string that separates each column
RowDelim	Optional. This field is used only if the TransformType is DELIM . This defines the string that separates each row
IncludeHeader	Optional. This field is used only if the TransformType is DELIM . If TRUE , then field headers will be included in the first row using the same delimiter strings; must either be TRUE or FALSE
ExtData	Optional. Extended data in XML format. Valid values are: <ul style="list-style-type: none"> • <IMAGE_TYPE>NO_IMAGE</IMAGE_TYPE> for no image • <IMAGE_TYPE>ONLY_IMAGE</IMAGE_TYPE> for only the image • <IMAGE_TYPE>ALL</IMAGE_TYPE> for all images

Example

The following table contains sample data you can use to test this Web service. The **UserName**, **Password**, and **RPNum** parameters should be changed when testing this

example yourself. The example data shown will create a summarized list of Visa card transactions that were processed between 1/1/2000 and 1/1/3000. The output data is in XML format because the **TransformType** parameter was not specified.

Parameter	Value
UserName	Test
Password	123
RPNum	1
BeginDt	1/1/2000
EndDt	1/1/3000
CardType	VISA
ExcludeVoid	FALSE

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="https://secure.redfinnet.com/Admin/ws">
  <CardTrxSummary> <PaymentMethod> <Payment_Type_ID>VISA
  </Payment_Type_ID> <Authorization>1.0000</Authorization>
  <Capture>0</Capture> <ForceCapture>0</ForceCapture>
  <PostAuth>0</PostAuth> <Return>0</Return> <Sale>2.0000</Sale>
  <Receipt>0</Receipt> <RepeatSale>0</RepeatSale>
  <Activate>0</Activate> <Deactivate>0</Deactivate>
  <Reload>0</Reload> <Authorization_Cnt>1</Authorization_Cnt>
  <Capture_Cnt>0</Capture_Cnt>
  <ForceCapture_Cnt>0</ForceCapture_Cnt>
  <PostAuth_Cnt>0</PostAuth_Cnt> <Return_Cnt>0</Return_Cnt>
  <Sale_Cnt>2</Sale_Cnt> <Receipt_Cnt>0</Receipt_Cnt>
  <RepeatSale_Cnt>0</RepeatSale_Cnt> <Activate_Cnt>0</Activate_Cnt>
  <Deactivate_Cnt>0</Deactivate_Cnt> <Reload_Cnt>0</Reload_Cnt>
  <Cnt>3</Cnt> </PaymentMethod> </CardTrxSummary>
</string>
```

GetCheckTrx

This Web service operation retrieves checks transaction details for a merchant:

<https://secure.redfinnet.com/admin/ws/trxdetail.aspx?op=GetCheckTrx>.

Parameter	Value
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
RPNum	Required. The merchant's RP Number in order to uniquely identify merchant's account for the query
PNRef	Optional. The unique payment reference number assigned to the transaction. If this field is provided, all other query fields will be ignored when using PNRef parameter to query the system
BeginDt	<p>Required, except when PNRef is provided. The begin date of the date range in MM/DD/YYYY (or YYYY-MM-DD or YYYY-MM-DDThh:mm:ss) format. This date will be converted to YYYY-MM-DDThh:mm:ss (time is in 24-hour format). If the passed-in BeginDt does not contain time information, BeginDt will default to midnight on the given date. For example:</p> <p>2005-08-19T12:00:12 is kept as is 2005-08-19 becomes 2005-08-19T00:00:00 2005/08/19 becomes 2005-08-19T00:00:00 08/19/2005 becomes 2005-08-19T00:00:00</p> <p>The query to obtain transactions in the date range is constructed as follows: (Date DT >=BeginDt) AND (Date DT<EndDt) where Date DT is the transaction timestamp</p>
EndDt	<p>Required, except when PNRef is provided. The end date of the date range in MM/DD/YYYY (or YYYY-MM-DD or YYYY-MM-DDThh:mm:ss) format. This date will be converted to YYYY-MM-DDThh:mm:ss (time is in 24-hour format). If the passed-in EndDt does not contain time information, EndDt will be incremented to the <i>next day</i> at midnight such that no transaction on the desired end date will be excluded based on its time. For example:</p> <p>2005-08-19T12:00:12 is kept as is 2005-08-19 becomes 2005-08-20T00:00:00 08/19/2005 becomes 2005-08-20T00:00:00</p>
PaymentType	<p>Optional. If provided, only those transactions matching the PaymentType will be included. Valid values are:</p> <ul style="list-style-type: none"> • 'ACH' Automated Clearing House • 'ECHECK' Electronic Check • 'GUARANTEE' Guarantee check • 'PAYRECEIPT' to retrieve receipt images that were uploaded to the payment server • 'SETTLE' to retrieve requests to settle transactions

	<ul style="list-style-type: none"> • 'VERIFY' to retrieve pre-authorized checks <p>Or any permutation of the above values, e.g. ""ACH','ECHECK"" will pull all transactions with either ACH or ECHECK payment types</p>
ExcludePaymentType	Optional. If provided, any transaction matching the ExcludePaymentType will be excluded
TransType	<p>Optional. If provided, only those transactions matching the TransType will be included. Valid values are:</p> <ul style="list-style-type: none"> • 'Authorization' to retrieve previously-authorized (pre-auth) transactions • 'Capture' to retrieve captured transactions • 'Credit' to retrieve return transactions • 'ForceCapture' to retrieve force-auth transactions • 'GetStatus' to make an inquiry to the EBT or gift card's balance • 'PostAuth' to retrieve post-auth transactions • 'Purged' to remove a transaction from the current batch due to an error • 'Receipt' to retrieve receipt images that were uploaded to the payment server • 'RepeatSale' to retrieve repeat-sale transactions • 'Sale' to retrieve sale transactions • 'Void' to retrieve void transactions <p>Or any permutation of the above values, e.g. ""Credit','Sale"" will pull all transactions with either Credit or Sale transaction types</p>
ExcludeTransType	Optional. If provided, any transaction matching the ExcludeTransType will be excluded
ApprovalCode	Optional. If provided, only those transactions matching the ApprovalCode parameter will be included
Result	<p>Optional. If provided, only those transactions matching the Result will be included. Valid values are:</p> <ul style="list-style-type: none"> • 0 is Approved • Anything else is declined, if you want all the declined transactions, you should leave this field empty and use the ExcludeResult=0 instead
ExcludeResult	Optional. If provided, any transactions matching the ExcludedResult will be excluded
NameOnCheck	Optional. Check owner's name as it appears on the check, if provided. Only those transactions with check owner's name matching NameOnCheck will be included. Matching is done using wild cards: e.g. "test" will match "test", "1test" and "1test234"

CheckNum	Optional. Check number. If provided, only those transactions with matching CheckNum will be included
AcctNum	Optional. Check account number. If provided, only those transactions matching the AcctNum will be included. Matching is done using wild cards
RouteNum	Optional. If provided, any transactions matching the RouteNum (Transit Number) will be excluded. Matching is done using wild cards
ExcludeVoid	Optional. Whether to exclude voided transactions; must either be TRUE or FALSE . The default value is TRUE
User	Optional. The user who originated the transactions. If provided, only those transactions created by the matching User will be included. Matching is done using wild cards
InvoiceId	Optional. The invoice ID that was included in the original transaction. If provided, only those transactions with matching invoiceId will be included. Matching is done using wild cards
SettleFlag	Optional. Whether the transaction was settled; must either be 1 for true or 0 for false
SettleMsg	Optional. The settlement ID or message returned from the host
SettleDt	Optional. The settlement timestamp
TransformType	<p>Optional. The type of format to transform the data into. Leave the field blank to default to XML</p> <ul style="list-style-type: none"> • XML will output the plain XML string • XSL will use XSL to transform the XML output • DELIM uses ColDelim and RowDelim to format the output
Xsl	<p>Optional. This field is used only if the TransformType is XSL. The XSL to transform the resulting dataset; if provided, the resulting dataset will be transformed using this XSL. You may pass in a URL to the XSL file, or the XSL string itself. If this field is not empty, the Web Services will try to locate the file from the URL. If that also fails, it will treat it as a XSL string. In any case, the final XSL string will be loaded and validated against the XSL schema; if it passes, then that XSL will be used for transformation. A sample predefined XSL is included with this Web Services:</p> <ul style="list-style-type: none"> • https://secure.redfinnet.com/admin/ws/TabDelim.xsl for a tab delimited transformation.
ColDelim	Optional. This field is used only if the TransformType is DELIM . This defines the string that separates each column
RowDelim	Optional. This field is used only if the TransformType is DELIM . This defines the string that separates each row
IncludeHeader	Optional. This field is used only if the TransformType is DELIM . If TRUE , then field headers will be included in the first row using the same delimiter strings; must either be TRUE or FALSE
ExtData	<p>Optional. Extended data in XML format. Valid values are:</p> <ul style="list-style-type: none"> • <IMAGE_TYPE>NO_IMAGE</IMAGE_TYPE> for no image

	<ul style="list-style-type: none"> • <IMAGE_TYPE>ONLY_IMAGE</IMAGE_TYPE> for only the image • <IMAGE_TYPE>ALL</IMAGE_TYPE> for all images • <CustomerID>CustomerID</CustomerID> for customer ID • <Amount>Amount</Amount> Total amount to search transactions for in DDDD.CC format. • <RegisterNum>RegisterNum</RegisterNum> Register number, originally passed with the transaction, to search transactions for
--	---

Example

The following table contains sample data you can use to test this Web service. The **UserName**, **Password**, and **RPNum** parameters should be changed when testing this example yourself. The example data shown will create a list of approved Verify check transactions that were processed between 1/1/2003 and 12/31/2003. The output data is in XML format because the **TransformType** parameter was not specified.

Parameter	Value
UserName	Test
Password	123
RPNum	1
BeginDt	1/1/2003
EndDt	12/31/2003
PaymentType	'VERIFY'
Result	0

Result:

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

```
<string xmlns="https://secure.redfinnet.com/admin/ws">
```

```

<RichDBDS> <TrxDetailCheck> <RichDBDS> <TrxDetailCheck>
<TRX_HD_Key>25</TRX_HD_Key> <Invoice_ID />
<Date_DT>2003-12-05T13:47:15.1230000-08:00</Date_DT>
<Merchant_Key>1</Merchant_Key> <User_Name_VC>test
</User_Name_VC> <Register_Number_CH> </Register_Number_CH>
<Reseller_Key>100</Reseller_Key> <Payment_Type_ID>VERIFY
</Payment_Type_ID> <Trans_Type_ID>Authorization </Trans_Type_ID>
<Processor_ID>RMRS </Processor_ID>
<Last_Update_DT>2003-12-05T13:47:15.1230000-08:00</Last_Update_D
T> <TRX_Check_Key>5</TRX_Check_Key> <CheckNum_CH>1001
    
```

```
</CheckNum_CH> <MICR_VC />
<AccountNum_VC>*****7890</AccountNum_VC>
<TransitNum_VC>123456780</TransitNum_VC> <RawMICR_VC />
<DL_VC /> <SS_CH /> <DOB_CH> </DOB_CH> <StateCode_CH>DE
</StateCode_CH> <NameOnCheck_VC>Jane Doe</NameOnCheck_VC>
<EMail_VC /> <Phone_VC /> <Amount_MN>1.0000</Amount_MN>
<Result_CH>0 </Result_CH> <Result_Txt_VC>&lt;INPUT TYPE=HIDDEN
NAME=PARAM_RESPONSE_CODE VALUE="AA"&gt;&lt;INPUT TYPE=HIDDEN
NAME=PARAM_AUTH_SOURCE VALUE="0"&gt;&lt;INPUT TYPE=HIDDEN
NAME=PARAM_CAP_CODE VALU</Result_Txt_VC>
<Host_Approval_CH>963852 </Host_Approval_CH> <Host_Ref_Num_CH>
</Host_Ref_Num_CH> <Result_Msg_VC>APPROVAL
963852</Result_Msg_VC> <Result_Msg1_VC /> <Result_Msg2_VC />
<Host_Date_CH>120503 </Host_Date_CH> <Host_Time_CH>164502
</Host_Time_CH> <IP_VC>192.168.2.22</IP_VC> </TrxDetailCheck>
<TrxDetailCheck> <TRX_HD_Key>24</TRX_HD_Key> <Invoice_ID />
<Date_DT>2003-12-05T13:46:50.0770000-08:00</Date_DT>
<Merchant_Key>1</Merchant_Key> <User_Name_VC>test
</User_Name_VC> <Register_Number_CH> </Register_Number_CH>
<Reseller_Key>100</Reseller_Key> <Payment_Type_ID>VERIFY
</Payment_Type_ID> <Trans_Type_ID>Authorization </Trans_Type_ID>
<Processor_ID>RMRS </Processor_ID>
<Last_Update_DT>2003-12-05T13:46:50.0770000-08:00</Last_Update_D
T> <TRX_Check_Key>4</TRX_Check_Key> <CheckNum_CH>4877
</CheckNum_CH> <MICR_VC />
<AccountNum_VC>*****7890</AccountNum_VC>
<TransitNum_VC>123456780</TransitNum_VC> <RawMICR_VC />
<DL_VC /> <SS_CH /> <DOB_CH> </DOB_CH> <StateCode_CH>WA
</StateCode_CH> <NameOnCheck_VC>John Doe</NameOnCheck_VC>
<EMail_VC /> <Phone_VC /> <Amount_MN>2.0000</Amount_MN>
<Result_CH>0 </Result_CH> <Result_Txt_VC>&lt;INPUT TYPE=HIDDEN
NAME=PARAM_RESPONSE_CODE VALUE="AA"&gt;&lt;INPUT TYPE=HIDDEN
NAME=PARAM_AUTH_SOURCE VALUE="0"&gt;&lt;INPUT TYPE=HIDDEN
NAME=PARAM_CAP_CODE VALU</Result_Txt_VC>
<Host_Approval_CH>963852 </Host_Approval_CH> <Host_Ref_Num_CH>
</Host_Ref_Num_CH> <Result_Msg_VC>APPROVAL
963852</Result_Msg_VC> <Result_Msg1_VC /> <Result_Msg2_VC />
<Host_Date_CH>120503 </Host_Date_CH> <Host_Time_CH>164437
</Host_Time_CH> <IP_VC>192.168.2.22</IP_VC> </TrxDetailCheck> </
RichDBDS>
```

</string>

GetCardType

This Web service operation returns the name of the card issuer, such as Visa, MasterCard, Amex:

<https://secure.redfinnet.com/SmartPayments/validate.aspx?op=GetCardType>

Parameter	Description
CardNumber	Required. The number of a credit card

Example

The following table contains sample data you can use to test this Web service.

Parameter	Value
CardNumber	5454545454545454

Result:

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

```
<string xmlns="https://secure.redfinnet.com/SmartPayments/">MASTERCARD</string>
```

GetInfo

This Web service retrieves information pertaining to the transaction type (TransType) specified:

<https://secure.redfinnet.com/SmartPayments/transact.asmx?op=GetInfo>.

Parameter	Description
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
TransType	<p>Required. Valid values are:</p> <ul style="list-style-type: none"> • BatchInquiry returns a comma delimited list in a single XML tag that contains the summarized transaction dollar amount and transaction count for each payment method in the current batch. The list is in the following format: <i>Payment_Method1=0.00, Payment_Method2=0.00</i> • Setup returns a comma delimited list in a single XML tag that contains merchant setup information. The list is in the following format: <i>Setup_Name1=Y N, Setup_Name2=Y N</i> • StatusCheck returns "OK" if a connection can be made to the payment server with the supplied user name and password; otherwise, an error message is returned • Initialize returns the merchant account setup, including Partner number, Merchant ID, credit card type, phone number, etc.
ExtData	<p>Optional. Extended data in XML format. Valid values are:</p> <ul style="list-style-type: none"> • <TrainingMode>T</TrainingMode> an indicator that specifies transactions will be processed for local loop back testing • <TrainingMode>F</TrainingMode> an indicator that specifies transactions will not be processed for local loop back testing • <BatchSequenceNum>Number</BatchSequenceNum> used when the TransType is BatchInquiry and it is a number that indicates which previous batch or current batch the Payment Server should query from the processor in order to get information about the batch. Currently only support with Global Payments. Valid values are: <ul style="list-style-type: none"> • 0 = Current open batch (default value if BatchSequenceNum is not specified in ExtData) • 1 = previous batch • 2 = the batch before the previous batch specified with the value 1 • N = and so on...

Examples

The following examples show the results of testing four **TransTypes**. When testing each example yourself, the **UserName** and **Password** parameters should be changed.

Example 1: BatchInquiry

Parameter	Value
UserName	Test
Password	123
TransType	BatchInquiry

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/"><Result>0</Result>
<RespMSG>Approved</RespMSG><ExtData>Credit_Sale_Amount=28.00,Credit_Sale_Count=28,Credit_Return_Amount=7.00,Credit_Return_Count=7,Credit_Net_Amount=21.00,Credit_Net_Count=35,Debit_Sale_Amount=9.00,Debit_Sale_Count=6,Debit_Return_Amount=1.00,Debit_Return_Count=1,Debit_Net_Amount=8.00,Debit_Net_Count=7,Check_Sale_Amount=0.00,Check_Sale_Count=0,Check_Net_Amount=0.00,Check_Net_Count=0</ExtData></Response>
```

Example 2: Setup

Parameter	Value
UserName	Test
Password	123
TransType	Setup

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/"><Result>0</Result>
<RespMSG>Approved</RespMSG><ExtData>Force_Duplicates=N,Auto_Close_Batch=Y,Industry=R,DEBIT=Y,AMEX=Y,CARTBLANCH=Y,DINERS=Y,DISCOVER=Y,JAL=Y,JCB=Y,MASTERCARD=Y,VISA=Y,EBT=Y</ExtData></Response>
```

Example 3: StatusCheck

Parameter	Value
UserName	Test
Password	123
TransType	StatusCheck

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/"><Result>0</Result>
  <RespMSG>Approved</RespMSG><ExtData>OK</ExtData></Response>
```

Example 4: Initialize

Parameter	Value
UserName	Test
Password	123
TransType	Initialize

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><RespMSG>Approved</RespMSG><ExtData><Partner>110<
  /Partner> <Vendor>206</Vendor> <MerchantID /> <PinPadKeyManagement
  /><LiveURL>https://www.YourWebSite.com</LiveURL><LiveURL1>https://w
  ww1.YourWebSite.com</LiveURL1><TestURL>https://test.YourWebSite.com
  </TestURL><TestURL1>https://test.YourWebSite.com</TestURL1><EPSPay>/
  pay/payxml.aspx</EPSPay><EPSLogin>/Admin/login.aspx</EPSLogin><Phone
  1>425-123-1234</Phone1><Phone2>425-123-1234</Phone2><Auto_Close_Ba
  tch>N</Auto_Close_Batch><eCheck>Y</eCheck><CreditCard>Y</CreditCard><P
  aymentTypes><CardType>Amex</CardType><CardType>CartBlanch</CardType
  ><CardType>Diners</CardType><CardType>Discover</CardType><CardType>J
  AL</CardType><CardType>JCB</CardType><CardType>MasterCard</CardType
  ><CardType>Visa</CardType></PaymentTypes><ExpressPay><CardType>Visa
  </CardType><LTAmount>25</LTAmount><Amount>0</Amount><Entry
  Method>S</EntryMethod><ProcessingRule>On-
  Line</ProcessingRule><PrintReceipt>N</PrintReceipt></ExpressPay
  ></ExtData>
</Response>
```

GetOpenBatchSummary

This Web service operation retrieves a payment type transaction summary of the current open batch for a merchant:

<https://secure.redfinnet.com/admin/ws/trxdetail.aspx?op=GetOpenBatchSummary>.

Parameter	Value
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
RPNum	Required. The number uniquely identifies each merchant
BeginDt	Optional. The begin date of the date range in MM/DD/YYYY format. This date will be converted to: MM/DD/YYYYT00:00:00:0000AM
EndDt	Optional. The end date of the date range in MM/DD/YYYY format. This date will be converted to: MM/DD/YYYYT12:59:59:9999PM
ExtData	Currently there is no data value available

Example

The following table contains sample data you can use to test this Web service. The **UserName**, **Password**, and **RPNum** parameters should be changed when testing this example yourself. The example data shown will create a categorized summary list of all transactions in the current open batch.

Parameter	Value
UserName	Test
Password	123
RPNum	1

Result:

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

```
<string xmlns="https://secure.redfinnet.com/Admin/ws"><OpenBatchSummary>
  <Table> <Payment_Type_ID>DEBIT </Payment_Type_ID>
  <Authorization>0</Authorization> <Capture>0</Capture>
  <ForceCapture>0</ForceCapture> <PostAuth>0</PostAuth>
  <Return>7.0000</Return> <Sale>61.0000</Sale>
  <Receipt>0</Receipt> <RepeatSale>0</RepeatSale>
  <Authorization_Cnt>0</Authorization_Cnt>
  <Capture_Cnt>0</Capture_Cnt>
```

```

<ForceCapture_Cnt>0</ForceCapture_Cnt>
<PostAuth_Cnt>0</PostAuth_Cnt> <Return_Cnt>7</Return_Cnt>
<Sale_Cnt>58</Sale_Cnt> <Receipt_Cnt>0</Receipt_Cnt>
<RepeatSale_Cnt>0</RepeatSale_Cnt> <Cnt>65</Cnt> </Table>
<Table> <Payment_Type_ID>EBT </Payment_Type_ID>
<Authorization>0</Authorization> <Capture>0</Capture>
<ForceCapture>0</ForceCapture> <PostAuth>0</PostAuth>
<Return>132.0000</Return> <Sale>150.0000</Sale>
<Receipt>0</Receipt> <RepeatSale>0</RepeatSale>
<Authorization_Cnt>0</Authorization_Cnt>
<Capture_Cnt>0</Capture_Cnt>
<ForceCapture_Cnt>0</ForceCapture_Cnt>
<PostAuth_Cnt>0</PostAuth_Cnt> <Return_Cnt>15</Return_Cnt>
<Sale_Cnt>24</Sale_Cnt> <Receipt_Cnt>0</Receipt_Cnt>
<RepeatSale_Cnt>0</RepeatSale_Cnt> <Cnt>39</Cnt> </Table>
<Table> <Payment_Type_ID>MASTERCARD</Payment_Type_ID>
<Authorization>0</Authorization> <Capture>0</Capture>
<ForceCapture>0</ForceCapture> <PostAuth>0</PostAuth>
<Return>0</Return> <Sale>4.0000</Sale> <Receipt>0</Receipt>
<RepeatSale>7.0000</RepeatSale>
<Authorization_Cnt>0</Authorization_Cnt>
<Capture_Cnt>0</Capture_Cnt>
<ForceCapture_Cnt>0</ForceCapture_Cnt>
<PostAuth_Cnt>0</PostAuth_Cnt> <Return_Cnt>0</Return_Cnt>
<Sale_Cnt>4</Sale_Cnt> <Receipt_Cnt>0</Receipt_Cnt>
<RepeatSale_Cnt>7</RepeatSale_Cnt> <Cnt>11</Cnt> </Table>
<Table> <Payment_Type_ID>VISA </Payment_Type_ID>
<Authorization>0</Authorization> <Capture>0</Capture>
<ForceCapture>1.0000</ForceCapture> <PostAuth>1.0000</PostAuth>
<Return>0</Return> <Sale>19.0000</Sale> <Receipt>0</Receipt>
<RepeatSale>0</RepeatSale>
<Authorization_Cnt>0</Authorization_Cnt>
<Capture_Cnt>0</Capture_Cnt>
<ForceCapture_Cnt>1</ForceCapture_Cnt>
<PostAuth_Cnt>1</PostAuth_Cnt> <Return_Cnt>0</Return_Cnt>
<Sale_Cnt>19</Sale_Cnt> <Receipt_Cnt>0</Receipt_Cnt>
<RepeatSale_Cnt>0</RepeatSale_Cnt> <Cnt>21</Cnt> </Table>
</OpenBatchSummary></string>
    
```

IsCommercialCard

This Web service operation checks whether the card number entered is for a commercial card or not:

<https://secure.redfinnet.com/SmartPayments/validate.aspx?op=IsCommercialCard>.

Please note due to ever-changing card bin-ranges and other factors, not all Commercial/Purchase cards can be definitely determined by this method. Developers should design their applications while keeping this fact in mind.

Parameter	Description
CardNumber	Required. The number of a credit card

Examples

The following tables contain sample data you can use to test this Web service. The return result will be either true or false.

Example 1: The example data shown will result in return false.

Parameter	Value
CardNumber	5454545454545454

Result:

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

```
<boolean
  xmlns="https://secure.redfinnet.com/SmartPayments/">false</boolean>
```

Example 2: The example data shown will result in return true.

Parameter	Value
CardNumber	4055011111111111

Result:

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

```
<boolean xmlns="https://secure.redfinnet.com/SmartPayments/">true</boolean>
```

ProcessCheck

This Web service operation processes check transactions for a merchant:

<https://secure.redfinnet.com/SmartPayments/transact.aspx?op=ProcessCheck>

Parameter	Value
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
TransType	<p>Required. Type of the check transaction. Valid values are:</p> <ul style="list-style-type: none"> • Sale to make a purchase with a check • Auth (Verify) to authorize/verify an amount of a check • Return to return the money of a settled check transaction to the check holder • Void to undo an unsettled check transaction • Force to force a previous Sale transaction into the current batch (ForceSale) • Capture to settle a single transaction in the current batch; only for terminal-based processors • CaptureAll to settle all transactions in the current batch; only for terminal-based processors
CheckNum	Required except for these TransType 's: Void, Capture, CaptureAll . Check number uniquely identifies each individual check
TransitNum	Required except for these TransType 's: Void, Capture, CaptureAll . Transit number uniquely identifies a bank routing number
AccountNum	Required except for these TransType 's: Void, Capture, CaptureAll . Account number uniquely identifies a check holder's bank account number
Amount	Required except for these TransType 's: Void, Capture, CaptureAll . The total transaction amount in DDDD.CC format
MICR	Optional. The Magnetic Ink Check Reader data line, which includes TransitNum , and AccountNum . Required for processing Check-Present transactions
NameOnCheck	Required except for these TransType 's: Void, Capture, CaptureAll . The check holder's name as it appears on the check. The parameter may be required, depending on the merchant's processor setup. This parameter will remove invalid characters. See list of Removed Characters for more details
DL	Optional. The check holder's driver's license number. This parameter will remove invalid characters. See list of Removed Characters for more details
SS	Optional. The check holder's Social Security Number. This parameter will remove invalid characters. See list of Removed Characters for more details
DOB	Optional. The check holder's date of birth. This parameter will remove invalid characters. See list of Removed Characters for more details

<p>StateCode</p>	<p>Optional. The check holder's 2 character state code. The parameter may be required depending on the merchant's processor setup. This parameter will remove invalid characters. See list of Removed Characters for more details</p>
<p>CheckType</p>	<p>Optional. The type of the check. Valid values are:</p> <ul style="list-style-type: none"> • Personal • Corporate • Government
<p>ExtData</p>	<p>Extended data in XML format</p> <p>These tags may be required for Sale and Return transactions depending on the merchant's processor setup: CityOfAccount, BillToStreet, and BillToPostalCode.</p> <p>Required tag for Return, Void, Force, and Capture transactions is: PNRef.</p> <p>RawMICR tag is required for processing Check-Present transactions.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> • <TimeOut>TimeOut</TimeOut> for timeout value in seconds (default = 30) • <PNRef>PNRef</PNRef> for a reference number assigned by the payment server • <Phone>Phone</Phone> for phone number. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <Email>EMail</EMail> for email address. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <RawMICR>RawMICR</RawMICR> for raw Magnetic Ink Check Reader data from the check reader in the format of: <i>TransitNumTAccountNumOCheckNum</i> • <InvNum>InvNum</InvNum> for invoice tracking number. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <TrainingMode>TrainingMode</TrainingMode> to process transaction in Training Mode; either T or F • <AllianceNum>AllianceNum</AllianceNum> is the Alliance number for the check • <AccountType>AccountType</AccountType> is the type bank account for the check. Valid values are Checking or Saving • <CityOfAccount>CityOfAccount</CityOfAccount> for city name of the check holder's residential address. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <BillToStreet>BillToStreet</BillToStreet> for street name of the check

	<p>holder's billing address. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details.</p> <ul style="list-style-type: none"> • <BillToCity>BillToCity</BillToCity> for city name of the check holder's billing address. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <BillToState>BillToState</BillToState> for the two character state code of the check holder's billing address. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <BillToPostalCode>BillToPostalCode</BillToPostalCode> for zip code of the check holder's billing address. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <BillToCountry>BillToCountry</BillToCountry> for country name of the check holder's billing address. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <CustomerID>CustomerID</CustomerID> for customer ID. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details
--	--

The following tables contain sample data you can use to test this Web service. The **UserName** and **Password** parameters should be changed when testing the examples yourself.

Examples

Example 1: The example data below will process a manually entered check **Sale** transaction through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	Sale
CheckNum	1001
TransitNum	123456780
AccountNum	1234567890
Amount	100.00
NameOnCheck	John Doe
StateCode	WA
ExtData	<CityOfAccount>City</CityOfAccount> <BillToStreet>123</BillToStreet> <BillToPostalCode>99999</BillToPostalCode>

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><RespMSG>Approved</RespMSG><Message>APPROVAL</
  Message><AuthCode>GUEIJQ</AuthCode><PNRef>2400</PNRef>
```

```
</Response>
```

Example 2: The example data below will process a swiped check **Sale** transaction through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	Sale
CheckNum	1001
TransitNum	123456780
AccountNum	1234567890
Amount	100.00
MICR	1234567801234567890
NameOnCheck	John Doe
StateCode	WA
ExtData	<RawMICR>123456780T1234567890O1001</RawMICR>

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><Message>APPROVAL</Message><AuthCode>AUTH NUM
  121-704</AuthCode><PNRef>11622</PNRef>
```

```
</Response>
```

Example 3: The example data below will process a manually-entered check **Return** transaction through the payment server. The **PNRef** element in the **ExtData** parameter should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	Return
CheckNum	100
TransitNum	123456780
AccountNum	1234567890
Amount	100.00
NameOnCheck	John Doe
StateCode	WA
ExtData	<PNRef>821</PNRef><CityOfAccount>Any Town</CityOfAccount><BillToStreet>123</BillToStreet><BillToPostalCode>99999</BillToPostalCode><InvNum>1001</InvNum>

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://
  secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><RespMSG>Approved</RespMSG><Message>APPROVAL</Message>
<AuthCode>GWNICP</AuthCode><PNRef>2406</PNRef><ExtData>InvNum=1001</ExtData>
```

```
</Response>
```

Example 4: The example data below will process a check **Void** transaction through the payment server. The **PNRef** element in the **ExtData** parameter should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	Void

ExtData	<PNRef>2405</PNRef>
----------------	---------------------

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
  <Result>0</Result><RespMSG>Approved</RespMSG><Message>APPROVAL</
  Message><AuthCode>GW5NPQ</AuthCode><PNRef>2405</PNRef>
```

```
</Response>
```

ProcessCreditCard

This Web service operation processes credit card transactions for a merchant:

<https://secure.redfinnet.com/SmartPayments/transact.aspx?op=ProcessCreditCard>.

Parameter	Value
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
TransType	<p>Required. Type of the credit card transaction. Valid values are:</p> <ul style="list-style-type: none"> • Sale to make a purchase on a credit card • Adjustment is used to modify an existing tip amount for an original sale. This applies to the processors that support restaurant adjustment transactions • Auth to authorize an amount on a credit card • Return to credit the cardholder's account • Void to undo an unsettled transaction. Note: pass the Card Number and ExpDate with null values on voids • Force to place an Auth transaction into the current batch (PostAuth) or to place a transaction not processed through the payment server into the current batch (ForceAuth) • Capture to settle a single transaction in the current batch; only for terminal-based processors • CaptureAll to settle all transactions in the current batch; only for terminal-based processors or host-based processors that support a batch release feature • RepeatSale to perform a recurring billing or installment payment transaction
CardNum	Optional except for these TransType's : Sale, Auth, Return, Force (ForceAuth). Credit card number to process the transaction. For all other transaction types the parameter needs to be included
ExpDate	Optional except for these TransType's : Sale, Auth, Return, Force (ForceAuth). Credit card's expiration date in MMY format. For all other transaction types the parameter needs to be included
MagData	<p>Optional except when processing swiped card transactions. Data located on the track 2 of the magnetic strip of the card. Once this field is populated, the transaction will be indicated as <i>Card-Present</i> transaction and usually result in a more favorable retail discount rate. This parameter will remove invalid characters. See list of Removed Characters for more details</p> <p>The format of the MagData (or Track 2 data) is CardNum=ExpDate followed by the service code and checksum. For example, 36438999960016=05121015432112345678</p>
NameOnCard	Optional, depending on different merchant processor setups. The cardholder's name as it appears on the card. This parameter will remove invalid characters. See list of Removed Characters for more details

Amount	Optional except for these TransType 's: Auth, Sale, Return, Force (both PostAuth and ForceAuth). The total transaction amount in DDDD.CC format
InvNum	Optional. Invoice tracking number. This parameter will remove invalid characters. See list of Removed Characters for more details
PNRef	Optional except for these TransType 's: Void, Force (PostAuth), Capture . Reference number assigned by the payment server
Zip	Optional depending on different merchant processor setups. Cardholder's billing address zip code used in address verification. This parameter will remove invalid characters. See list of Removed Characters for more details
Street	Optional depending on different merchant processor setup. Cardholder's billing street address used in address verification. This parameter will remove invalid characters. See list of Removed Characters for more details
CVNum	Optional. Card verification number
ExtData	<p>Optional except in the cases of: AuthCode (required for a Force (ForceAuth) transaction); City and BillToState (required by certain processors); Invoice and associated nested data elements (required for Concord EFS Purchase Card Level 3 and Fuel purchases- see section below). Extended data in XML format. Valid values are:</p> <ul style="list-style-type: none"> • <AuthCode>ApprovalCode</AuthCode> for original authorization code • <CustCode>CustomerCode</CustCode> for customer code or PO number of the customer. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details (Note* This is the tag to be passed for Level II information for Global Payments only, please use PONUM for the other processors that support level II.) • <TipAmt>TipAmount</TipAmt> for tip amount in DDDD.CC format • <TaxAmt>TaxAmount</TaxAmt> for tax amount in DDDD.CC format • <SequenceNum>SequenceNum</SequenceNum> for sequence number used with RepeatSale installment transactions; this designates which number in the sequence the transaction is; it must be a positive integer smaller than or equal to the SequenceCount • <SequenceCount>SequenceCount</SequenceCount> for sequence count used with RepeatSale installment transactions; this designates the total number of charges that will be made; it must be a positive integer greater than or equal to the SequenceNum • <ServerID>ServerID</ServerID> for a unique server identification number. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <TimeOut>TimeOut</TimeOut> for timeout value in seconds (default = 40) • <TrainingMode>TrainingMode</TrainingMode> to process transaction in Training Mode; either T or F • <Force>Force</Force> for forcing duplicate transactions to be processed; either T or F. Note that some processors, including Concord EFS, will not utilize this tag and may still reject a duplicate transaction • <RegisterNum>RegisterNum</RegisterNum> for register number. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <City>City</City> for the city name of the cardholder's billing address. The

data within this XML tag parameter will remove invalid characters. See list of [Removed Characters](#) for more details

- **<BillToState>State</BillToState>** for the two character state code of the cardholder's billing address. The data within this XML tag parameter will remove invalid characters. See list of [Removed Characters](#) for more details
- **<CustomerID>CustomerID</CustomerID>** for customer ID
- **<PONum>PONum</PONum>** for purchase order number. The data within this XML tag parameter will remove invalid characters. See list of [Removed Characters](#) for more details (Note* This is to be used for Level II information except Global Payments)
- **<BillPayment>BillPayment</BillPayment>** to indicate that a transaction is a utility bill payment; either **T** or **F**; only supported for **TransType's Sale** and **RepeatSale**; This tag is only relevant to Retail, MOTO, and eCommerce markets. Currently, this information is only supported for Vital, First Data North, and Global Payments processors; other processors may be supported in the future
- **<Authentication>**

<XID>AuthenticationID</XID> Verified By VISA

<CAVV>CAVV</CAVV> CAVV response value

<UCAF>UCAF</UCAF> Universal Card Holder Authentication Field

Verified by Visa and Universal Cardholder Authentication Field are programs implemented by Visa and MasterCard respectively to verify that an account number is being submitted by the cardholder. These programs are for the Ecommerce market exclusively and only MasterCard and Visa cards support this feature. The data for Visa and MasterCard is different. There is a possible CAVV response for Visa cards that will be returned via ExtData * Please see sample in the section Examples for Verified by VISA and UCAF for Mastercard.

<TrustAttempt>T</TrustAttempt>

The trust attempt tag will indicate VBV or MS was attempted but no XID, CAVV, or UCAFF is available for the transaction.

</Authentication>

- **<CVPresence>CVPresence Value</CVPresence>** CVV2 / CVC2 / CID Presence, indicates whether a CVV2 or CID has been sent along with the request. The valid values for this tag are: **None, NotSubmitted, Submitted Illegible, NotPresent (Not present on card)**. *Please see sample request and response for this **CVPresence tag field**.
- **<EntryMode>EntryModeValue</EntryMode>** this was added to indicate how the values for the payment information were obtained. The Valid values for this tag are: UNKNOWN, MANUAL, MAGNETIC STRIPE, ICC, and PROXIMITY
- **<Invoice>Invoice</Invoice>** to indicate invoice details will be included. Required for Concord EFS Purchase Card Level 3 but optional for fuel purchases. However, fuel purchases must contain the **<Items>** element for transactions of type **Sale** and **Force**. See below for hierarchy of required and optional elements

nested within. Please note that all elements included inside **<Invoice>** must be in the specific order listed below

❖ **<Invoice>**

- **<InvNum>***InvNum***</InvNum>** Purchase invoice number
- **<Date>***Date***</Date>** Date of invoice in YYMMDD format for Concord
- **<BillTo>**
 - **<CustomerId>***CustomerId***</CustomerId>** Customer ID number
 - **<Name>***Name***</Name>** Customer name
 - **<Address>** Customer address
 - **<Street>***Street***</Street>** Customer address street
 - **<City>***City***</City>** Customer address city
 - **<State>***State***</State>** Customer address state
 - **<Zip>***Zip***</Zip>** Customer address zip code
 - **<Country>***Country***</Country>** Customer address country
 - **</Address>**
 - **<Email>***Email***</Email>** Customer email
 - **<Phone>***Phone***</Phone>** Customer phone number
 - **<Fax>***Fax***</Fax>** Customer fax number
 - **<CustCode>***CustCode***</CustCode>** Customer code
 - **<PONum>***PONum***</PONum>** Purchase order number from customer
 - **<TaxExempt>***TaxExempt***</TaxExempt>** Customer tax exempt status
- **</BillTo>**
- **<Description>***Description***</Description>** Description of purchase
- **<Items>** Required for Concord EFS Purchase Card Level 3 and fuel purchases. Items contained in invoice. Contains one or more **<Item>** elements
 - **<Item>** Required for Concord EFS Purchase Card Level 3 and fuel card purchases. One item in invoice (item details nested within). There may be multiple **<Item>** nested within **<Items>** tag
 - **<SKU>***SKU***</SKU>** SKU number of item
 - **<UPC>***UPC***</UPC>** Required for Concord EFS Purchase Card Level 3 and fuel purchases. UPC number of item for Purchase Card Level

3 or the NACS (National Association of Convenience Stores) product code for fuel purchases. The NACS is an industry standard list that Concord is utilizing. For a list of NACS product codes, please contact EFSnet™ customer support at support@concordefsn.net or 1-877-852-2637.

- **<Description>***Description***</Description>** Item description
- **<Quantity>***Quantity***</Quantity>** Required for Concord EFS Purchase Card Level 3 and fuel purchases. Quantity purchased of item
- **<UnitOfMeasurement>***UnitOfMeasure***</UnitOfMeasurement>** Unit of measurement for item
- **<UnitPrice>***UnitPrice***</UnitPrice>** Required for Concord EFS Purchase Card Level 3 and fuel purchases. Unit price of item
- **<DiscountAmt>***DiscountAmount***</DiscountAmt>** Possible discount amount on item
- **<TaxAmt>***TaxAmt***</TaxAmt>** Possible tax amount on item
- **<TotalAmt>***TotalAmt***</TotalAmt>** Total amount of item
- **<Category>***Category***</Category>** Required for Concord EFS Purchase Card Level 3 and fuel purchases. Item category for Purchase Card Level 3 or the specific value **Fuel** to designate a fuel purchase item
- **<TaxRate>***TaxRate***</TaxRate>** Possible tax rate applied to item

▪ **</Item>**

➤ **</Items>**

➤ **<DiscountAmt>***DiscountAmount***</DiscountAmt>** Possible total discount for invoice

➤ **<ShippingAmt>***ShippingAmt***</ShippingAmt>** Possible shipping amount for invoice

➤ **<DutyAmt>***DutyAmt***</DutyAmt>** Possible duty amount for invoice

➤ **<TaxAmt>***TaxAmt***</TaxAmt>** Possible tax amount for invoice

➤ **<NationalTaxInc>***NationalTaxInc***</NationalTaxInc>** Possible additional tax amount included in invoice total

➤ **<TotalAmt>***TotalAmt***</TotalAmt>** Total amount of the transaction on the invoice

❖ **</Invoice>**

- **<Fleet>***Fleet***</Fleet>** Required for Concord EFS Fleet card purchases. Information on fleet member making purchase. See below for hierarchy of elements nested within. Please note that all elements included inside **<Fleet>** must be in the specific order listed below

	<ul style="list-style-type: none"> ❖ <Fleet> <ul style="list-style-type: none"> ➤ <VehicleNum><i>VehicleNum</i></VehicleNum> May be required for specific Concord EFS Fleet card purchases. The vehicle number ➤ <DriverNum><i>DriverNum</i></DriverNum> May be required for specific Concord EFS Fleet card purchases. The vehicle driver's number ➤ <OdometerReading><i>OdometerReading</i></OdometerReading> May be required for specific Concord EFS Fleet card purchases. The current odometer reading of the fleet vehicle ❖ </Fleet> ❖ <CardType><i>CardType</i></CardType> <ul style="list-style-type: none"> ➤ When TransType does not equal Capture or CaptureAll: Required for manually-entered fleet card transactions that have the ISO prefix of the fleet card present only in the track data and not in the embossed data on the front of the card. Valid values are WEX and Voyager. Concord currently does not support manually entered Voyager cards ➤ When TransType equals CaptureAll: Optional valid values can be: ALL to specify all payment methods assigned to the merchant account should be settled, or a combination of the specific payment methods separated by a colon (i.e. CREDIT:DEBIT:EBT:EGC) in order to specify which individual payment methods assigned to the merchant account should be settled. Please note that if the processor requires all payment methods to settle at the same time, it is required to use the ALL value or the appropriate combination of the specific payment methods in order to settle the account correctly. Currently, only host-based processors that support a manual batch settlement (or batch release) require all payment methods to be settled at the same time ➤ When TransType equals Capture: This element does not apply since only 1 transaction will be settled

Purchase Card Level 3 Data Use

Level 3 data on Purchase Card transactions is now available, but currently only through the Concord EFS processor. This sends invoice information with line-item detail provided to the processor for validation. The data is all sent through the **ExtData** parameter nested within the **<Invoice>** element (see chart above). Note that if you use the **<Invoice>** element it will ignore any data of the same purpose specified elsewhere by way of a parameter or other **ExtData** parameter tag, i.e. it will use one or the other but not both. For example, if you supply the **<TaxAmt>** element within the **<Invoice>** element *and* as a separate element in the **ExtData** parameter, the separate **<TaxAmt>** element outside the **<Invoice>** element will be ignored (duplicate information will not cause a problem). See Example 10 below for a sample transaction sending Purchase Card Level 3 data.

Fuel Purchases: Standard and Fleet Card Use

Credit card processing for fuel purchases with both Standard and Fleet type cards are now available, currently through Concord EFS only. This functionality allows for fuel purchases with standard credit cards (Visa, Mastercard, etc) and with Fleet type cards (Wex, Voyager, and MasterCard Fleet are currently supported). Fuel purchases are differentiated at the gateway from other purchases by the **Fuel** designation placed within the **<Category>** tag in item descriptions (see Examples 11 through 13 below). In effect, a transaction will only be treated as a fuel transaction if at least one of the items within **<Items>** is designated as category **Fuel**.

Both Standard and Fleet cards require item-level purchase information for fuel purchases (for **TransType's Sale** and **Force**), and Fleet cards may additionally require vehicle number, driver number, and/or odometer information on such purchases. If all the required information for a certain purchase is not provided, the transaction will be rejected and an error message generated. Note that Fleet cards in some cases can be used to purchase non-fuel items on a transaction designated as fuel, but item-level information must be present for all items in the transaction, otherwise the transaction may be declined. The main implication for the developer is that additional data must be passed to the gateway in order for fuel purchases to process correctly.

For standard credit card processing on **Sale** and **Force** fuel transactions, item-level purchase information must be provided. It can be passed inside the **<Items>** tag alone or nestled within **<Invoice>** information in **ExtData**. See above chart for details on these and other required XML tags for standard card processing on fuel transactions, and see Example 11 below.

For Fleet card processing on fuel purchases, additional information on the fleet member such as vehicle number, driver number, and/or odometer information may also need to be provided (according to the requirements for the particular Fleet card; for example, Wex typically requires the **<DriverNum>** tag). See the **<Fleet>** tag as described in the table above, and Examples 12 and 13 below. Fleet data must be provided on **Sale**, **Auth**, **Force**, and **Return** transactions. This information will be saved and, if not obtained a second time for a **Force** (PostAuth), will be automatically sent to the processor.

The Fleet data can generally come directly from the card's magnetic data or the POS system can acquire the data by examining information found on the card's magnetic data and then prompting the cardholder for the required data. When transactions are submitted, the card's requirements will be validated by the payment processor and an error will be generated if the proper Fleet data is not submitted. Exceptions: If a MasterCard Fleet card is used and Fleet information is not provided, the transaction will be processed as a standard fuel transaction, rather than generating an error. Also, MasterCard Fleet allows the user to exclude fleet data on manually-entered transactions.

Manual Fleet transactions present a special case. Fleet cards are different in that the account data embossed on a card is often not the same as the track (magnetic) data. This is true for both Wex and Voyager Fleet cards. The result of this is that in a manually-entered (non-swiped) card submission, the card type cannot be identified by the account number data displayed on the card. For a Wex card, the user must identify the card manually, and this information must be passed in **<CardType>** tag in the **ExtData** parameter (see Example 13). However, a Voyager card cannot be processed manually through Concord EFS at this time.

Examples

The following tables contain sample data you can use to test this Web service. The **UserName** and **Password** parameters should be changed when testing the examples yourself.

Example 1: The example data below will process a manually-entered credit card **Sale** transaction through the payment server. It will be processed even if it is a duplicate transaction.

Parameter	Value
UserName	test
Password	123
TransType	Sale
CardNum	5454545454545454
ExpDate	0509
NameOnCard	John Doe
Amount	1.00
ExtData	<Force>T</Force>

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><RespMSG>Approved</RespMSG><Message>APPROVAL</Message><AuthCode>006063</AuthCode><PNRef>2296</PNRef><HostCode>03530
  EVV3K2ZA2F453Q</HostCode><GetAVSResult>N</GetAVSResult><GetAVSResultTXT>No Match</GetAVSResultTXT><GetStreetMatchTXT>No
  Match</GetStreetMatchTXT><GetZipMatchTXT>No Match</GetZipMatchTXT><GetCommercialCard>False</GetCommercialCard><Ext
  Data>CardType=MASTERCARD</ExtData>
```

</Response>

Example 2: The example data below will process a manually-entered credit card **Auth** transaction as a commercial card through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	Auth
CardNum	4055016727870315
ExpDate	0509
NameOnCard	John Doe
Amount	1.00
InvNum	1001
ExtData	<TaxAmt>0.50</TaxAmt><CustCode>102</CustCode>

Result:

<?xml version="1.0" encoding="utf-8" ?>

<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://secure.redfinnet.com/SmartPayments/">

<Result>0</Result><RespMSG>Approved</RespMSG><Message>APPROVAL
VITAL1</Message><AuthCode>VITAL1</AuthCode><PNRef>2275</PNRef><GetCommercialCard>True</GetCommercialCard><ExtData>InvNum=1001,CardType=VISA
 </ExtData>

</Response>

Example 3: The example data below will process a swiped credit card **Return** transaction through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	Return
CardNum	371449635398431
ExpDate	1205
MagData	371449635398431=05121015432112345678

NameOnCard	John Doe
Amount	1.00

Result:

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

<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://
  secure.redfinnet.com/SmartPayments/">

<Result>0</Result><RespMSG>Approved</RespMSG><Message>APPROVAL
  VITAL7</Message><AuthCode>VITAL7</AuthCode><PNRef>2307</PNRef><GetComm
  ercialCard>False</GetCommercialCard><ExtData>CardType=AMEX</ExtData>

</Response>
```

Example 4: The example data below will process a credit card **Void** transaction through the payment server. The **PNRef** parameter should be changed when testing this example yourself. **Note:** pass the card number and ExpDate with null values on voids

Parameter	Value
UserName	test
Password	123
TransType	Void
PNRef	2308

Result:

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

<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://
  secure.redfinnet.com/SmartPayments/">

<Result>0</Result><RespMSG>Approved</RespMSG><AuthCode>VITAL4</AuthCode>
  <PNRef>2309</PNRef>

</Response>
```

Example 5: The example data below will process a credit card **Force** (PostAuth) transaction through the payment server using Training Mode. The **PNRef** parameter should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	Force
NameOnCard	John Doe
Amount	1.00
PNRef	2310
ExtData	<TrainingMode>T</TrainingMode>

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
  <Result>0</Result><RespMSG>Approved</RespMSG><AuthCode>DEMO-2</AuthCode><PNRef>2318</PNRef>
```

```
</Response>
```

Example 6: The example data below will process a manually-entered credit card **Force** (ForceAuth) transaction through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	Force
CardNum	5454545454545454
ExpData	0509
NameOnCard	John Doe
Amount	1.00
ExtData	<AuthCode>123456</AuthCode>

Result:

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

```

<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://
  secure.redfinnet.com/SmartPayments/">

  <Result>0</Result><RespMSG>Approved</RespMSG><Message>APPROVAL</Message>
  <AuthCode>123456</AuthCode><PNRef>2317</PNRef><GetCommercialCard>False
  </GetCommercialCard><ExtData>CardType=MASTERCARD</ExtData>

</Response>
    
```

Example 7: The example data below will process a credit card **Capture** transaction through the payment server to settle a single transaction in the current batch. The **PNRef** parameter should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	Capture
PNRef	2327

Result:

```

<?xml version="1.0" encoding="utf-8" ?>

<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">

  <Result>0</Result><RespMSG>Approved</RespMSG><Message>ACCEPTED</Message>
  <AuthCode>GB00029
  ACCEPTED</AuthCode><ExtData>Net_Count=1,Net_Amount=1,Settle_DT=20
  04-04-13 15:36:26</ExtData>

</Response>
    
```

Example 8: The example data below will process a credit card **CaptureAll** transaction through the payment server to settle all transactions in the current batch.

Parameter	Value
UserName	test
Password	123
TransType	CaptureAll

Result:

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

<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">

  <Result>0</Result><RespMSG>Approved</RespMSG><Message>ACCEPTED</M
  essage><AuthCode>GB00030
  ACCEPTED</AuthCode><ExtData>Net_Count=1,Net_Amount=1,Settle_DT=20
  04-04-13 15:42:45</ExtData>

</Response>
```

Batch Close Response Detail Detail

There is a programmatic way to add batch detail in the responses for the settlement commands. To avail of this feature, there would be an adjustment to the SQL database and would need to add the following key values in the AppSetting_T

8, ExpandExtDataWhen, ALWAYS NEVER EXTERNAL or INTERNAL, Null

Always = Always send the information to all interfaces.
 Never = Never send the information to any interface.
 External = Send the information to PAY and Web Services only.
 Internal (Default) = Send the information to the Virtual Terminal

The integrator or Database Administrator may avail of different methods to add these values but this is a simple SQL script example on how to do this is:

Use PayServerSQLV2k5

```
Insert into AppSetting_T (Application_Key, AppSetting_Key, AppSetting_Value, XmlProfile_TXT)
VALUES ('8', 'ExpandExtDataWhen', 'ALWAYS', NULL)
```

Examples of Responses with Single/Multiple Batch Detail Enabled

The following examples highlight the additions that are available for multi-batch settlements.

1. Single Batch Success

Response (Web service)

```
<Response><Result>0</Result><RespMSG>Approved</RespMSG><Message>ACCEP  
TED</Message><Message1></Message1><Message2></Message2><AuthCode>GB00  
258 ACCEPTED</AuthCode><PNRef></PNRef><HostCode>GB00258 ACCEPTED</  
HostCode><HostURL></HostURL><ReceiptURL></ReceiptURL><GetAVSResult></  
GetAVSResult><GetAVSResultTXT></GetAVSResultTXT><GetStreetMatchTXT></G  
etStreetMatchTXT><GetZipMatchTXT></GetZipMatchTXT><GetCVResult></GetCV  
Result><GetCVResultTXT></GetCVResultTXT><GetGetOrigResult></GetGetOrigRes  
ult><GetCommercialCard>False</GetCommercialCard><WorkingKey></WorkingKey>  
<KeyPointer></KeyPointer><ExtData>CardType=ALL,Net_Count=3,Net_Amount=-3.0  
0,Settle_DT=2006-06-27  
12:41:52,BatchNum=258,Batch=<Summary>Net_Count=3,Net_Amount=-3.00,Settle_D  
T=2006-06-27  
12:41:52,Result=0</Summary><Detail>Net_Count=3,Net_Amount=-3.00,Settle_DT=20  
06-06-27 12:41:52,Result=0,Number=258,AuthCode=GB00258  
ACCEPTED,Message=ACCEPTED</Detail></ExtData></Response>
```

2. Single Batch Failure

Response (Web service)

```
<Response><Result>12</Result><RespMSG>Decline</RespMSG><Message>RB E  
0004 D  
24</Message><Message1></Message1><Message2></Message2><AuthCode></AuthC  
ode><PNRef></PNRef><HostCode></HostCode><HostURL></HostURL><ReceiptUR  
L></ReceiptURL><GetAVSResult></GetAVSResult><GetAVSResultTXT></GetAVS  
ResultTXT><GetStreetMatchTXT></GetStreetMatchTXT><GetZipMatchTXT></GetZi  
pMatchTXT><GetCVResult></GetCVResult><GetCVResultTXT></GetCVResultTXT  
><GetGetOrigResult></GetGetOrigResult><GetCommercialCard>False</GetCommerci  
alCard><WorkingKey></WorkingKey><KeyPointer></KeyPointer><ExtData>CardTyp  
e=ALL,Batch=<Summary>Net_Count=0,Net_Amount=.00,Settle_DT=2006-06-27  
12:46:07,Result=12</Summary><Detail>Net_Count=3,Net_Amount=-3.00,Settle_DT=2  
006-06-27 12:46:07,Result=12,Number=262,Message=RB E 0004 D  
24</Detail></ExtData></Response>
```

3. Mutli Batch Success

Response (Web service)

```
<Response><Result>0</Result><RespMSG>Approved</RespMSG><Message>ACCEP  
TED</Message><Message1></Message1><Message2></Message2><AuthCode>GB00  
261 ACCEPTED</AuthCode><PNRef></PNRef><HostCode>GB00261 ACCEPTED</  
HostCode><HostURL></HostURL><ReceiptURL></ReceiptURL><GetAVSResult></  
GetAVSResult><GetAVSResultTXT></GetAVSResultTXT><GetStreetMatchTXT></G  
etStreetMatchTXT><GetZipMatchTXT></GetZipMatchTXT><GetCVResult></GetCV  
Result><GetCVResultTXT></GetCVResultTXT><GetGetOrigResult></GetGetOrigRes  
ult><GetCommercialCard>False</GetCommercialCard><WorkingKey></WorkingKey>
```

```

<KeyPointer></KeyPointer><ExtData>CardType=ALL,Net_Count=9,Net_Amount=-9.0
0,Settle_DT=2006-06-27
12:43:49,BatchNum=261,Batch=<Summary>Net_Count=9,Net_Amount=-9.00,Settle_D
T=2006-06-27
12:43:49,Result=0</Summary><Detail>Net_Count=4,Net_Amount=-4.00,Settle_DT=20
06-06-27 12:43:49,Result=0,Number=259,AuthCode=GB00259
ACCEPTED,Message=ACCEPTED</Detail><Detail>Net_Count=4,Net_Amount=-4.00,
Settle_DT=2006-06-27 12:43:52,Result=0,Number=260,AuthCode=GB00260
ACCEPTED,Message=ACCEPTED</Detail><Detail>Net_Count=1,Net_Amount=-1.00,
Settle_DT=2006-06-27 12:43:54,Result=0,Number=261,AuthCode=GB00261
ACCEPTED,Message=ACCEPTED</Detail></ExtData></Response>
    
```

4. Multi-Batch Partial Success

Response (Web service)

```

<Response><Result>15</Result><RespMSG>Partial</RespMSG><Message>ACCEPT
ED</Message><Message1></Message1><Message2></Message2><AuthCode>GB0026
2 ACCEPTED</AuthCode><PNRef></PNRef><HostCode>GB00262
ACCEPTED</HostCode><HostURL></HostURL><ReceiptURL></ReceiptURL><Get
AVSResult></GetAVSResult><GetAVSResultTXT></GetAVSResultTXT><GetStreet
MatchTXT></GetStreetMatchTXT><GetZipMatchTXT></GetZipMatchTXT><GetCVR
result></GetCVResult><GetCVResultTXT></GetCVResultTXT><GetGetOrigResult></
GetGetOrigResult><GetCommercialCard>False</GetCommercialCard><WorkingKey>
</WorkingKey><KeyPointer></KeyPointer><ExtData>CardType=ALL,Net_Count=1,N
et_Amount=-1.00,Settle_DT=2006-06-27
12:48:15,BatchNum=262,Batch=<Summary>Net_Count=1,Net_Amount=-1.00,Settle_D
T=2006-06-27
12:48:15,Result=15</Summary><Detail>Net_Count=4,Net_Amount=-4.00,Settle_DT=2
006-06-27 12:48:15,Result=12,Number=262,Message=RB E 0004 D
24</Detail><Detail>Net_Count=4,Net_Amount=-4.00,Settle_DT=2006-06-27
12:48:17,Result=12,Number=262,Message=RB E 0006 D
24</Detail><Detail>Net_Count=1,Net_Amount=-1.00,Settle_DT=2006-06-27
12:48:19,Result=0,Number=262,AuthCode=GB00262
ACCEPTED,Message=ACCEPTED</Detail></ExtData></Response>
    
```

Example 9: The example data below will process a credit card **RepeatSale** transaction as a recurring payment through the payment server based on the **PNRef** number of a previous **Sale** transaction. The **PNRef** parameter should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	RepeatSale

PNRef	2329
--------------	------

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://
  secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><RespMSG>Approved</RespMSG><Message>APPROVAL
  VITAL2</Message><AuthCode>VITAL2</AuthCode><PNRef>2332</PNRef><GetComm
  ercialCard>False</GetCommercialCard>
```

```
</Response>
```

Example 10: The example data below will process a credit card **Sale** transaction through the Payment Server, passing the appropriate **ExtData** to provide Purchase Card Level 3 information to the processor. Note that Concord EFS is the only processor which will actually use such data at this time; if such data were sent via **ExtData** to another processor, the transaction would process, but the Level 3 data would not be utilized.

Parameter	Value
UserName	concord
Password	123
TransType	Sale
CardNum	5233272716340016
ExpDate	0208
MagData	5233272716340016=080212121228
NameOnCard	John Doe
Amount	26.50
ExtData	<pre><Invoice><InvNum>123</InvNum><Date>050421</Date><BillTo><Custom erId>CID101</CustomerId><Name>John Doe</Name><Address><Street>123 Main Street</Street><City>Any City</City><State>WA</State><Zip>98052</Zip><Country>USA</Country ></Address><Email>test@test.com</Email><Phone>132-123-1234</Phone ><Fax>123-123-1235</Fax><CustCode>CCode123</CustCode><PONum>P O123</PONum><TaxExempt>True</TaxExempt></BillTo><Description>One big sale</Description><Items><Item><TotalAmt>1</TotalAmt><UPC>001</UP C><Quantity>1</Quantity><UnitPrice>1</UnitPrice><Category>Fuel</Cate gory><Description>UnLeaded</Description><UnitOfMeasurement>Gallon</Uni tOfMeasurement></Item></Items><DiscountAmt>0</DiscountAmt><Shippin</pre>

	gAmt>1.11</ShippingAmt><DutyAmt>0</DutyAmt><TaxAmt>1.22</TaxAmt><NationalTaxInc>0</NationalTaxInc><TotalAmt>26.50</TotalAmt></Invoice>
--	--

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><RespMSG>OK</RespMSG><Message>APPROVAL</Message>
<AuthCode>990477</AuthCode><PNRef>38473</PNRef><HostCode>10008
693148</HostCode><GetCommercialCard>True</GetCommercialCard><ExtData>
CardType=MASTERCARD</ExtData>
```

```
</Response>
```

Example 11: The example data below will process a swiped credit card (Mastercard) **Sale** transaction as a fuel transaction through the payment server. Note that **ExtData** contains both required and optional tags for this transaction (see table above).

Parameter	Value
UserName	test
Password	123
TransType	Sale
CardNum	5233272716340016
ExpDate	0208
MagData	5233272716340016=080212121228
NameOnCard	John Doe
Amount	26.50
ExtData	<Items><Item><UPC>001</UPC><Quantity>1</Quantity><UnitPrice>1</UnitPrice><Category>Fuel</Category><Description>UnLeaded</Description><UnitOfMeasurement>Gallon</UnitOfMeasurement></Item></Items>

Result:

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

```
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
  <Result>0</Result><RespMSG>OK</RespMSG><Message>APPROVAL</Message
  ><AuthCode>333333</AuthCode><PNRef>39081</PNRef><HostCode>10008
  778024</HostCode><GetCommercialCard>False</GetCommercialCard><ExtData
  >CardType=MASTERCARD</ExtData>
```

```
</Response>
```

Example 12: The example data below will process a swiped Fleet card **Sale** transaction as a fuel transaction through the payment server. Note that **ExtData** contains both required and optional tags for this transaction (see table above).

Parameter	Value
UserName	test
Password	123
TransType	Sale
CardNum	6900460420001234566
ExpDate	0306
MagData	6900460420001234566=06031000563100000
NameOnCard	John Doe
Amount	1.00
ExtData	<Fleet><DriverNum>123</DriverNum><OdometerReading>78964</OdometerReading></Fleet><Items><Item><UPC>001</UPC><Quantity>1</Quantity><UnitPrice>1</UnitPrice><Category>Fuel</Category><Description>UnLeaded</Description><UnitOfMeasurement>Gallon</UnitOfMeasurement></Item></Items>

Result:

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

```
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
  <Result>0</Result><RespMSG>OK</RespMSG><Message>APPROVAL</Message
  ><AuthCode>333333</AuthCode><PNRef>39082</PNRef><HostCode>10008
```

```
778025</HostCode><GetCommercialCard>False</GetCommercialCard><ExtData>
<CardType=WEX</ExtData>
```

```
</Response>
```

Example 13: The example data below will process a manually-entered Fleet card **Sale** transaction as a fuel transaction through the payment server. Note that the **CardType** must be designated on a manually-entered Fleet transaction. Also note that **ExtData** contains both required and optional tags for this transaction (see table above).

Parameter	Value
UserName	test
Password	123
TransType	Sale
CardNum	0420001234566
ExpDate	0306
NameOnCard	John Doe
Amount	50.00
ExtData	<Fleet><VehicleNum>321</VehicleNum><DriverNum>123</DriverNum><OdometerReading>78964</OdometerReading></Fleet><Items><Item><UPC>001</UPC><Quantity>1</Quantity><UnitPrice>1</UnitPrice><Category>Fuel</Category><Description>UnLeaded</Description><UnitOfMeasurement>Gallon</UnitOfMeasurement></Item></Items><CardType>WEX</CardType>

Result:

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

```
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><RespMSG>OK</RespMSG><Message>APPROVAL</Message>
<AuthCode>333333</AuthCode><PNRef>39083</PNRef><HostCode>100008
778047</HostCode><GetCommercialCard>False</GetCommercialCard><ExtData>
<CardType=WEX</ExtData>
```

```
</Response>
```

Example 14. This example illustrates the new command ADJUSTMENT that can modify an original Sale transaction specific to tip adjustment. This is applicable to all restaurant supported processors. This sample starts with an original sale amount with no tip added in the Ext Data field using HTTP POST

Parameter	Value
UserName:	<input type="text" value="vitalr"/>
Password:	<input type="text" value="1234"/>
TransType:	<input type="text" value="Sale"/>
CardNum:	<input type="text" value="5439750001500347"/>
ExpDate:	<input type="text" value="1208"/>
MagData:	<input type="text"/>
NameOnCard:	<input type="text" value="John Smith"/>
Amount:	<input type="text" value="12.00"/>
InvNum:	<input type="text"/>
PNRef:	<input type="text"/>
Zip:	<input type="text" value="33019"/>
Street:	<input type="text" value="123 Main Street"/>
CVNum:	<input type="text"/>
ExtData:	<input type="text" value="998"/>

Response :

```

<?xml version="1.0" encoding="utf-8" ?>
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="SmartPayments">
  <Result>0</Result>
  <RespMSG>Approved</RespMSG>
  <Message>NO MATCH</Message>
  <AuthCode>TAS770</AuthCode>
  <PNRef>27722</PNRef>
  <HostCode>704623500829</HostCode>

```

```
<GetAVSResult>N</GetAVSResult>  
<GetAVSResultTXT>No Match</GetAVSResultTXT>  
<GetStreetMatchTXT>No Match</GetStreetMatchTXT>  
<GetZipMatchTXT>No Match</GetZipMatchTXT>  
<GetCommercialCard>False</GetCommercialCard>  
<ExtData>CardType=MASTERCARD</ExtData>  
  
</Response>
```

To add a tip to this original Sale transaction, the command ADJUSTMENT is used along with the username,login, PNREF of the original sale and tip amount in the extended data field as shown below.

Currently only the tip amount is adjustable. Please observe the screenshot below there is no need to resend all the credit card information data to perform a tip adjustment

Parameter	Value
UserName:	<input type="text" value="vitalr"/>
Password:	<input type="text" value="1234"/>
TransType:	<input type="text" value="Adjustment"/>
CardNum:	<input type="text"/>
ExpDate:	<input type="text"/>
MagData:	<input type="text"/>
NameOnCard:	<input type="text"/>
Amount:	<input type="text"/>
InvNum:	<input type="text"/>
PNRef:	<input type="text" value="27722"/>
Zip:	<input type="text"/>
Street:	<input type="text"/>
CVNum:	<input type="text"/>
ExtData:	<input type="text" value="<TipAmt>3.00</TipAmt>"/>

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

```
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="SmartPayments">
```

```
<Result>0</Result>
```

```
<RespMSG>Approved</RespMSG>
```

```
<Message>APPROVAL</Message>
```

```
<PNRef>27723</PNRef>
```

```
<GetCommercialCard>False</GetCommercialCard>
```

```
</Response>
```

Examples for Verified By VISA and UCAF for Mastercard

VISA Requests

```
<?xml version="1.0" encoding="UTF-8"?><XMLPayRequest version="2"
Timeout="40"><RequestData><Vendor>2</Vendor><Partner>101</Partner><Transacti
ons><Transaction><Sale><PayData><Invoice><BillTo><Address><Street>8320</Stree
t><Zip>85284</Zip></Address></BillTo><ShipTo><Address><Zip>85284</Zip></Ad
dress></ShipTo><TotalAmt>1.00</TotalAmt></Invoice><Tender><Card><CardType>
VISA</CardType><CardNum>4003002345678903</CardNum><ExpDate>0809</ExpD
ate><CVNum>999</CVNum><Amount>1.00</Amount><ExtData>
<Authentication><XID>123456789012345678901234567890</XID><CAV
V>0987654321098765432109876543210987654321</CAVV>
</Authentication></ExtData></Card></Tender></PayData></Sale></Transaction></Tra
nsactions></RequestData><RequestAuth><UserPass><User>vital</User><Password>12
3</Password></UserPass></RequestAuth></XMLPayRequest>
```

Parameter	Value
UserName:	<input type="text" value="vital"/>
Password:	<input type="text" value="1234"/>
TransType:	<input type="text" value="Sale"/>
CardNum:	<input type="text" value="4003002345678903"/>
ExpDate:	<input type="text" value="0809"/>
MagData:	<input type="text"/>
NameOnCard:	<input type="text"/>
Amount:	<input type="text" value="1.00"/>
InvNum:	<input type="text"/>
PNRef:	<input type="text"/>
Zip:	<input type="text"/>
Street:	<input type="text"/>
CVNum:	<input type="text" value="999"/>
ExtData:	<input type="text" value="< Authentication > < XID > 1234567890123456789012345"/>

Visa Response


```
</Address></BillTo><ShipTo><Address><Zip>85284</Zip></Address></ShipTo><TotalAmt>3.00</TotalAmt></Invoice><Tender><Card><CardType>MASTERCARD</CardType><CardNum>5499990123456781</CardNum><ExpDate>0809</ExpDate><CVNum>998</CVNum><Amount>3.00</Amount><ExtData><Authentication><UCAF>09876543210987654321098765432109</UCAF></Authentication></ExtData></Card></Tender></PayData></Sale></Transaction></Transactions></RequestData><RequestAuth><UserPass><User>vital</User><Password>123</Password></UserPass></RequestAuth></XMLPayRequest>
```

Example of CVPresence

Request

```
CC61| ProcessCreditCard| 1| fdcnorth| 123| sale| 4012000033330026| 0408| || 29.95| || 631461234| 12115 LACKLAND| <CVPresence>Illegible</CVPresence>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?><XMLPayRequest version="2" Timeout="40"><RequestData><Vendor>6</Vendor><Partner>101</Partner><Transactions><Transaction><Sale><PayData><Invoice><BillTo><Address><Street>12115 LACKLAND</Street><Zip>631461234</Zip></Address></BillTo><ShipTo><Address><Zip>631461234</Zip></Address></ShipTo><TotalAmt>29.95</TotalAmt></Invoice><Tender><Card><CardType>VISA</CardType><CardNum>4012000033330026</CardNum><ExpDate>0408</ExpDate><Amount>29.95</Amount><ExtData>CVPresence=Illegible</ExtData></Card></Tender></PayData></Sale></Transaction></Transactions></RequestData><RequestAuth><UserPass><User>fdcnorth</User><Password>123</Password></UserPass></RequestAuth></XMLPayRequest>
```

ProcessDebitCard

This Web service operation processes debit card transactions for a merchant. The URL to access this Web service is: <https://secure.redfinnet.com/SmartPayments/transact.aspx?op=ProcessDebitCard>. The text “localhost” in the URL should be replaced with the actual host name or static IP address of the payment server. Descriptions of the parameters are listed below.

Parameter	Value
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
TransType	<p>Required. Type of the debit card transaction. Valid values are:</p> <ul style="list-style-type: none"> • Sale to make a purchase on a debit card • Return to credit the cardholder’s account • Auth to authorize an amount on a debit card. Pertains only to Concord EFS fuel transactions • Force to place Auth transactions into the current batch (PostAuth). Pertains only to Concord EFS fuel transactions • Capture to settle a single transaction in the current batch; only for terminal-based processors • CaptureAll to settle all transactions in the current batch; only for terminal-based processors or host-based processors that support a batch release feature
CardNum	Required except for Capture and CaptureAll . Debit card number to process the transaction
ExpDate	Required except for Capture and CaptureAll . Debit card’s expiration date in MMY format
MagData	<p>Required except for Capture and CaptureAll; required for all swiped card transactions. Data located on the track 2 of the magnetic strip of the card. Once this field is populated, the transaction will be indicated as <i>Card-Present</i> transaction and usually result in a more favorable retail discount rate. This parameter will remove invalid characters. See list of Removed Characters for more details</p> <p>The format of the MagData (or Track 2 data) is CardNum=ExpDate followed by the service code and checksum. For example, 36438999960016=05121015432112345678</p>
NameOnCard	Optional, depending on different merchant processor setup. The cardholder’s name as it appears on the card. This parameter will remove invalid characters. See list of Removed Characters for more details
Amount	Required except for CaptureAll . The total transaction amount in DDDD.CC format. This amount includes CashBackAmt and SureChargeAmt
InvNum	Optional. Invoice tracking number. This parameter will remove invalid

	characters. See list of Removed Characters for more details
PNRef	Optional except for Force and Capture . The reference number assigned by the payment server
Pin	Required except for Capture and CaptureAll transactions and PIN-less debit transactions. The encrypted pin block returned by the pin-pad. The transaction will fail if an unencrypted pin value is used
RegisterNum	Optional. A number uniquely identifies the register or computer on which the transaction is performed. This parameter will remove invalid characters. See list of Removed Characters for more details
SureChargeAmt	Optional. The amount in DDDD.CC format that a merchant charges for processing a debit card transaction
CashBackAmt	Optional. The amount in DDDD.CC format that a cardholder requests for cash back
ExtData	<p>Optional, except for <KeySerialNumber>, which is required for all non-PIN-less Sale, Auth, Force, and Return debit transactions, and <Items> and associated nested data elements (required for Concord EFS fuel purchases- see section below). Extended data in XML format. Valid values are:</p> <ul style="list-style-type: none"> • <TimeOut><i>TimeOut</i></TimeOut > for timeout value in seconds (default = 40) • <TrainingMode><i>TrainingMode</i></TrainingMode> to process transaction in Training Mode; either T or F • <KeySerialNumber><i>KeySerialNumber</i></ KeySerialNumber > for managing DUKPT pin-pads for non-PIN-less debit transactions • <Force><i>Force</i></Force> for forcing duplicate transactions to be processed; either T or F. Note that some processors, including Concord EFS, will not utilize this tag and may still reject a duplicate transaction ❖ <Items> Required for Concord EFS fuel purchases. Items included in invoice. Contains one or more <Item> elements <ul style="list-style-type: none"> ➤ <Item> Required for Concord EFS fuel transactions of type Sale and Force. One item in invoice (item details nested within). There may be multiple <Item> nested within <Items> tag <ul style="list-style-type: none"> ▪ <SKU><i>SKU</i></SKU> SKU number of item ▪ <UPC><i>UPC</i></UPC> Required for Concord EFS fuel purchases. The NACS (National Association of Convenience Stores) product code for fuel purchases. The NACS is an industry standard list that Concord is utilizing. For a list of NACS product codes, please contact EFSnet™ customer support at support@concordefsn.com or 1-877-852-2637. ▪ <Description><i>Description</i></Description> Item description ▪ <Quantity><i>Quantity</i></Quantity Required for Concord EFS fuel purchases. Quantity purchased of item ▪ <UnitOfMeasurement><i>UnitOfMeasure</i></UnitOfMeasurement> Unit of measurement for item ▪ <UnitPrice><i>UnitPrice</i></UnitPrice> Required for Concord EFS fuel

	<p>purchases. Unit price of item</p> <ul style="list-style-type: none">▪ <DiscountAmt><i>DiscountAmount</i></DiscountAmt> Possible discount amount on item▪ <TaxAmt><i>TaxAmt</i></TaxAmt> Possible tax amount on item▪ <TotalAmt><i>TotalAmt</i></TotalAmt> Total amount of item▪ <Category><i>Category</i></Category> Required for Concord EFS fuel purchases. The specific value Fuel to designate fuel purchases▪ <TaxRate><i>TaxRate</i></TaxRate> Possible tax rate applied to item <p>➤ </Item></p> <p>❖ </Items></p>
--	---

PIN-less Debit Transactions

In some cases, debit transactions can actually be processed without the customer's entering a PIN number (a "PIN-less" debit transaction). Essentially, the same information is sent as in a typical PIN-based debit transaction, with the exception of the encrypted PIN-block and key serial number. This transaction type is currently only available with Concord EFS and Global Payments processors.

So, if the processor is not Concord or Global, then **both** the PIN-block and key serial number are required, and without both pieces of data, a transaction will be rejected at the Payment Server. If the designated processor is Concord or Global, then the transaction will be accepted either with **both** pieces of data (interpreted as a PIN-based debit transaction) **or** accepted with **neither** piece of data (interpreted as a PIN-less debit transaction). See Example 3 below.

After the above requirements are met for a transaction, a PIN-less debit transaction will be allowed through the Payment Server. However, it still must have sufficient information to be accepted as a PIN-less transaction only when Concord is the processor. In order for the proper information to be forwarded to Concord for PIN-less debit (and thus for the transaction to be accepted at the processor), the Payment Server must be configured as described below:

Application Id Setup

To process PIN-less debit through Concord, the Application Id sent to Concord must be specified to identify the application in use. Use the following SQL script to change this value in your database:

```
INSERT INTO [dbo].[AppSetting_T] ([Application_Key], [AppSetting_Key],  
[AppSetting_Value], [XmlProfile_TXT]) VALUES (8, 'CustomAppName', 'Your  
Application Name', '')
```

In the above script, you must change '**Your Application Name**' to the Application ID value Concord is expecting, which is typically your company name. Follow these steps in order to execute this script:

- a) Open Query Analyzer
- b) Set the current database to your server database
- c) Paste the above script into the query editor and change '**Your Application Name**' to your company name
- d) Execute the script and verify its success

The CustomAppName is only sent to Concord for PIN-less debit transactions. If CustomAppName is not specified, then the default Application ID will be sent.

Register Number and Terminal Id Setup

When processing transactions with Concord, the Payment Server will detect that the register number passed from the client-side application matches the Register Number field setup in the merchant account. Once it has made the match, then it will send the corresponding Terminal ID field set up for that Register Number to Concord. When no Terminal ID field is sent to Concord, it defaults to what is set up at the processor (usually Terminal ID "01"). If you are also doing VRU (phone-originated) transactions, a separate Terminal ID field will need to be set up in the Registers of the merchant account and submitted in your request through the Web Service. However, if the merchant will be doing both Internet and VRU transactions at the same time, the Terminal ID value will be required to differentiate between the two. For example, you may set up "01" for Internet and "02" for VRU, and the request sent through the ProcessDebitCard operation, from the merchant's PIN-less Debit application, must send the appropriate Register Number to reflect what Terminal ID should be sent.

Fuel Purchases: Debit Card Use

Debit card processing for fuel purchases is now available, currently through Concord EFS only. This functionality allows for fuel purchases with standard debit cards (Visa, Mastercard, etc). Debit fuel purchases (**TransType's Sale and Force**) require item-level purchase information. If all the required information for a certain purchase is not provided, the transaction will be rejected and an error message generated. The main implication for the developer is that additional data must be passed to the gateway in order for fuel purchases to process correctly.

Item-level debit fuel purchase information is passed inside the **<Items>** tag in **ExtData**. Fuel purchases are differentiated at the gateway from other purchases by the **Fuel** designation placed within the **<Category>** tag in item descriptions (see Examples 4 through 6 below). In effect, a transaction will only be treated as a fuel transaction if at least one of the items within **<Items>** is designated as category **Fuel**. See the above chart for details on these and other required XML tags for standard debit card processing on fuel transactions.

Note that PIN information and Key Serial Data must be passed on all debit transactions. This data will *not* be retained after a transaction, so the customer must be present to re-enter the PIN. This is important in the case of a **Force** (PostAuth). See examples 5 and 6 below.

Examples

The following tables contain sample data you can use to test this Web service. The **UserName**, **Password**, **Pin**, and **KeySerialNumber** parameters should be changed when testing the examples yourself.

Example 1: The example data below will process a swiped debit card **Sale** transaction through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	Sale
CardNum	4055011111111111
ExpDate	1205
MagData	4055011111111111=05121015432112345678
NameOnCard	John Doe
Amount	1.00

InvNum	1001
Pin	6366C0466A74C3F6
CashBackAmt	0.5
ExtData	<Timeout>100</Timeout> <KeySerialNumber>4A003102930003BB</KeySerialNumber>

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result> <RespMSG>Approved</RespMSG> <Message>APPROVAL
  VITAL7</Message> <AuthCode>VITAL7</AuthCode> <PNRef>2428</PNRef> <Ext
  Data>InvNum=1001,CardType=DEBIT</ExtData>
```

```
</Response>
```

Example 2: The example data below will process a swiped debit card **Return** transaction through the payment server. The **PNRef** parameter should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	Return
CardNum	4055011111111111
ExpDate	1205
MagData	4055011111111111=05121015432112345678
Amount	1.00
PNRef	2428
Pin	6366C0466A74C3F6
ExtData	<KeySerialNumber>4A003102930003BB</KeySerialNumber>

Result:

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

```

<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
  <Result>0</Result><RespMSG>Approved</RespMSG><Message>APPROVAL
  VITAL9</Message><AuthCode>VITAL9</AuthCode><PNRef>2430</PNRef><Ext
  Data>CardType=DEBIT</ExtData>
</Response>
    
```

Example 3: The example data below will process a swiped PIN-less debit card **Sale** transaction through the payment server. Note that this will currently only work with specific payment processors and will be processed as PIN-less debit when both the PIN-block and key serial number information are purposefully omitted.

Parameter	Value
UserName	test
Password	123
TransType	Sale
CardNum	4011190070070071
ExpDate	0606
MagData	4011190070070071=060600199100
NameOnCard	John Doe
Amount	1.00

Result:

```

<?xml version="1.0" encoding="utf-8" ?>
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
  <Result>0</Result><Message>APPROVAL</Message><AuthCode>216880</AuthCod
  e><PNRef>38472</PNRef><HostCode>100008691797</HostCode><ExtData>C
  ardType=DEBIT</ExtData>
</Response>
    
```

Example 4: The example data below will process a swiped debit card **Sale** transaction as a fuel transaction through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	Sale
CardNum	4011190070070071
ExpDate	0606
MagData	4011190070070071=060600199100
NameOnCard	John Doe
Amount	1.00
Pin	A0C98099B1341075
ExtData	<KeySerialNumber>1234567890000343</KeySerialNumber><Items><Item><UPC>001</UPC><Quantity>1</Quantity><UnitPrice>1</UnitPrice><Category>Fuel</Category><Description>UnLeaded</Description><UnitOfMeasurement>Gallon</UnitOfMeasurement></Item></Items>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
<Result>0</Result><Message>APPROVAL</Message><AuthCode>245028</AuthCode><PNRef>39548</PNRef><HostCode>10008813318</HostCode><ExtData>
  CardType=DEBIT</ExtData>
</Response>
```

Example 5: The example data below will process a swiped debit card **Auth** transaction through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	Auth
CardNum	4011190070070071
ExpDate	0606

MagData	4011190070070071=060600199100
NameOnCard	John Doe
Amount	50.00
Pin	A0C98099B1341075
ExtData	<KeySerialNumber>1234567890000343</KeySerialNumber>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
<Result>0</Result><Message>APPROVAL</Message><AuthCode>245267</AuthCo
de><PNRef>39549</PNRef><HostCode>100008813334</HostCode><ExtData>
CardType=DEBIT</ExtData>
</Response>
```

Example 6: The example data below will process a swiped debit card **Force** transaction (after Example 5 **Auth**) as a fuel transaction through the payment server. The **PNRef** parameter should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	Force
CardNum	4011190070070071
ExpDate	0606
MagData	4011190070070071=060600199100
NameOnCard	John Doe
Amount	50.00
PNRef	39549
Pin	A0C98099B1341075
ExtData	<KeySerialNumber>1234567890000343</KeySerialNumber><Items><Item><UPC>001</UPC><Quantity>1</Quantity><UnitPrice>1</UnitPrice><Category>Fuel</Category><Description>UnLeaded</Description><UnitOfMeasurement>Gallon</UnitOfMeasurement></Item></Items>

Result:

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

```
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
  <Result>0</Result><Message>APPROVAL</Message><AuthCode>245267</AuthCode><PNRef>39554</PNRef><HostCode>100008813347</HostCode><ExtData>CardType=DEBIT</ExtData>
```

```
</Response>
```

ProcessEBTCard

This Web service operation processes EBT card transactions for a merchant:

<https://secure.redfinnet.com/SmartPayments/transact.aspx?op=ProcessEBTCard>.

Parameter	Value
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
TransType	<p>Required. Type of the EBT card transaction. Valid values are:</p> <ul style="list-style-type: none"> • FoodStampSale to make a purchase on an EBT cardholder's food stamp account • FoodStampReturn to credit to an EBT cardholder's food stamp account • CashBenefitSale to make a purchase on an EBT cardholder's cash benefit account • Inquire to check the balance on an EBT card • Capture to settle a single transaction in the current batch; only for terminal-based processors • CaptureAll to settle all transactions in the current batch; only for terminal-based processors or host-based processors that support a batch release feature
CardNum	Required except for Capture and CaptureAll . EBT card number to process the transaction
ExpDate	Required except for Capture and CaptureAll . EBT card's expiration date in MMY format
MagData	<p>Optional. Data located on the track 2 of the magnetic strip of the card. Once this field is populated, the transaction will be indicated as <i>Card-Present</i> transaction and usually result in a favorable retail discount rate. This parameter will remove invalid characters. See list of Removed Characters for more details</p> <p>The format of the MagData (or Track 2 data) is CardNum=ExpDate followed by the service code and checksum. For example, 36438999960016=05121015432112345678</p>
NameOnCard	Optional, depending on different merchant processor setup. The cardholder's name as it appears on the card
Amount	Required except for CaptureAll . The total transaction amount in DDDD.CC format. This amount includes CashBackAmt and SureChargeAmt
InvNum	Optional. Invoice tracking number. This parameter will remove invalid characters. See list of Removed Characters for more details
PNRef	Optional except for FoodStampReturn and Capture . The reference number assigned by the payment server
Pin	Required except for Capture and CaptureAll . The encrypted pin block returned

	by the pin-pad. The transaction will fail if an unencrypted pin value is used
RegisterNum	Optional. A number uniquely identifies a register or computer, on which the transaction is performed. This parameter will remove invalid characters. See list of Removed Characters for more details
SureChargeAmt	Optional. The amount in DDDD.CC format that a merchant charges for processing an EBT card transaction
CashBackAmt	Optional. The amount in DDDD.CC format that a cardholder requests for cash back. If used, only good for TransType of CashBenefitSale
ExtData	<p>Optional except for <KeySerialNumber>, which is required for FoodStampSale, FoodStampReturn, CashBenefitSale, and Inquire with DUKPT pin-pad setup. Extended data in XML format. Valid values are:</p> <ul style="list-style-type: none"> • <TimeOut><i>TimeOut</i></TimeOut> for timeout value in seconds (default = 40) • <TrainingMode><i>TrainingMode</i></TrainingMode> to process transaction in Training Mode; either T or F • <KeySerialNumber><i>KeySerialNumber</i></KeySerialNumber > for managing DUKPT pin-pads for EBT transactions • <Force><i>Force</i></Force> for forcing duplicate transactions to be processed; either T or F. Note that some processors, including Concord EFS, will not utilize this tag and may still reject a duplicate transaction

The following tables contain sample data you can use to test this Web service. The **UserName**, **Password**, **Pin**, and **KeySerialNumber** parameters should be changed when testing the examples yourself.

Examples

Example 1: The example data below will process a swiped EBT card **FoodStampSale** transaction through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	FoodStampSale
CardNum	4055011111111111
ExpDate	1205
MagData	4055011111111111=05121015432112345678
Amount	10.00
Pin	6366C0466A74C3F6
ExtData	<KeySerialNumber> 4A003102930003BB </KeySerialNumber>

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><RespMSG>Approved</RespMSG><Message>FoodStampBa
  lanceAmount:
  6543.21</Message><AuthCode>VITAL1</AuthCode><PNRef>2431</PNRef><Ex
  tData>CardType=EBT</ExtData>
```

```
</Response>
```

Example 2: The example data below will process a manually entered EBT card **FoodStampReturn** transaction through the payment server. The **PNRef** parameter should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	FoodStampReturn
CardNum	4055011111111111
ExpDate	1205
Amount	10.00
PNRef	2459
Pin	6366C0466A74C3F6
ExtData	<KeySerialNumber>4A003102930003BB</KeySerialNumber>

The following is the result from using the Web service with the above sample data.

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><RespMSG>Approved</RespMSG><Message>
  FoodStampBalanceAmount:
  6543.21</Message><AuthCode>VITAL6</AuthCode><PNRef>2460</PNRef><Ex
  tData>CardType=EBT</ExtData>
```

```
</Response>
```

Example 3: The example data below will process a manually entered EBT card **CashBenefitSale** transaction through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	CashBenefitSale
CardNum	4055011111111111
ExpDate	1205
Amount	10.00
InvNum	1002
Pin	6366C0466A74C3F6
CashBackAmt	5
ExtData	<KeySerialNumber>4A003102930003BB</KeySerialNumber>

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
  <Result>0</Result><RespMSG>Approved</RespMSG><Message>CashBenefitB
  alanceAmount:
  1234.56</Message><AuthCode>VITAL8</AuthCode><PNRef>2461</PNRef><Ex
  tData>CardType=EBT</ExtData>
```

```
</Response>
```

ProcessGiftCard

This Web service operation processes gift card transactions for a merchant:

<https://secure.redfinnet.com/SmartPayments/transact.aspx?op=ProcessGiftCard>.

Parameter	Value
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
TransType	<p>Required. Type of the gift card transaction. Valid values are:</p> <ul style="list-style-type: none"> • Redeem to make a purchase on a gift card • Reload to increase the balance on a gift card • Refund to refund money back to a gift card • Activate to activate a gift card • Deactivate to deactivate a gift card • Inquire to check the balance on a gift card • Void to undo an unsettled transaction • Force to place a transaction not processed through the payment server into the current batch (ForceAuth). Currently only available through Paymentech • Capture to settle a single transaction in the current batch; only for terminal-based processors • CaptureAll to settle all transactions in the current batch or host-based processors that support a batch release feature
CardNum	Required. Gift card number used to process the transaction
ExpDate	Required. Gift card's expiration date in MMY format
MagData	<p>Optional except when processing swiped card transactions. Data located on the track 2 of the magnetic strip of the card. Once this field is populated, the transaction will be indicated as <i>Card-Present</i> transaction and usually result in a more favorable retail discount rate. This parameter will remove invalid characters. See list of Removed Characters for more details</p> <p>The format of the MagData (or Track 2 data) is CardNum=ExpDate followed by the service code and checksum. For example, 36438999960016=05121015432112345678</p>
Amount	Required except for TransType with Inquire . The total transaction amount in DDDD.CC format
InvNum	Optional. Invoice tracking number. This parameter will remove invalid characters. See list of Removed Characters for more details
PNRef	Optional except for TransType with Void . Reference number assigned by the payment server of a previous gift card transaction
ExtData	Optional. Extended data in XML format. Valid values are:

	<ul style="list-style-type: none"> • <TrainingMode><i>TrainingMode</i></TrainingMode> for Training Mode in either T or F • <Force><i>Force</i></Force> for forcing duplicate transactions to be processed; either T or F. Note that some processors, including Concord EFS, will not utilize this tag and may still reject a duplicate transaction • <TimeOut><i>TimeOut</i></TimeOut> for timeout value in seconds (default = 40) • <RegisterNum><i>RegisterNum</i></RegisterNum> for register number. The data within this XML tag parameter will remove invalid characters. See list of Removed Characters for more details • <ForceAuth><i>Force</i></ForceAuth> For previously authorized Paymentech transactions of type Redeem, Reload, and Activate to place these transactions into the current batch. (See hierarchy below) Currently only available through Paymentech. See Examples 2 and 3 below • <AuthCode><i>AuthCode</i></AuthCode> Required for Paymentech transactions of type Force (Redeem ForceAuth) See hierarchy below. The authorization code previously obtained for the transaction. Currently only available through Paymentech <p>❖ <ForceAuth></p> <ul style="list-style-type: none"> ➤ <AuthCode><i>AuthCode</i></AuthCode> Required for Paymentech transactions being processed using <ForceAuth> (placed within <ForceAuth> tag). The authorization code previously obtained for the transaction. Currently only available through Paymentech <p>❖ </ForceAuth></p>
--	--

Gift Card ForceAuth Transactions

When processing gift cards through Paymentech, it is possible to do a transaction of type **Force** (a **Redeem** ForceAuth, for example after a phone authorization is obtained for a purchase) in order to place a gift card **Redeem** transaction in the current batch. Such a transaction requires **<AuthCode>** to be passed in ExtData (See above table and see Example 2 below).

It is also possible to place previously authorized transactions of type **Redeem**, **Reload**, and **Activate** in the current batch using the **<ForceAuth>** tag with **<AuthCode>** nested inside (See above table and see Examples 3 and 4 below). Essentially, that means that a **Redeem** can be forced into the batch in two different ways, whereas the **Reload** and **Activate** types must be placed in the batch through the second method. Please note that Gift Card ForceAuth transactions are only required to place a previously authorized **Redeem**, **Reload**, or **Activate** in the current batch.

Examples

The following tables contain sample data to process gift card transactions through the payment server. The **UserName** and **Password** parameters should be changed when testing the example yourself.

Example 1: The example data below will process a **Redeem** transaction on a gift card through the payment server.

Parameter	Value
UserName	test
Password	123
TransType	Redeem
CardNum	6032250001350000156
ExpDate	0509
MagData	6032250001350000156=09051015432112345678
Amount	10.00
InvNum	1001
ExtData	<Force>T</Force>

Result:

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

```
<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><Message>GiftCardBalanceAmount:
  10.00</Message><PNRef>2355</PNRef><HostCode>100004389913</HostCod
  e><ExtData>InvNum=1001,CardType=EGC</ExtData>
```

```
</Response>
```

Example 2: The example data below will process a swiped gift card **Force** transaction through the payment server. The **AuthCode** value should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	Force
CardNum	6035718888880552378

ExpDate	1210
MagData	6035718888880552378=1012000876414
NameOnCard	John Doe
Amount	3.00
ExtData	<AuthCode>104013</AuthCode>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
  <Result>0</Result><Message>APPROVED</Message><AuthCode>104013</AuthCode><PNRef>39782</PNRef><HostCode>0000014</HostCode><ExtData>CardType=EGC</ExtData>
</Response>
```

Example 3: The example data below will place a gift card **Redeem** transaction into the current batch. The AuthCode value should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	Redeem
CardNum	6035718888880552378
ExpDate	1210
NameOnCard	John Doe
Amount	3.00
ExtData	<ForceAuth><AuthCode>105258</AuthCode></ForceAuth>

```
<?xml version="1.0" encoding="utf-8" ?>
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><Message>APPROVED</Message><AuthCode>105259</AuthCode><PNRef>39783</PNRef><HostCode>0000015</HostCode><ExtData>CardType=EGC</ExtData>
```

```
</Response>
```

Example 4: The example data below will place a gift card **Reload** transaction into the current batch. The AuthCode value should be changed when testing this example yourself.

Parameter	Value
UserName	test
Password	123
TransType	Reload
CardNum	6035718888880552378
ExpDate	1210
NameOnCard	John Doe
Amount	20.00
ExtData	<ForceAuth><AuthCode>105260</AuthCode></ForceAuth>

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

```
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/SmartPayments/">
```

```
<Result>0</Result><Message>APPROVED</Message><AuthCode>105261</AuthCode><PNRef>39784</PNRef><HostCode>0000016</HostCode>
<ExtData>CardType=EGC</ExtData>
```

```
</Response>
```

ProcessLoyaltyCard

This Web service operation processes loyalty card transactions for a merchant:

<https://secure.redfinnet.com/SmartPayments/transact.aspx?op=ProcessLoyaltyCard>.

Parameter	Value
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
TransType	<p>Required. Type of the gift card transaction. Valid values are:</p> <ul style="list-style-type: none"> • Redeem to make a purchase on a gift card • Reload to increase the balance on a gift card • Refund to refund money back to a gift card • Activate to activate a gift card • Deactivate to deactivate a gift card • Inquire to check the balance on a gift card • Void to undo an unsettled transaction • Force to place a transaction not processed through the payment server into the current batch (ForceAuth). Currently only available through Paymentech • Capture to settle a single transaction in the current batch; only for terminal-based processors • CaptureAll to settle all transactions in the current batch or host-based processors that support a batch release feature
CardNum	Required. Gift card number used to process the transaction
ExpDate	Required. Gift card's expiration date in MMY format
MagData	<p>Optional except when processing swiped card transactions. Data located on the track 2 of the magnetic strip of the card. Once this field is populated, the transaction will be indicated as <i>Card-Present</i> transaction and usually result in a more favorable retail discount rate. This parameter will remove invalid characters. See list of Removed Characters for more details</p> <p>The format of the MagData (or Track 2 data) is CardNum=ExpDate followed by the service code and checksum. For example, 36438999960016=05121015432112345678</p>
Amount	Required except for TransType with Inquire . The total transaction amount in DDDD.CC format
InvNum	Optional. Invoice tracking number. This parameter will remove invalid characters. See list of Removed Characters for more details
PNRef	Optional except for TransType with Void . Reference number assigned by the payment server of a previous gift card transaction
ExtData	Optional. Extended data in XML format. Valid values are:

- **<TrainingMode>***TrainingMode***</TrainingMode>** for Training Mode in either **T** or **F**
- **<Force>***Force***</Force>** for forcing duplicate transactions to be processed; either **T** or **F**. Note that some processors, including Concord EFS, will not utilize this tag and may still reject a duplicate transaction
- **<TimeOut>***TimeOut***</TimeOut>** for timeout value in seconds (default = 40)
- **<RegisterNum>***RegisterNum***</RegisterNum>** for register number. The data within this XML tag parameter will remove invalid characters. See list of [Removed Characters](#) for more details

Examples

This example shows a redeem transaction for Loyalty cards.

Parameter	Value
UserName:	<input type="text" value="smart"/>
Password:	<input type="text" value="1234"/>
TransType:	<input type="text" value="Redeem"/>
CardNum:	<input type="text" value="6043990501000200"/>
ExpDate:	<input type="text" value="1212"/>
MagData:	<input type="text" value="6043990501000200=12129881"/>
Amount:	<input type="text" value="1"/>
InvNum:	<input type="text"/>
PNRef:	<input type="text"/>
ExtData:	<input type="text"/>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="SmartPayments">
  <Result>0</Result>
  <Message>LoyaltyCardBalanceAmount: 953</Message>
```

```

<AuthCode>A</AuthCode>
<PNRef>27120</PNRef>
<ExtData>CardType=LOYALTY</ExtData>
</Response>
    
```

2. This example shows the Reload function where I am adding 5 points to the loyalty card.

Parameter	Value
UserName:	<input type="text" value="smart"/>
Password:	<input type="text" value="1234"/>
TransType:	<input type="text" value="Reload"/>
CardNum:	<input type="text" value="6043990501000200"/>
ExpDate:	<input type="text" value="1212"/>
MagData:	<input type="text" value="6043990501000200=12129881"/>
Amount:	<input type="text" value="5"/>
InvNum:	<input type="text"/>
PNRef:	<input type="text"/>
ExtData:	<input type="text"/>
<input type="button" value="Invoke"/>	

Result:

```

<?xml version="1.0" encoding="utf-8" ?>

= <Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="SmartPayments">

  <Result>0</Result>
  <Message>LoyaltyCardBalanceAmount: 957</Message>
  <AuthCode>A</AuthCode>
  <PNRef>27122</PNRef>
  <ExtData>CardType=LOYALTY</ExtData>
</Response>
    
```

ProcessSignature

Signature capture can be accomplished by using this Web service:

<https://secure.redfinnet.com/SmartPayments/transact.aspx?op=ProcessSignature>.

Parameter	Description
UserName	Required. User name assigned in the payment server
Password	Required. Password for the user name assigned in the payment server
SignatureType	<p>Required. The type of signature to capture. Valid values are:</p> <ul style="list-style-type: none"> • Signature1 for Lipman credit • Signature2 for Lipman check • Signature3 for handheld application using AppForge • Signature4 for application not using AppForge • Receipt1 for TIFF file
SignatureData	<p>Required. If the SignatureType is set to Signature4, a string value of vector coordinates delimited with a ^ character in the following format: $x1,y1^x2,y2^xN,yN^~$</p> <p>If there is a pen-up event, then you use the coordinate 0,65535 to signal a break in the line</p> <p>^ is the coordinate delimiter, ~ is the ending delimiter, and a comma (,) is the vector delimiter</p> <p>If the SignatureType is set to Receipt1, then you must compress and Base64 encode the image data. See the examples section below for more information</p>
PNRef	Required. The unique payment reference number assigned to the transaction
Result	Optional. An indicator that specifies if the processed transaction was approved
AuthCode	Optional. The authorization code from the authorization of a transaction
ExtData	<p>Optional. Extended data in XML format. Valid values are:</p> <ul style="list-style-type: none"> • <TrainingMode>T</TrainingMode> an indicator that specifies transactions will be processed for local loop back testing • <TrainingMode>F</TrainingMode> an indicator that specifies transactions will not be processed for local loop back testing

The following tables contain sample data you can use for this Web service. The **UserName**, **Password**, and **PNRef** parameters should be changed when testing this

example yourself. If a transaction already has an image, then the Web service will return “Original Transaction Already Has Receipt.”

Examples

Example 1: The example data below will draw a little square on the receipt and associate the receipt with the transaction with reference number 9568.

Parameter	Value
UserName	test
Password	123
PNRef	9568
SignatureType	Signature4
SignatureData	20,20^20,30^30,30^30,20^20,20^~

Result:

```

Testing Merchant
123
New York, NY 12345

Date: 07/26/04 Time: 09:06:48

ForceCapture:

Transaction #:          9568
Card Type:             Diners
AccNum:  XXXXXXXXXXXX0016
Exp Date:              XX/XX
Entry:                 Manual
Amount:                1.00
Response:  001350

I Agree to Pay Above Total
Amount According to Card
Issuer Agreement (Merchant
Agreement if Credit Voucher)

      □

Signature X.....
    
```

Creating a Receipt Image Transaction from a File

It is possible to send a receipt image file through the ProcessSignature web service operation from a client-side application in order to associate it with a transaction. Due to the overall complexity of creating a receipt image with ProcessSignature, here is a general list of steps your client-side application would need to perform in order to send images to the Payment Server.

1. Get image file from hardware device, etc
2. Convert image format from hardware device to the TIFF image format, if it isn't already in that format.
3. It is required that at this point you perform an LZW compression on the image data in the TIFF format to reduce the file size because the Payment Server will only accept image data up to 25KB. Here's some general information about LZW compression for TIFF images:

Compresses and decompresses without information loss, achieving compression ratios up to 5:1. It may be somewhat slower to compress and decompress than the PackBits scheme.

4. At this point, it is required to compress the file itself with Zip compression to reduce the file size. Any PKZip-compatible Zip compressor and decompressor will work. IPWorks (www.ipworks.com) is a third-party provider of software tools and they have a product called "IPWorks! Zip" that can simplify the programmatic compression because it implements a PKZip-compatible Zip compressor and decompressor: the most common compression used on Windows Platforms
5. Base64 encode the image. This ensures that the binary-based information transported can be converted properly into text-based characters to send in the SignatureData parameter of ProcessSignature
6. Input the compressed/base64 encoded image data into the SignatureData parameter of ProcessSignature, and send it to the Payment Server

Example 2: The example data below will save a receipt image to the Payment Server and associate the receipt with the transaction with reference number 3.

Parameter	Value
UserName	test
Password	123
PNRef	3
SignatureType	Receipt1

SignatureData	UESDBBQAAAAIAI2MPzKHziuFjggAAAO+AAA/AAAAUHJvZ3JhbSBGaWxlc9YDb21tb24gRmlsZXMvU21hcnRQYXltZW50cyBtaGFyZWQvVGvVtcFJlY2VpcHQyYm1w7dp/TBvXHQBw82MEOnfx4TIymoIldL9gUSKOkBR4gaiLfk7dNKm/gFS+GGvPJSSkkWCRC65/KAxVRmQ5Y9tKpPJH1VXJov+UduZLHRNQGbTKDBpdRjCOMAVnjSOw0Oc7dyPt3d3vmLfe4dCFxam3gWf7/mj9+6979179+4u9T995phJXqSvF+Hnc/h5OdNkyoD/pKW/1oRdAADSB7/Ey8oMM8wwwzwzBFROJbfKRuha9lp3TUhL/YCE2xji2Vg8kY2n32VCXoM0s5xNo969D55Cdysb1Z5UUHsAbC7PCP2yZCubaTFjwbcAuhhlmmLxQqcZtY0Lar+kmyuuI/Ic3l1p4mgEiueJ1TM5sQY1ULYIapRqHGqMaj1pMWlmlVRZiCwK1o4aBSTPx68yoyY2Iq5WJpRup7tCJGty9yQobzhM448Fn8Hvj60ZuGZM8TjMYiyWPBYkxLnksSLUdKSYkf6a0ZpUbl+8qIh3iFOM/s0glylWM7wfdhLhlciesiAUTUjvkGcDITDAb6y64oyLpJx6QiRSTt0wmq3KSmQjRIG6wEpS6CURLum2AIKU93RkPvptufKqB5037dfvY0p7p04btm3gLDIAyl8xVcTk3Kaw/KrptfhNc8twUU2vzdS+MVR2yko5ok5GkOC27/wkZc7uzBPbdjP3KeS9mmNvS3EV96pH/Zy3ffmKdreOKvajdbeiSrHE/NXB8xHLrYXOOnftNWpZfpgj+bkuZQodACKoqWeqphQLdVfHliAMAD+AP4CT3LxKcYzpfslZ39bJqbkILDs+WS12XQKVPOvlXjKz1oKsZv+35me3KuOEb7O4MizTV1iom/NRMen+MfLkXYusetWlhTjBxpIN715f6iOglsoNp8ovadai5QIkavvWnxZgxdL3cqJIWRgJXhCFGukTzmJfNFYS27gBRRXg7qwp8iZrqmnJrrWFPOUAFjUXhq2viZye/wx6fP8+2DWRM/W3xNnZfDU5PifMwvxaBnVax1MvTPG0OqwVOzgfOF379c7Lkthi4ySykGT83I+gxdcOzvf70lrhQwq+bBktWkA80BGoaAaB6RBLMhX4KUuini5jgQD0nXCjAQ3u2YGbanjEETPlOWI6cCz04TGH08NCWnONJDihZVLIXqYoFOUWhtnk5OZUYR22dUGakYM2K1OWx2vckLb6NqR0dF88rDiU1eRSNJ00qqUCUgJpVvjDWmyxzAbVNdSf3UFvvUoU5B+L50IaN5wdKChdPr1VJ4eIZSP6Ci2dfssydxIO1/+N4Lv/oiJLC9Hd2JjNc9Pd11TD9nc1Olomp7wl1noHp71E1tRf6u2GGGfYtssltjNG15R02m/t+KLaUtFkr2yQr7b8Cgp+OUYhFe2j7aYvgXxtHjZ3OGgwfFfYWN9AyWXIT8DveROuyJG+KNjNrwQBv+yyIwXjY1hbPnEup2vZ3DoU3o8YjOc8YD0+ +xkKMRjPeXjT6bPXo8ZO5z3oYnNbbx2yoAbAvHJfjOZbkp6Iyhcll04bkMUww76IztnGSF2LBqquVliIKnyltIKdsJXcPr7ZezTyU6H0zTG34wz4cMTOAN8YzQYzT70p6BIT5uzvur/baPlw5K0OxdZeftU+ +8VR2bjpvMGw9WZG6TjptQ2xC5t/uvjGj93hIphZt9IzeN/ovzoXPG57Y8IzqtXbyFYhPTGZai3K PpXTej2z4Eb5Hoy1vkX7LpftIwewxs6TtC1bT0zTTKCIH9zXmw4avDVJzudL9JD2Jp188ZeO5 WcBfF22a8PumPyeQi8fqWeLReXke6WF56jzefdMGM09Mdcc53GDuaDhr7FNb6xj7jlooWLVe Y1FzWWHxqv6z1P3yl9waMuE90chjhRAK00loVIXuPF7YE3AFQjvqO17x+JXql64m+skSntf/95 5jbFe2t55nwL2RsozqrFYANqsYre11nfYHDvjGYfnrN4L2jKB0CFNvxgK+JC6gPhPJl0AOF1gv9Z 2o+27YdG+c71diSstOOMild0gwVRibYh+FrCrUawF8n7XxfZYXE3aYnXqIm191YG17drwVIZa2 2I+F81rK8Ia6wFck9eHtc0pgo/8GT9GRj/36xocP7mmIL7MxTLi3wUT2sOwR+OpY8snD1099f k95bn1+SaTnc1YOGrcORNegYM+x0J6vvUxDkTqpi8MRepmtMaOcZSeJTwjumVGA2VgUK5JT 5mtJN12o+27YdHCC9k2Em/wUhSidPJJ1qBnebdK2/XK3K4umpumx23DXrL4tbLT/PCN7IyX0 PtNlo41ix/N0/YOERHYao1VHJ53v4+ztdbT/IEbFfnjqLGMPPiWgEERFNC/IZ0BovWJte+/sR5C39 Z08sv7svm3a5mwqQI1Yd4trtwH9NsUxhYqxbIZwPWPY6ynQeytY2LPR1DbJMAjEgBaBKitE9Jb TRCNY0xnMcwww3Zg8avHKQD45x5EPkHm0AL9urTtAE1CDLXj+tZbDcsE3V90js1oLfk+7ixo RvMp7zf51ng4OqC1b9q+3TQyfrIq29L1wxS8V225c/cS0gbWKz2HFvXUlxpOxtY0rc+KZ6C/8 uYfy1THjgFP9MVJJG6yJui7Z+TD53fpH2GGWaYYU/GyJ3ZeyUvnoCj1qyLEuumWtMtyH7shte KL2C2oM+XboFi2f5oBaDY50+3UG1OAxzBPyaB+PDTbo2J8mvCEQKIQQ+dbk5e/o+M+QQA dVPVTzmeS9sYp2N8hqX3Vf6S2ebC5Oti5h4KpJ9FYMzB3AuCrjsNOot2PTh4ecm8YsHYWSYCB N/WU5rUlywTKbdIe2R83MHFp+4xLeVxm+aczPbtSaQNVyzQxj1zwUpXhpi3LhDu01vKkZtu pxbyeerzSfLnVrbJGIcwQkqvF1H8sGbjJ4ACdfWO869GTPDNPyfUEsBAHQLFAAAAAAgAjYw/Mof OK4WOCaAACj4AAD8AAAAAAAAAAAAAAlaBAAAAAFByb2dyYW0gRmlsZXMvU29tbW9uIEZp bGVzL1NtYXJ0UGF5bWVudHMgU2hhcmVkl1RlBXSZWwNlaXB0LmJtcFBLBQYAAAAAAQABAG 0AAADrCAAAAAA=
----------------------	---

Result:

```

Test Cert Merchant
123 Any Street
Redmond WA , 98052

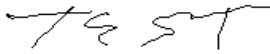
Date: 01/31/05
Time: 17:36:13

Tran Type: Sale
Clerk: merchant1
Invoice:
ACCT: XXXXXXXXXXXXXXX1111
EXP: XX/XX
Issuer: Visa
CardHolder: John Doe
Entry Method: Manual
SYS REF: 3
Sequence Number: 0353YIU798GHE5A8G5

Result: Approved
Response: 015245

Amt: $1.00
Tax: $0.00
Tip: $0.00
Total: $1.00

I Agree to Pay Above Total Amount
According to Card Issuer Agreement
(Merchant Agreement/Credit Voucher)



Signature X.....
    
```

Example 3: The example data below will save a check image to the Payment Server and associate it with the transaction with reference number 5454


Parameter	Value
UserName	test
Password	123
PNRef	5454
SignatureType	Receipt1
SignatureData	UESDBBQAAAAIAAJt9zC2UqIIAxYAAEYAAAzAAAAUHJvamVjdHMvU0RLL1RQSSBTb2Z0d2FyZS9FUFNDbGllbnQvVGVTcFJlY2VpcHQyYm1wdVgHXBPZ1gcMEKoJXQkQEkQsYKTHFKoksEhRUSy4kbWBjSKlOjBLQIiABBBlxQbIYIsXUCwrCyMgC+uiUj6NAhKpCyIiLDVlzdB93v7vvd7JzNzz71z59w7c//nf8+Jj4/CYgUFBXX81FFUUCDgpbw4p6j4t65kojhHYc6sjrcoqnzV5RctBSW8MMDPAUXtWZ0qv6U4d1bHbSpSFU/91Fcqkj+26ZiuKLOVzvymse/xjL8WzdoUpyvoPy1P1vR+P/0dYpWX5/FLSsun7WvimsBirazu6uBymunNVn5ymX5bYMe4ajA4OxnOHkxJCLvb0j3h6w3NbO3sHRyZkRQKX+rTMZfSqq72drYLCBrYP1T0sMtwG77vWHX+eEfrvtr6Yb/g61mxpa7PchonfKI+ygrEcP6hyRX4oQFb+Kii94ZeErGbKIoqaSzknPWH996tWE/mn5zAOURK60i7pMX0jnlHKnlfqz1Iy+4K3bLjgadXhdPkn+0Enz/Kzhc67EyKH+qLohfRCy3TaigVbw50iOB4B88wcTzerpBU6H3QW7CX+etyIHcVhn3yUFO6oGEv+N4ka8osaFEYNJNgbdedubdZmauYwwz0jd2Xz6s1VnfsplZatVfHv0w56xsCnkdb1nqncUda4JVbPcqPLyeE9Amv7gYz0ZIP6PYW7nN0RWd3HQg2+u7UqLeGKvf8juGjrrOvq+nR3MX/giam6FdvT2EyPqRJDp1L98UdlW0UDFvVXk5+CLA2OyeIk4jsIwBSUwKx03L+BfmABBhOayAK2CQvPuG1c68DwewT+ur3R3UlgHRvdatEiqOvTdwOzDM6MSX0pFar3pFF+7Cwu1YfzvrFzoT+qHw+i7aVfQdgcLASEQe0+aEzEQOxCCRIdQAY1c4240rtBT90Qc01SoYXAgTt3CQUs77VhcBrLR4XotuJy+Dd4QqEKzTO50KQJMAA/asAXsOkMMwBjRnshpfb+Tx/U n1ZA2kh3N7OhfCaNhTbdjZRQ3NxC0ajrUQmsG/I+JKQq3NsXS/UXwsKeXkvQZMYv8bOnfr2aOya7eBGawpWIYW2hRaXflw4WaDqhQzbg6TSXtaKQuj1OFmcbCBTjI41Be3p3DGyfes+m87T

Iox0eAgePPYDyTggQpispPyBEj3wRKSniKl2hsF6a9HhiKYL9BsWgV/oFqCWtd8h2MWZXYIKD
zNxxWhpxMeyWhjb5wKPx+qI9yqOycRalw3hIgr09MzEfATk7VAOSO/ltuf+ CCTWIZ9urq3bZ8
ZSL4RChFsaDkSavSQ2DNOfCAzcw0clxw/jrIXTKPjBw7w9rc5oVC3BoJAUvNqTh8MhAm8ybE
0AYdDtOn3qPS9HAfFVImZjIT7PKJ6Xx+TtT8tNZp+Apm800iS6pgp+zsT23HGgr6L+NDWS0
Ji6kCWDtI6G+4cEMvAvXbLRc1674zYZ5CneZ7le0qLzFMYQvQDZda4IsffQILpB+Lzh7oeAKMS
SDL2ZdrYds3qq9F2osOHQyv8kl0ck0I0v6xCuHyYj6yu7QxvsvWWhx5aPjyy1TID5/eYeOo9KVH
fPqId/9vZ868nqlwDfA9SG00ngiD8XcrWgfjooZ/hEEndGgQwSq4LRCIfwKoRAY/s4+jkNq2UfD
tEniwuEQWI4zPTYUuy0XLk07b0d031Zbp/4VePwDv7iATcWGj3Kmh9nN1LEzrWvvtadoq6Wt
Z90xZTzfBkxEFPfBuS1p0MXZRzD2eIYy7DthgePI0EUWwMaeEqzqB8gu/8Bk3gDF4Zvn9h8
XiRInEclWiewKyAN2FwwMjf9GIhKSOj+84C1bteIkToIy4dV8kLvtIWW4ID8h7Dgq4w0/OpOb5
Z+fsG7pB67kH5Bmj6v+IBd4+glu3qxIldrhrozA8rFv9bEHLK1y4E1/Y3X5AbHj6vIWL5qvrp/ZZ
tBY7s9XSVor2Z/mPNezQkQHx6/GtYSVI7+YXDw9WCrn18A7xYCLUHJSTFqsg05vv6K1curAt2
cT7J5HSNcmIoVDJBNfxNeHpGVNLTplG1O/A2NHRTCXIuR7iJfv7oNnVv1x0659Pr9H3tzRfg87
24pOq1eOQ2msdqObROx9fdwBARB1v+jn9dREHYzynZCh7nW/OJ6mFCQKisv6j0uN5xX+JK
w8gI8IPgyH003wUjkZG2kXcpcf9RCAlm159qcDH0MGfGAlP7Ig62Da1n6fy+/hVcfdMxuLVgt7
DzWWIrPzhfWFxu/cfz3+cfHCntTXUwcdhE1cfZ2TTrfVGeIDu362fIGTYqpCcc2nw+S4EOqB9
6VvNlemqdpMueG98/OyTEeiC1iYeg0PmEU/ouUyGAIEPEXMP504rTKC8i6VWrd
pCbTcjNZvTCFV8OG/5ABn6ckVdmNfkYyLvNfbPLjNjCjO7Z42xT8p7CejETHzfQMxkEDZ7KyW
TxrQ4VXQv4l/1qSuOI9K64Fetle5mT8sZ0J5mSaNr4drN8Ad1M8Cxpnc1FIIt96WWS5F2Wv
1c52q7ZlkmvIzZQNK+jJOI56K9ya3c8zd2W6virmcGUWRLyJ6Z6zwsuCMMHzoqWEmpP1npIs
1cyd+7Txe/Y0WnCLVQRlGUngGQQTWuRmk7K9LFPZluFu2R7pq5kd3yy1mPWE9Qs8kguSb
zP/t3NROf8VJM0rPT9H+KjEwMG6EyougX/Ayr2QkZDMtD4be4hgqJcsF68jrZGODSe7ZmBYjX
0YUtiZcEHsF6q3k7Ee20t2OJamOITj/qU8gA7XX25B2RL15C4MFSpAVDhEw+jAURBPru5A/B
VHsc8ibWk2yCyNIO2a6A6d26271FUqPD2ROgdbOT37SHEMKXEDg7oD+5hn7t3npei0b28FX
ACf28xZq3XQOIZtESz9aS46Onz3c9QjivVvmmiHOafg+vMegBVroJdjzuaI7E0hUtiy+V4Mq7K
wUYs04tDeuZiVaC7AWb5hJtNnTW6PFNNXdRygzFsrsOVy0hVjbdjTL/40/+7gS/U0wIT8149n4x
GX7ZaudknWihj40m8RXbbMumQraHOAoKSnKhvGSbVD4tkbi2+YdbIm/WpfJKHC/zTLZqJenh
w8oixXJsA7GFXaHBYq0nExnfPLyGQzIh7kOPgk73POm0twi3xDwK9uNGfjRzw7F52hR2F0K
8RnxtzSFyM5oz7xaicq3UokSnOvVWRzL8+4n+8zavO6Ltpu3fT6nN4Igd2erOo4uwX9RXuhKs
CQZWOEovW+CAdc13OUvcwBhWImRSvftkgHHL9IKil4hu7l2xkB0VTG5Or8VBkNrnm1R1i4e
ocUlwT7psZfS3pDDdF8Z4Kff18GUZeQ7WwFwD7Qb70AWpYwWk03ITrRakee7nCoJwUPv3bXX
PYW60ikxZkqDkZepJssgszzJR9yKmulnTik+d+12JgIN8YxFRZ602MeNUzqvqJdwFjLjoJpQK5w/
+HCEae9uqytVzLiy8oo2N6mxPIYwUUD2/5C2Wwj9DrbeVuyUg3h4HPxt1fGJkOLVpCaAmDb6
uiQDTJmZdwyz7jth4kj+3Xdz1e5QSGHNihA0Pi15mnWmMlgIWKRTThNIPAJBSEmevhwCCZrv
A3pyXXL4Dv9exTrJJA7BiKRMwkX50iZzaF4KfLtoXi8jmmASNwFqBnAcznVTMAIfh3FtqUtP1a1
wzitkGmIdywg+HdReRrEghrCJFEfwbZ973q3vn6qcwn0FW1+MDIdD9hvnNdLiMuSSfwudrOG
KKgV4JGch3GpnD0QWegjnMJe3f1UwvrB4/gdE/L4DsO0pzEw1mJPFm0rOoNxcSJKK+IPtNF7
4/BdH+MRutzogL1BIhDY1vN0F1YHOP9wtBadbk+Bz+wTCHRv9rcm8WIBokgrhECFvtX86Pj
sJpKheIy3K6yE0uV6ZvRKwVgoRIPFD4BTI56WJ9OXkEI82cvFDcmqBYIPxZ6v9h2bXCWASO3
fjBBdkiH4XFpf117ykCKOMtc29HA/bAUGW3NG8pXRBO0a0Xp1kqKak7Hf/N3zrFPsHkHkV2j
EfiHbYmVrb59QAxuU0RdFGb23V7IwZf3mmK2Rt22wYLiMI3Uem8i6XMCyyqCHU8j2hB7kjiABf
ry25bw2YsZp6MiyE8a5/tbmn/73NSc49+xxTX4tx7/oM0sWLDzG62NDU+DGuT0bebIH8C/Ku
dLxGMGDcYeyeqJepORNd6Lsw2gbTBTb6M/21z+brjg/CrJgMxVcAYKi+HcFDgPwDhkmxkqu
GR+xw7O7fvphnpklWNBIZ5w/vGxvQ9bxuoCAQYueYpb3IYccVuAtz3lWATklZ4gfnuqQqy3wE
zGUHHw1tlhgjsCHU1eqjErwji+6KuYqzOOjxOlcaL54PInFaKyMYWpA4arAQzGC3FmFdAdhLbq
hIRxsR48NigZOYqYDuULFidNkSS+PzIvQ/LpHg6kOu+7w6GI mollSWATQqFkuvt3svj23JRM0
dSNo8NlzR4SkN+PTCij8Dk1Za4KdktOIrCeI2m9dD+
+1JGXkzo4C2Y/ttH8fhTgFosDJEi2NZy0AEDaXFBMe9NbpZ0+OIo11IkxvP6UqjeviZP+m7We
Dn51dR0Vq5scUPEouvJCdrPp3vdMuhKvZMWH/BnUE6B8CvtQSVgE/oRoCPJ4xorNjH46wbde
5mMKV6d0ldO+2XVkmB88ULQNpWslHFFFEQR1r4k0YSKt/NbUhm/38y8TzC8X9ploA+Z2et
QeGnq/5E/2r8205wF9+2MnrKSiMthpAInk9iQ4jypeceIhDPjtjujw7RmKTuKjeXD2/wEHF+253r
1+OKLxfrVnDhX5meTrMG9PwLAvIln5VxmKkHMYqvwldMIF9CE2Q2FpFfRH/nZf94VaxT5/PrY
KxxC8cjLrv/eB5hhXrUPCplxmS9JpX6As+YtiZodumpaYwV5s2tCZ8xF8Fw/umKmHhf91nNvP
tzp74eLEthAXtUbpHX933PvU7jsRPQ3Y0GHJaIFXGQ5n6CyJmwxGZCU/wIQo6NNMykku2hF
ohs1chofd4+iQ5creeqF4GnYVM53+zDKFwapLORe+z50QhU3ZL4mNcZ9oFdWjWEWgGFv5Q
doMKMSL25M1TjDf7YgNnPmDOZCOTJUcetwR86ri0GasJVlpcxs4tdzauVt3ww+Ntba/s2Rq+p
umUPwUej33Z2p/LrlanFPPVC0mzav/bRRJHVZz0yJTcp56L9zssYpq9d68TYICnWst6H+BDMs
cbSHwIuyh2e2m5t61kDAZWvo9Zus1vraTvn8nkPIUC713RiLUPQ13M8mfPy2vFjTcZZb00Mvel
JNg1deZw9zRQGhZzNMxGL6ORfXSHoUq2c2hWraccg5uZBVbct0+h0b4x1S6k0Ica5WhtTt8
u+ZnjVWac14j06g3kEw2v7Qdnq79IbLe/AZT2fzpeBZ23p/cpu/Q9aT0VyV3i05KpUijRmjV
7vryqdyP/Tfe7t9kVWw/wm7s5GJSw7sibB9OQsbox8ijanXV6vv1kXwXeTA/6MQ6T0m9x+Yaa4
MJ0Uhd0SOHdyf7vHVfRtcTjn7QfHIEhi6DjPKHsVs2zVRpJykhaTXIORF1EUkzk4e6S1R75L7lsjw

	<p>PsVEI9773j+vsy0A6Tn1PFKrFSNvfwgxcd2FeUZk+1ZfoWtOz/x3cuuDQwgU4+ME3dh7b2t07r y4izcex4lyt0ZcvCtxsMM5EGK0J9AR37hPXncvT1RNIH8ZqMQnfMYLZngDo58IUDYBAMnYArP TOUi4aKVyAj04LAz9zMn0AmAOWA9uL90oxMw17UEb8ffdMAaH/3nq/B7hOd3kZTP37Amtm WOB1v1wIuZX2rp6ltLNwGmt7gt6uqMkV6+tUhnFdKG8G2oq6o/T42FmRwX6yIRadFQkhyC 9lzsitLS6MXxLQPngtX7XbnxzGNouXsKxLB+PuKQFzjesOVZT19AC8MR8f+BHSMAXhohH9nB Hxi5N5btQSfzxpU7p4zIt7u772IBzdFAOT0wdR9zMMQTWC5nVfFsnjh6xoB9vizMjHb/VxQEdn SCuz55R9PLh5jLTGX83MTI9G/xGr2LFhskqIPaiA+z6bVm7YHFxj8Lj4ZnFqZZJm08ZFRH9G2D g1WLu9gj8Yz8k1Ogr5VTFPjyngyUYQqy+/VLOW8VNC1537z/rKzGyIO/SfFhb9FKRJvH3Tk9L 9jR3I1jMnaNtoZfJc0lofOb4IjFjwUMh1eVP5ro6jo4bPVGJZvGu8To7gtj1c0y+KSEpukynKUJL3f +9957ozP1LEi5QHdY7PEefBuvQnHeraCp2EbuyjJUjnfmdPfns1PB9IR5F2W/MDKBYnqQ2ffagJ Ga4ZxD1M2MSaXPvakX4siM7eAmtq3TvPXGij+Wf3htZ/7vr+6Ytvfba7g4vTWNsPn5W93Cmfu vf+44Q8WIVLcVyBRuPUrtMLkV/5/qoP3L+G5dszX6ejgal6ZuXKonV2ivmLW/imOB9meZ/UHQ S9MspZDOChnrIDQty/XojF/40x4RWb9nHdE/Spfc1ZqQVOi9QVvV3KectajZLMhy35r/ni0kPHtI HyCZKi7pOhfZH1tOY5Gry2Grz96l/RhKZy5vWavMIInJpFL5zDtOcSg6kv+pjvT1GuW/Z2CDSO aLsfjDR5z+tOgQxdzj76cvuWuc/KT18WU+giceCXzrzu5D6ypzprqyxNWq5iwnqRktPxnAHzB nD5weVCI7XKT35a51pK1yV6nfJxGbJx5JriB5GSyxtTaxoMW8Hs/KYhwuv1QoCnzBpBhFprD KE/Skhj085UKW1DH3DE/5Usx6KaPUyZXRtnJeLSYcswaIWgoH0hMp/rKkjQ647j5+06oEwU 9dXFiw5UgKAp+dtK9Rmta8IC3LIQr622MCZU/oK9OH0txDVSz4VRTIsa1UIIGQcN0OUUnuVnv BNnr7ffROxOI1WldSjXKm2vJBm04Lc0nkW2Swoce8EpdS96aEdRX6WsTDXmEqs4EJWTQo6E/ 6EqEqEb5c6rpM3wp8ERzTz9pKcmqnrI0J8nwfciULTC5pHJevLwNPhfgXvqSJKGptWWUcy5Pn 662nydvdoWl1sMlxw2ycaIn06uJxtSjnJc9Vb/WOrdC3xXSrgLStaZp9Bm7z94EM+cI8jBbqTMB 9a4vTSiB4nPL2/4y6jAkxpQrb7u1yQ2783OI65B7p541nNT5XKCXi+RUvaC6evfH7n8WTZPve eK0kc3Nxy5PUS5dyV7Gt9dIOW18FSTd9akUHc2ftbRUU30L0tZdJl7byG11icg8fCT7A5vl+JDZ PYhw93hbfS7eC4x30Y1cWv6G1NV5S2HkirOJ9hz5IQX/WLmsbPN0/Yh+UmHwPj5NfrmoLlaEq a5ZgEO1xxXz0XryhoOZGzpY86p7ZrcsDFzP2V8aYRz/277t1o5jF69/QH4C50mRaZQE/mnD3t GpqZ309iUm/zThH1+maVPvqQ5+lgZKJbM/1v/X1BLAQIUCxQAAAAIAAJt9zC2UqllAxYAAEAY AAAzAAAAAAAAAAAAIAAC2gQAAAABQcm9qZWN0cy9TREsvVFBjIFNvZnR3YXJlOVVQU0NsaW VudC9UZw1wUmVjZWlwdC5ibXBQSwUGAAAAAAEAQAQBhAAAAVBAAAAAA</p>
--	---

Result:

Receipt for this transaction:

EPSON AMERICA, INC. 3840 KILROY AIRPORT WAY LONG BEACH, CA 90806		Date _____	15432
Pay to the _____ Order of _____		\$	VOID
EPSON AMERICA, INC. TEST SAMPLE CHECKS		Dollars	 EPSON CORPORATION
For _____		NON-NEGOTIABLE	
⑆ 123456780⑆		1234567890⑆	5432

ValidCard

This Web service operation does the validation check on a credit card. It checks the card length based on the card type, performs a mod 10 checksum, and checks the expiration date. The return value could be: 0 – good, 1001 – no card number, 1002 – no expiration date, 1003 – invalid card type, 1004 – invalid card length, 1005 – invalid mod 10 check, 1006 – invalid expiration date:

<https://secure.redfinnet.com/SmartPayments/validate.asmx?op=ValidCard>.

Parameter	Description
CardNumber	Required. The number of a credit card
ExpDate	Required. The expiration date of a credit card

Example 1: The example data below will result in return 0.

Parameter	Value
CardNumber	5454545454545454
ExpDate	0509

Result:

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

```
<int xmlns="https://secure.redfinnet.com/SmartPayments/">0</int>
```

Example 2: The example data below will result in return 1005.

Parameter	Value
CardNumber	5454545454545453
ExpDate	0509

Result:

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

```
<int xmlns="https://secure.redfinnet.com/SmartPayments/">1005</int>
```

ValidCardLength

This Web service operation checks for the card length based on the card type:

<https://secure.redfinnet.com/SmartPayments/validate.asmx?op=ValidCardLength>.

Parameter	Description
CardNumber	Required. The number of a credit card

Example 1: The example data below will result in return false.

Parameter	Value
CardNumber	54545454545454

Result:

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

```
<boolean
  xmlns="https://secure.redfinnet.com/SmartPayments/">false</boolean>
```

Example 2: The example data below will result in return true.

Parameter	Value
CardNumber	4126196901499

Result:

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

```
<boolean xmlns="https://secure.redfinnet.com/SmartPayments/">true</boolean>
```

ValidExpDate

This Web service operation validates the expiration date of a credit card:

<https://secure.redfinnet.com/SmartPayments/validate.asmx?op=ValidExpDate>.

Parameter	Description
ExpDate	Required. The expiration date of a credit card

Example 1: The example data below will result in return true.

Parameter	Value
ExpDate	0509

Result:

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

```
<boolean
  xmlns="https://secure.redfinnet.com/SmartPayments/">true</boolean>
```

Example 2: The example data below will result in return false.

Parameter	Value
ExpDate	0304

Result:

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

```
<boolean xmlns="https://secure.redfinnet.com/SmartPayments/">>false</boolean>
```

ValidMod10

This Web service operation validates the credit card by performing a mod 10 checksum on the card number:

<https://secure.redfinnet.com/SmartPayments/validate.asmx?op=ValidMod10>.

Parameter	Description
CardNumber	Required. The number of a credit card

Example 1: The example data below will result in return true.

Parameter	Value
CardNumber	545454545454

Result:

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

```
<boolean
  xmlns="https://secure.redfinnet.com/SmartPayments/">true</boolean>
```

Example 2: The example data below will result in return false.

Parameter	Value
CardNumber	545454545454

Result:

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

```
<boolean
  xmlns="https://secure.redfinnet.com/SmartPayments/">>false</boolean>
```

AddMerchant

This Web service operation allows you to add a merchant:

<https://secure.redfinnet.com/admin/ws/admin.aspx?op=AddMerchant>.

Parameter	Value
UserName	Required. User name assigned in the payment server. This user should have administrative rights to perform this web service.
SecureToken	Required. The secure code given by the payment server located at the Preferences menu, under Password
ResellerKey	Optional. The key of the reseller under which the merchant is added
MerchantUsername	Required. User name for the merchant assigned in the payment server
MerchantPassword	Required. Password for the merchant name assigned in the payment server
MerchantID	Optional. ID for the merchant
MerchantID2	Optional. ID for a second merchant
AnnualSales	Optional. Annual sales of the company
BusinessStartDate	Optional. Date the business started
CompanyName	Optional. Name of the company
DoingBusinessAs	Optional. Type of company (ex: retail)
Url	Optional. URL of the company website
FederalTaxID	Optional. Company's federal tax ID
StateTaxID	Optional. Company's state tax ID
SalesTaxID	Optional. Company's sales tax ID
OwnershipType	Optional. The type of company the merchant belongs to (Ex: corporation)
AutoCloseBatch	Required. (True or False) Set the close batch automation
AutoCloseBatchHour	Required with True, Optional with False. Set the close batch time
ForceDuplicate	Required. (True or False) Allows duplicate transaction
RequirePNRef	Required. (True or False)
ContactFirstName	Required. First name of the contact person
ContactLastName	Required. Last name of the contact person
ContactEmail	Required. Email address of the contact person
ContactDayPhone	Required. Day time phone number of the contact person
ContactFax	Optional. Fax number of the contact person

ContactStreet1	Required. Line one of the of the contact person's street address
ContactStreet2	Optional. Line two of the contact person's street address
ContactCity	Required. Contact person's city
ContactSate	Required. Contact person's state
ContactPostalCode	Required. Contact person's zip code
ContactCountryCode	Optional. Contact person's country
TimeZoneOffset	Optional. Hours off time zone
PaymentMethodXml	Required. Extended data in XML format
RegistersXml	Optional. Extended data in XML format
ExtData	<p>Optional. Extended data in XML format</p> <p>Note: By default, sending the email to the new user/merchant is set to false but the system will send out an email to the merchant by adding this tag:<SendEmail>T</SendEmail></p> <p><VirtualTerminalTipAmount>T or F</VirtualTerminalTipAmount> For enabling tip amount to be processed using the virtual terminal.</p> <p><VirtualTerminalTaxAmount>T or F</ VirtualTerminalTaxAmount> for enabling the virtual terminal tax amount to be processed using the virtual terminal.</p> <p><VirtualTerminalConvenienceAmount>T or F </ VirtualTerminalConvenienceAmount> for enabling the convenience amount to be processed using the virtual terminal.</p> <p><AutoSettleMerchantEmail>T or F</AutoSettleMerchantEmail> for enabling the auto settlement merchant email function for the account if auto settle function is enabled on terminal based accounts.</p>

Example 1: EFSNet

This example table shows the information needed to add a merchant using the EFSNet processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	admin
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxcg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex1
MerchantPassword	123
AutoCloseBatch	FALSE
ForceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactState	WA
ContactPostalCode	98034
PaymentMethodXml	<PaymentMethod><ProcessorID>EFSNet</ProcessorID><PaymentTypeID>EGC</PaymentTypeID><HostBased>True</HostBased><XmlProfile><StoreID>richso01</StoreID><StoreKey>A011CD09BB77BD85530F6B2FD3BD8550F62FD377F68798DD1EE0653E6DE5AED2</StoreKey><CurrencyID>USD</CurrencyID><IndustryID>R</IndustryID><DebitSurcharge>0</DebitSurcharge></XmlProfile></PaymentMethod>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
<Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://secure.redfinnet.com/Admin/ws">
  <Code>OK</Code>
  <Error>A new Merchant was successfully created!</Error>
  <Partner />
  <Vendor>307</Vendor>
  <Username>ex1</Username>
```

</Result>

Example 2: FDCN

This example table shows the information needed to add a merchant using the FDCN processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	admin
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxcg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex2
MerchantPassword	123
AutoCloseBatch	FALSE
ForceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactState	WA
ContactPostalCode	98034
PaymentMethodXml	<PaymentMethod><ProcessorID>FDCN</ProcessorID><PaymentTypeID>MASTERCARD</PaymentTypeID><HostBased>FALSE</HostBased><XmlProfile><DatawireID>00003F1FF8B71A94485E</DatawireID><MerchantID>00001210566</MerchantID><TerminalID>00001367275</TerminalID><IndustryID>R</IndustryID></XmlProfile></PaymentMethod>

Result:

<?xml version="1.0" encoding="utf-8" ?>

```

=<Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
xmlns="https://secure.redfinnet.com/Admin/ws">

<Code>OK</Code>

<Error>A new Merchant was successfully created!</Error>

<Partner />

<Vendor>308</Vendor>

<Username>ex2</Username>

</Result>

```

Example 3: FDCNorth

This example table shows the information needed to add a merchant using the FDCNorth processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	Test
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex3
MerchantPassword	123
AutoCloseBatch	FALSE
FourceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactSate	WA

ContactPostalCode	98034
PaymentMethodXml	<pre><PaymentMethod><ProcessorID>FDCNorth</ProcessorID><PaymentTypeID>DISCOVER</PaymentTypeID><HostBased>FALSE</HostBased><XmlProfile><DatawireID>00000B81C6C0BA03F945</DatawireID><MerchantID>000000925990</MerchantID><TerminalID>462862</TerminalID><IndustryID>R</IndustryID><CSPhoneNumber>123-123-1234</CSPhoneNumber><URLEmail>billp@RFNsoft.com</URLEmail><DebitSurcharge>0</DebitSurcharge></XmlProfile></PaymentMethod></pre>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
= <Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://secure.redfinnet.com/Admin/ws">
<Code>OK</Code>
<Error>A new Merchant was successfully created!</Error>
<Partner />
<Vendor>309</Vendor>
<Username>ex3</Username>
</Result>
```

Example 4: FDCSouth

This example table shows the information needed to add a merchant using the FDCSouth processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	Test
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxcg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex4
MerchantPassword	123

AutoCloseBatch	FALSE
FourceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactSate	WA
ContactPostalCode	98034
PaymentMethodXml	<pre><PaymentMethod><ProcessorID>FDCSouth</ProcessorID><PaymentTypeID>AMEX</PaymentTypeID><HostBased>FALSE</HostBased><XmlProfile><DatawireID>0000D900A4BF93DA74CA</DatawireID><MerchantID>67888886496</MerchantID><TerminalID>99</TerminalID><MerchantNumber>67888886496</MerchantNumber><TerminalNumber>99</TerminalNumber><QualCode>25800000</QualCode><TransNumber/><AMEXSENumber>5041079856</AMEXSENumber><JALSENumber>0000000000</JALSENumber><JCBSSENumber>2222222222</JCBSSENumber><DinersSENumber>4444444444</DinersSENumber><CarteBlancheSENumber>5555555555</CarteBlancheSENumber><DiscoverSENumber>1111111111</DiscoverSENumber><IndustryID>E</IndustryID><InputDeviceKey>1</InputDeviceKey><DebitSurcharge>0</DebitSurcharge></XmlProfile></PaymentMethod></pre>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
<Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://secure.redfinnet.com/Admin/ws">
  <Code>OK</Code>
  <Error>A new Merchant was successfully created!</Error>
  <Partner />
  <Vendor>310</Vendor>
  <Username>ex4</Username>
</Result>
```

Example 5: InterceptD

This example table shows the information needed to add a merchant using the InterceptD processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	Test
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxcg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex5
MerchantPassword	123
AutoCloseBatch	FALSE
ForceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactState	WA
ContactPostalCode	98034
PaymentMethodXml	<PaymentMethod><ProcessorID>InterceptD</ProcessorID><PaymentTypeID>DINERS</PaymentTypeID><HostBased>True</HostBased><XmlProfile><CompanyKey>8990</CompanyKey><PIN>1234</PIN><TerminalID>6177</TerminalID><IndustryID>E</IndustryID><CurrencyID>USD</CurrencyID><DebitSurcharge>0</DebitSurcharge></XmlProfile></PaymentMethod>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
= <Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://secure.redfinnet.com/Admin/ws">
<Code>OK</Code>
<Error>A new Merchant was successfully created!</Error>
```

<Partner />

<Vendor>311</Vendor>

<Username>ex5</Username>

</Result>

Example 6: PayTampa

This example table shows the information needed to add a merchant using the PayTampa processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	Test
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex6
MerchantPassword	123
AutoCloseBatch	FALSE
ForceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactState	WA
ContactPostalCode	98034
PaymentMethodXml	<PaymentMethod><ProcessorID>PayTampa</ProcessorID><PaymentTypeID>CARTBLANCH</PaymentTypeID><HostBased>True</HostBased><XmlProfile><MerchantNumber>700000000544</MerchantNumber><TerminalNumber>002</TerminalNumber><ClientNumber>0002</ClientNumber><UserID>RFNtest1</UserID><Password>bpittman1</Password><HostBased_Manual_Settle>FALSE</HostBased_Manual_Settle><Industry>E</Industry><DebitSurcharge>0</DebitSurcharge></XmlProfile></PaymentMethod>

Result:

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

= <Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
xmlns="https://secure.redfinnet.com/Admin/ws">

<Code>OK</Code>

<Error>A new Merchant was successfully created!</Error>

<Partner />

<Vendor>312</Vendor>

<Username>ex6</Username>

</Result>
```

Example 7: RMRS

This example table shows the information needed to add a merchant using the RMRS processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	Test
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex7
MerchantPassword	123
AutoCloseBatch	FALSE
FourceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond

ContactSate	WA
ContactPostalCode	98034
PaymentMethodXml	<PaymentMethod><ProcessorID>RMRS</ProcessorID><PaymentTypeID>ECHECK</PaymentTypeID><HostBased>True</HostBased><XmlProfile><VerificationPortNumber>54081</VerificationPortNumber><VerificationSiteNumber>641</VerificationSiteNumber><VerificationTerminalNumber>232993</VerificationTerminalNumber><VerificationRuleSet>833</VerificationRuleSet><TruncationPortNumber>54080</TruncationPortNumber><TruncationSiteNumber>65301</TruncationSiteNumber><TruncationTerminalNumber>66</TruncationTerminalNumber><TruncationRuleSet>66</TruncationRuleSet></XmlProfile></PaymentMethod>

Result:

```

<?xml version="1.0" encoding="utf-8" ?>

= <Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
xmlns="https://secure.redfinnet.com/Admin/ws">

<Code>OK</Code>

<Error>A new Merchant was successfully created!</Error>

<Partner />

<Vendor>313</Vendor>

<Username>ex7</Username>

</Result>
    
```

Example 8: VITAL

This example table shows the information needed to add a merchant using the VITAL processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	Test
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex8
MerchantPassword	123

AutoCloseBatch	FALSE
ForceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactState	WA
ContactPostalCode	98034
PaymentMethodXml	<pre><PaymentMethod> <ProcessorID>VITAL</ProcessorID> <PaymentTypeID>DEBIT</PaymentTypeID> <HostBased>FALSE</HostBased> <XmlProfile> <SSL2>True</SSL2> <ConnectionMethod>SSL2</ConnectionMethod> <Processor_ID>VITAL</Processor_ID> <MerchantNumber>88800000394</MerchantNumber> <TerminalNumber>1515</TerminalNumber> <AgentBankNumber>000000</AgentBankNumber> <ChainNumber>111111</ChainNumber> <StoreNumber>5999</StoreNumber> <BinNumber>999995</BinNumber> <CustomerServicePhone>123-1231234</CustomerServicePhone> <LocationNumber>00001</LocationNumber> <PostalCode>98052</PostalCode> <CategoryCode>5999</CategoryCode> <VNumber>71004021</VNumber> <IndustryID>R</IndustryID> <TimezoneID>PST</TimezoneID> <CurrencyID>840</CurrencyID> <CountryID>USA</CountryID> <LanguageID>00</LanguageID> <InputDeviceID>2</InputDeviceID> <ABANumber>111111111</ABANumber> <SettlementAgentNumber>V001</SettlementAgentNumber> <SharingGroup>ABC</SharingGroup> <ReimbursementAttribute>Z</ReimbursementAttribute> <FCSID /> <DebitSurcharge>0</DebitSurcharge> </XmlProfile> </PaymentMethod></pre>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
= <Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://secure.redfinnet.com/Admin/ws">
<Code>OK</Code>
<Error>A new Merchant was successfully created!</Error>
<Partner />
```

<Vendor>**314**</Vendor>

<Username>**ex8**</Username>

</Result>

Example 9: GlobalEast

This example table shows the information needed to add a merchant using the GlobalEast processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	Test
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex9
MerchantPassword	123
AutoCloseBatch	FALSE
ForceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactState	WA
ContactPostalCode	98034
PaymentMethodXml	<PaymentMethod><ProcessorID>GlobalEast</ProcessorID><PaymentTypeID>JAL</PaymentTypeID><HostBased>True</HostBased><XMLProfile><HostBased_Manual_Settle>True</HostBased_Manual_Settle><MerchantID>3929300</MerchantID><BankID>095000</BankID><Key>3606</Key><IndustryID>R</IndustryID></XMLProfile></PaymentMethod>

Result:

<?xml version="1.0" encoding="utf-8" ?>

```

=<Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
xmlns="https://secure.redfinnet.com/Admin/ws">

<Code>OK</Code>

<Error>A new Merchant was successfully created!</Error>

<Partner />

<Vendor>315</Vendor>

<Username>ex9</Username>

</Result>
    
```

Example 10: RS

This example table shows the information needed to add a merchant using the RS processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	Test
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex10
MerchantPassword	123
AutoCloseBatch	FALSE
ForceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactState	WA
ContactPostalCode	98034

PaymentMethodXml	<PaymentMethod><ProcessorID>RS</ProcessorID><PaymentTypeID>IMAGE</PaymentTypeID><HostBased>True</HostBased><XMLProfile/></PaymentMethod>
-------------------------	--

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
= <Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://secure.redfinnet.com/Admin/ws">
<Code>OK</Code>
<Error>A new Merchant was successfully created!</Error>
<Partner />
<Vendor>316</Vendor>
<Username>ex10</Username>
</Result>
```

Example 11: Register

This example table shows the information needed to add a merchant with a register using RS processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	Test
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex11
MerchantPassword	123
AutoCloseBatch	FALSE
ForceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com

ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactSate	WA
ContactPostalCode	98034
PaymentMethodXml	<PaymentMethod><ProcessorID>RS</ProcessorID><PaymentTypeID>PAYRECEIPT</PaymentTypeID><HostBased>True</HostBased><XmlProfile /></PaymentMethod>
RegistersXml	<Registers><Register><RegisterName>Terminal1</RegisterName><RegisterNum>01</RegisterNum><DebitTerminalNum>01</DebitTerminalNum><EBTTerminalNum>01</EBTTerminalNum></Register></Registers>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
= <Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://secure.redfinnet.com/Admin/ws">
<Code>OK</Code>
<Error>A new Merchant was successfully created!</Error>
<Partner />
<Vendor>317</Vendor>
<Username>ex11</Username>
</Result>
```

Example 12: Heartland Payment Systems

This example table shows the information needed to add a merchant using the Heartland Payment Systems processor. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	Test

SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxcg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex8
MerchantPassword	123
AutoCloseBatch	FALSE
ForceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5555
ContactStreet1	123 Any St
ContactCity	Redmond
ContactState	WA
ContactPostalCode	98034
PaymentMethodXml	<pre><PaymentMethod><ProcessorID>Heartland</ProcessorID><PaymentTypeID>DEBIT</PaymentTypeID><HostBased>FALSE</HostBased><XmlProfile><SSL2>True</SSL2><ConnectionMethod>SSL2</ConnectionMethod><Processor_ID>Heartland</Processor_ID><MerchantNumber>88800000394</MerchantNumber><TerminalNumber>1515</TerminalNumber><AgentBankNumber>000000</AgentBankNumber><ChainNumber>111111</ChainNumber><StoreNumber>5999</StoreNumber><BinNumber>999995</BinNumber><CustomerServicePhone>123-1231234</CustomerServicePhone><LocationNumber>00001</LocationNumber><PostalCode>98052</PostalCode><CategoryCode>5999</CategoryCode><VNumber>71004021</VNumber><IndustryID>R</IndustryID><TimezoneID>PST</TimezoneID><CurrencyID>840</CurrencyID><CountryID>USA</CountryID><LanguageID>00</LanguageID><InputDeviceID>2</InputDeviceID><ABANumber>11111111</ABANumber><SettlementAgentNumber>V001</SettlementAgentNumber><SharingGroup>ABC</SharingGroup><ReimbursementAttribute>Z</ReimbursementAttribute><FCSID /><DebitSurcharge>0</DebitSurcharge></XmlProfile></PaymentMethod></pre>

Result:

```
<?xml version="1.0" encoding="utf-8" ?>
=<Result xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://secure.redfinnet.com/Admin/ws">
<Code>OK</Code>
```

<Error>**A new Merchant was successfully created!**</Error>

<Partner />

<Vendor>**314**</Vendor>

<Username>**ex12**</Username>

</Result>

Note: When using AddMerchant to programmatically add merchant accounts, there is a requirement to run a sql script that will add the Admin HOST IP. If there were no prior entries, the SQL script to run is the statement below to add the IP of Admin Host. This will create this key in AppSetting_T table entry.

```
INSERT INTO AppSetting_T (Application_Key, AppSetting_Key, AppSetting_Value, XmlProfile_TXT)
VALUES (8, 'AdminHostIP', 'localhost|127.0.0.1', NULL)
```

If the key was already created by running the sql statement above, you can simple update it by running this sql statement:

```
Update AppSetting_T set AppSetting_Value = 'localhost|127.0.0.1|
WHATEVERIPYOUWANTHERE'
```

UpdateMerchant

This Web service operation allows you to update existing merchant information:

<https://secure.redfinnet.com/admin/ws/admin.asmx?op=UpdateMerchant>.

This is a useful web service when it comes to updating business or merchant configuration data.

Parameter	Value
UserName	Required. User name assigned in the payment server. This user should have administrative rights to perform this web service.
SecureToken	Required. The secure code given by the payment server located at the Preferences menu, under Password
ResellerKey	Optional. The key of the reseller under which the merchant is added
MerchantUsername	Required. User name for the merchant assigned in the payment server
MerchantPassword	Required. Password for the merchant name assigned in the payment server
MerchantID	Optional. ID for the merchant
MerchantID2	Optional. ID for a second merchant
AnnualSales	Optional. Annual sales of the company
BusinessStartDate	Optional. Date the business started
CompanyName	Optional. Name of the company
DoingBusinessAs	Optional. Type of company (ex: retail)
Url	Optional. URL of the company website
FederalTaxID	Optional. Company's federal tax ID
StateTaxID	Optional. Company's state tax ID
SalesTaxID	Optional. Company's sales tax ID
OwnershipType	Optional. The type of company the merchant belongs to (Ex: corporation)
AutoCloseBatch	Required. (True or False) Set the close batch automation
AutoCloseBatchHour	Required with True, Optional with False. Set the close batch time
ForceDuplicate	Required. (True or False) Allows duplicate transaction
RequirePNRef	Required. (True or False)
ContactFirstName	Required. First name of the contact person
ContactLastName	Required. Last name of the contact person
ContactEmail	Required. Email address of the contact person

ContactDayPhone	Required. Day time phone number of the contact person
ContactFax	Optional. Fax number of the contact person
ContactStreet1	Required. Line one of the of the contact person's street address
ContactStreet2	Optional. Line two of the contact person's street address
ContactCity	Required. Contact person's city
ContactSate	Required. Contact person's state
ContactPostalCode	Required. Contact person's zip code
ContactCountryCode	Optional. Contact person's country
TimeZoneOffset	Optional. Hours off time zone
PaymentMethodXml	Required. Extended data in XML format, this is the XML string that will contain the configuration settings for the merchant account.
RegistersXml	Optional. Extended data in XML format
ExtData	<p>Optional. Extended data in XML format</p> <p>Note: By default, sending the email to the new user/merchant is set to false but the system will send out an email to the merchant by adding this tag:<SendEmail>T</SendEmail></p> <p><VirtualTerminalTipAmount>T or F</VirtualTerminalTipAmount> For enabling tip amount to be processed using the virtual terminal.</p> <p><VirtualTerminalTaxAmount>T or F</VirtualTerminalTaxAmount> for enabling the virtual terminal tax amount to be processed using the virtual terminal.</p> <p><VirtualTerminalConvenienceAmount>T or F </VirtualTerminalConvenienceAmount> for enabling the convenience amount to be processed using the virtual terminal.</p> <p><AutoSettleMerchantEmail>T or F</AutoSettleMerchantEmail> for enabling the auto settlement merchant email function for the account if auto settle function is enabled on terminal based accounts.</p>

Example

Parameter	Value
UserName	admin
SecureToken	jYizQ51MNs1wf1fcP25eB3iYRxg/u95tGvMRUhyhsIo7HIDMBhq0Lw==
MerchantUserName	ex8
MerchantPassword	123

AutoCloseBatch	FALSE
ForceDuplicate	FALSE
RequirePNRef	FALSE
ContactFirstName	John
ContactLastName	Doe
ContactEmail	JDoe@test.com
ContactDayPhone	555-555-5552
ContactStreet1	123 Any St
ContactCity	Redmond
ContactState	WA
ContactPostalCode	98034
PaymentMethodXml	<pre><PaymentMethods><PaymentMethod><ProcessorID>Heartland</ProcessorID><PaymentTypeID>DEBIT</PaymentTypeID><HostBased>FALSE</HostBased><XmlProfile><SSL2>True</SSL2><ConnectionMethod>SSL2</ConnectionMethod><Processor_ID>Heartland</Processor_ID><MerchantNumber>888000000394</MerchantNumber><TerminalNumber>1515</TerminalNumber><AgentBankNumber>000000</AgentBankNumber><ChainNumber>111111</ChainNumber><StoreNumber>5999</StoreNumber><BinNumber>999995</BinNumber><CustomerServicePhone>123-1231234</CustomerServicePhone><LocationNumber>00001</LocationNumber><PostalCode>98052</PostalCode><CategoryCode>5999</CategoryCode><VNumber>71004021</VNumber><IndustryID>R</IndustryID><TimezoneID>PST</TimezoneID><CurrencyID>840</CurrencyID><CountryID>USA</CountryID><LanguageID>00</LanguageID><InputDeviceID>2</InputDeviceID><ABANumber>111111111</ABANumber><SettlementAgentNumber>V001</SettlementAgentNumber><SharingGroup>ABC</SharingGroup><ReimbursementAttribute>Z</ReimbursementAttribute><FCSID /><DebitSurcharge>0</DebitSurcharge></XmlProfile></PaymentMethod></PaymentMethods></pre>

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

```
= <Result xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="https://secure.redfinnet.com/Admin/ws">
```

```
<code>OK</code>
<error>Merchant was successfully updated!</error>
<Partner>100</Partner>
<Vendor>51</Vendor>
<Username />
</Result>
```

DeleteMerchant

This Web service operation allows you to delete a merchant:

<https://secure.redfinnet.com/admin/ws/admin.asmx?op=DeleteMerchant>.

Parameter	Value
UserName	Required. Reseller name assigned in the payment server (ex: admin)
SecureToken	Required. The Reseller's secure code given by the payment server located at the Preferences menu, under Password
ResellerKey	Required. The key of the reseller under which the merchant is deleted. This would be located at the Preference menu, under Password (ex: MSP/ Partner Number: 100)
MerchantKey	Required. The key of the merchant to be deleted. Located at the Manage Merchants, under Find/Edit and search for the merchant, the ID will be in the far left column.
ExtData	Optional. Extended data in XML format

Example

The following table contains sample data you can use to test this Web service. The parameters should be changed when testing this example yourself.

Parameter	Value
UserName	admin
SecureToken	iEeVzZQ8LeFS8o2HRXN097MZiaur9k1hEoLo1iYnLRPvN81t/xJoTA==
ResellerKey	100
MerchantKey	316

Result:

```

<Result>
<Code>OK</Code>
<Error>Merchant 316 deleted.</Error>
<Partner />100</Partner>
<Vendor>316</Vendor>
<Username/>
</Result>
    
```

AddRecurringCreditCard

This web service operation allows you to add a customer, a contract and a credit card payment method all in one call. All parameters marked as required must be supplied. Optional parameters can be left blank and the default value will be used. Default values are empty strings for *string type* and 0 for integer type:

<https://secure.redfinnet.com/admin/ws/recurring.aspx?op=AddRecurringCreditCard>.

Parameter	Value
Username	Required. The username of the admin user.
Password	Required. The password of the admin user
Vendor	Required. The numerical Vendor/Merchant Key.
CustomerID	Required. A merchant supplied a unique identifier for a customer.
Customer Name	Required. The customer's name is to be submitted in this field.
FirstName	Optional. The customer's first name.
LastName	Optional. The customer's last name.
Title	Optional. The customer's title.
Department	Optional. The customer' department.
Street1	Optional. The customer's street address 1.
Street2	Optional. The customer's street address 2.
Street3	Optional. The customer's street address3.
City	Optional. The customer's city.
StateID	Optional. The customer's 2 character State Code
Province	Optional. The customer's province if it is outside the USA
Zip	Optional. The customer's zip code if in the USA, postal code if outside the USA
CountryID	Optional. The customer's 3 character country code, for example, USA or CAN
Email	Optional. The customer's email address.
Mobile	Optional. The customer's mobile phone.
ContractID	Required. The merchant supplied unique identifier for the contract.
ContractName	Optional. The contract's name.

BillAmt	Required. The amount to be billed in relation to the contract.
TaxAmt	Optional. The tax amount.
TotalAmt	Required. This is the total amount. $BillAmt + TaxAmt = TotalAmt$.
StartDate	Required. The start date of the contract.
EndDate	Optional. The end date of the contract. If this date is not given, the contract will continue to run until manually cancelled or suspended by the system due to failure of payment
BillingPeriod	Required. Specifies the Billing Period Type, used in conjunction with BillingInterval to compute the next bill date.
BillingInterval	Required. Depending on the billing period, it can mean different things such as DAY = every X number of days, WEEK = number of times per week, MONTH = number fo times per month; YEAR = number of times per year.
MaxFailures	Optional. The number of times the system will wait after each retry when a recurring payment fails to process before it puts the contract in suspended mode.
FailureInterval	Optional. Number of days the system will wait after each payment retry when the payment fails.
EmailCustomer	Optional. TRUE/FALSE setting whether to email the customer regarding the status of the recurring payment.
EmailMerchant	Optional. TRUE/FALSE setting whether to email the merchant regarding the status of recurring payment.
EmailCustomerFailure	Optional. TRUE/FALSE setting whether to email the customer when the recurring payment fails.
EmailMerchantFailure	Optional. TRUE/FALSE setting whether to email the merchant when the recurring payemt fails.
CcAccountNum	Required. The customer's credit card number.
CcExpdate	Required. The credit card expiration date.
CcNameOnCard	Optional. The Card Holder's name as it is on the card.
CcStreet	Optional. The Card Holder's billing address

CcZip	Optional. The Card Holder's billing zip code.
ExtData	Optional. Extended Data.

Example

Parameter	Value
Username:	vital
Password:	1234
Vendor:	1
CustomerID:	44444
CustomerName:	Fitness Forever
FirstName:	John
LastName:	Smith
Title:	Sales Director
Department:	Sales
Street1:	123 Main Street
Street2:	
Street3:	
City:	Redmond
StateID:	WA

Email:	JSmith@hotmail.com
DayPhone:	
NightPhone:	
Fax:	
Mobile:	
ContractID:	22222
ContractName:	Fitness 4ever
BillAmt:	19.00
TaxAmt:	1.00
TotalAmt:	20.00
StartDate:	02/13/07
EndDate:	04/13/07
BillingPeriod:	DAY
BillingInterval:	2
MaxFailures:	1
FailureInterval:	1
EmailCustomer:	TRUE
EmailMerchant:	TRUE

CcAccountNum:	5439750001500347
CcExpDate:	1208
CcNameOnCard:	Andy Chau
CcStreet:	
CcZip:	
ExtData:	

Invoke

Response

```
<?xml version="1.0" encoding="utf-8" ?>  
  
= <RecurringResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xmlns="https://secure.redfinnet.com/Admin/ws">  
  
  <CustomerKey>11</CustomerKey>  
  
  <ContractKey>7</ContractKey>  
  
  <CcInfoKey>26607</CcInfoKey>  
  
  <CheckInfoKey />  
  
  <code>OK</code>  
  
  <error>OK</error>  
  
  <Partner>100</Partner>  
  
  <Vendor>1</Vendor>  
  
  <Username>vital</Username>  
  
  </RecurringResult>
```

AddRecurringCheck

This web service allows for adding a customer, a contract and a credit card payment method all in one call. All parameters marked as required must be supplied. Optional parameters can be left blank and the default value will be used. Default values are as follows, empty string for string type and 0 for integer:

<https://secure.redfinnet.com/admin/ws/recurring.aspx?op=AddRecurringCheck>.

Parameter	Value
Username	Required. The username of the admin user.
Password	Required. The password of the admin user
Vendor	Required. The numerical Vendor/Merchant Key.
CustomerID	Required. A merchant supplied a unique identifier for a customer.
Customer Name	Required. The customer's name is to be submitted in this field.
FirstName	Optional. The customer's first name.
LastName	Optional. The customer's last name.
Title	Optional. The customer's title.
Department	Optional. The customer' department.
Street1	Optional. The customer's street address 1.
Street2	Optional. The customer's street address 2.
Street3	Optional. The customer's street address3.
City	Optional. The customer's city.
StateID	Optional. The customer's 2 character State Code
Province	Optional. The customer's province if it is outside the USA
Zip	Optional. The customer's zip code if in the USA, postal code if outside the USA
CountryID	Optional. The customer's 3 character country code, for example, USA or CAN
Email	Optional. The customer's email address.
Mobile	Optional. The customer's mobile phone.
ContractID	Required. The merchant supplied unique identifier for the contract.

ContractName	Optional. The contract's name.
BillAmt	Optional. The amount to be billed in relation to the contract.
TaxAmt	Optional. The tax amount.
TotalAmt	Required. This is the total amount. $BillAmt + TaxAmt = TotalAmt$.
StartDate	Required. The start date of the contract.
EndDate	Optional. The end date of the contract. If this date is not given, the contract will continue to run until manually cancelled or suspended by the system due to failure of payment
BillingPeriod	Required. Specifies the Billing Period Type, used in conjunction with BillingInterval to compute the next bill date.
BillingInterval	Required. Depending on the billing period, it can mean different things such as DAY = every X number of days, WEEK = number of times per week, MONTH = number fo times per month; YEAR = number of times per year.
MaxFailures	Optional. The number of times the system will wait after each retry when a recurring payment fails to process before it puts the contract in suspended mode.
FailureInterval	Optional. Number of days the system will wait after each payment retry when the payment fails.
EmailCustomer	Optional. TRUE/FALSE setting whether to email the customer regarding the status of the recurring payment.
EmailMerchant	Optional. TRUE/FALSE setting whether to email the merchant regarding the status of recurring payment.
EmailCustomerFailure	Optional. TRUE/FALSE setting whether to email the customer when the recurring payment fails.
EmailMerchantFailure	Optional. TRUE/FALSE setting whether to email the merchant when the recurring payemt fails.
CheckType	Required. Two types of checks whether PERSONAL or BUSINESS.
AccountType	Required. Two types of account whether CHECKING or SAVINGS.
CheckNum	Optional. This is the check number.
MICR	Optional. This is the scanned MICR data of the check.

AccountNum	Required. This is the account number.
TransitNum	Required. This is the transit number.
SS	Optional. Social Security number of the check holder.
DOB	Optional. Date of Birth of the check holder.
BranchCity	Optional. The city of the bank where the branch is located.
DL	Optional. The driver's license number of the check holder
StateCode	Optional. The 2 character State Code of the driver's License of the check holder.
NameOnCheck	Optional. The check holder's name as it is on the check.
ExtData	R Optional. Extended Data.

Example

Parameter	Value
Username:	<input type="text" value="ncn1"/>
Password:	<input type="text" value="1234"/>
Vendor:	<input type="text" value="41"/>
CustomerID:	<input type="text" value="2929292"/>
CustomerName:	<input type="text" value="Fitness Land"/>
FirstName:	<input type="text" value="John"/>
LastName:	<input type="text" value="Jones"/>
Title:	<input type="text" value="Sales Manager"/>
Department:	<input type="text" value="Sales"/>
Street1:	<input type="text" value="123 Main Street"/>
Street2:	<input type="text"/>
Street3:	<input type="text"/>
City:	<input type="text" value="Atlanta"/>
StateID:	<input type="text" value="GA"/>
Province:	<input type="text"/>

PostalCode:	41344
CountryID:	USA
Email:	john@hotmail.com
DayPhone:	912-333-7777
NightPhone:	
Fax:	
Mobile:	912-888-7799
ContractID:	898989
ContractName:	Fit4now
BillAmt:	25.00
TaxAmt:	0.00
TotalAmt:	25.00
StartDate:	04/20/07
EndDate:	04/20/08
BillingPeriod:	MONTH

BillingInterval:	<input type="text" value="1"/>
MaxFailures:	<input type="text" value="2"/>
FailureInterval:	<input type="text" value="2"/>
EmailCustomer:	<input type="text" value="TRUE"/>
EmailMerchant:	<input type="text" value="TRUE"/>
EmailCustomerFailure:	<input type="text" value="TRUE"/>
EmailMerchantFailure:	<input type="text" value="TRUE"/>
CheckType:	<input type="text" value="BUSINESS"/>
AccountType:	<input type="text" value="CHECKING"/>
CheckNum:	<input type="text" value="1001"/>
MICR:	<input type="text" value="90000018 100124413815"/>
AccountNum:	<input type="text" value="24413815"/>
TransitNum:	<input type="text" value="90000018"/>
SS:	<input type="text"/>
DOB:	<input type="text"/>
BranchCity:	<input type="text"/>
DL:	<input type="text"/>
StateCode:	<input type="text"/>
NameOnCheck:	<input type="text" value="John Jones"/>
ExtData:	<input type="text"/>

Response

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

```
= <RecurringResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/Admin/ws">

  <CustomerKey>12</CustomerKey>

  <ContractKey>8</ContractKey>

  <CcInfoKey />

  <CheckInfoKey>3</CheckInfoKey>

  <code>OK</code>

  <error>OK</error>

  <Partner>100</Partner>

  <Vendor>41</Vendor>

  <Username>ncn1</Username>

  </RecurringResult>
```

ProcessCreditCard - Recurring Billing

This web service operation processes credit card transactions within the recurring billing module:

<https://secure.redfinnet.com/admin/ws/recurring.asmx?op=ProcessCreditCard>.

Parameter	Value
Username	Required. The username of the admin user.
Password	Required. The password of the admin user
Vendor	Required. The numerical Vendor/Merchant Key.
CcInfoKey	Required. The numerical Credit Card Info key.
Amount	Required. The amount that will be processed for that transaction.
InvNum	Optional. The associated invoice number.
ExtData	Optional. Extended Data.

Example

Parameter	Value
Username:	<input type="text" value="vital"/>
Password:	<input type="text" value="1234"/>
Vendor:	<input type="text" value="1"/>
CcInfoKey:	<input type="text" value="26607"/>
Amount:	<input type="text" value="1.00"/>
InvNum:	<input type="text" value="12345"/>
ExtData:	<input type="text"/>

Response:

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

```
= <RecurringResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xmlns="https://secure.redfinnet.com/Admin/ws">
```

```
<code>OK</code>
```

```
<error>APPROVED</error>
```

```
<Result>0</Result>
```

```
<AuthCode>TAS351</AuthCode>
```

```
<PNRef>26873</PNRef>
```

```
<Message>NO MATCH</Message>
```

```
</RecurringResult>
```

ProcessCheck – Recurring Billing

This web service operation allows for the processing of check transactions within the recurring billing module:

<https://secure.redfinnet.com/admin/ws/recurring.asmx?op=ProcessCheck>.

Parameter	Value
Username	Required. The username of the admin user.
Password	Required. The password of the admin user
Vendor	Required. The numerical Vendor/Merchant Key.
CheckInfoKey	Required. The numerical Check Payment Info key.
Amount	Required. The amount that will be processed for that transaction.
InvNum	Optional. The associated invoice number.
ExtData	Optional. Extended Data.

Example

Parameter	Value
Username:	<input type="text" value="ncn1"/>
Password:	<input type="text" value="1234"/>
Vendor:	<input type="text" value="41"/>
CheckInfoKey:	<input type="text" value="7"/>
Amount:	<input type="text" value="5.00"/>
InvNum:	<input type="text" value="12345"/>
ExtData:	<input type="text"/>

Response

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

```
= <RecurringResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/Admin/ws">

  <code>OK</code>

  <error>APPROVED</error>

  <Result>0</Result>

  <AuthCode>AUTH NUM 674-564</AuthCode>

  <PNRef>26945</PNRef>

  <Message>APPROVAL</Message>

  </RecurringResult>
```

ManageCheckInfo

This Web Service operation allows for managing check information:

<https://secure.redfinnet.com/admin/ws/recurring.aspx?op=ManageCheckInfo>.

Parameter	Value
Username	Required. The username of the admin user.
Password	Required. The password of the admin user
Vendor	Required. The numerical Vendor/Merchant Key.
CustomerKey	Required. The numerical customer key.
CheckInfoKey	Required for TransType UPDATE and DELETE. The numerical Customer Key
CheckType	Required. Two types of checks whether PERSONAL or BUSINESS.
AccountType	Required. Two types of account whether CHECKING or SAVINGS.
CheckNum	Optional. This is the check number.
MICR	Optional. This is the scanned MICR data of the check.
AccountNum	Required. This is the account number.
TransitNum	Required. This is the transit number.
SS	Optional. Social Security number of the check holder.
DOB	Optional. Date of Birth of the check holder.
BranchCity	Optional. The city of the bank where the branch is located.
DL	Optional. The driver's license number of the check holder
StateCode	Optional. The 2 character State Code of the driver's license of the check holder, e.g. NY or GA
NameOnCheck	Optional. The check holder's name as it is on the check.
Email	Optional. The customer's email address.
DayPhone	Optional. The customer's day phone.
Street1	Optional. The customer's street address 1.
Street2	Optional. The customer's street address 2.
Street3	Optional. The customer's street address3.
City	Optional. The customer's city.

StateID	Optional. The customer's 2 character State Code
Province	Optional. The customer's province if it is outside the USA
PostalCode	Optional. The customer's zip code if in USA , postal code if outside USA
CountryID	Optional. The customer's 3 character country code, for example, USA or CAN
ExtData	R Optional. Extended Data.

Example

Parameter	Value
Username:	ncn1
Password:	1234
TransType:	ADD
Vendor:	41
CustomerKey:	12
CheckInfoKey:	3
CheckType:	PERSONAL
AccountType:	CHECKING
CheckNum:	1001
MICR:	490000018100124413815
AccountNum:	24413815
TransitNum:	490000018
RawMICR:	
SS:	

DOB:	<input type="text"/>
BranchCity:	<input type="text"/>
DL:	<input type="text"/>
StateCode:	<input type="text"/>
NameOnCheck:	<input type="text"/>
Email:	<input type="text"/>
DayPhone:	<input type="text"/>
Street1:	<input type="text" value="123 Main Street"/>
Street2:	<input type="text"/>
Street3:	<input type="text"/>
City:	<input type="text" value="Atlanta"/>
StateID:	<input type="text" value="GA"/>
Province:	<input type="text"/>
PostalCode:	<input type="text" value="40190"/>
CountryID:	<input type="text" value="USA"/>
ExtData:	<input type="text"/>
<input type="button" value="Invoke"/>	

Response

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

```
= <RecurringResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/Admin/ws">
```

```
<CustomerKey>12</CustomerKey>
```

```
<ContractKey />
```

```
<CcInfoKey />
```

<CheckInfoKey>**4**</CheckInfoKey>

<code>**OK**</code>

<error>**OK**</error>

<Partner>**100**</Partner>

<Vendor>**41**</Vendor>

<Username>**ncn1**</Username>

</RecurringResult>

ManageCreditCardInfo

This Web Service allows for managing the credit card information:

<https://secure.redfinnet.com/admin/ws/recurring.aspx?op=ManageCreditCardInfo>.

Parameter	Value
Username	Required. The username of the admin user.
Password	Required. The password of the admin user
TransType	Required. The type of transaction being performed. Valid values are: ADD UPDATE DELETE
Vendor	Required. The numerical Vendor/Merchant Key.
CustomerKey	Required. The numerical Customer Key.
CardInfoKey	Required. The numerical credit card info key.
CcAccountNum	Required. The credit card account number.
CcExpDate	Required. The credit card expiration date.
CcNameonCard	Optional. The name of the card holder
CcStreet	Optional. The card holder's billing address.
CcZip	Optional. The card holder's billing zip code.
ExtData	Optional. Extended Data.

Example

Parameter	Value
Username:	<input type="text" value="vital"/>
Password:	<input type="text" value="1234"/>
TransType:	<input type="text" value="ADD"/>
Vendor:	<input type="text" value="1"/>
CustomerKey:	<input type="text" value="11"/>
CardInfoKey:	<input type="text" value="26607"/>
CcAccountNum:	<input type="text" value="5439750001500347"/>
CcExpDate:	<input type="text" value="1209"/>
CcNameOnCard:	<input type="text" value="John Smith"/>
CcStreet:	<input type="text" value="123 Main Street"/>
CcZip:	<input type="text" value="40490"/>
ExtData:	<input type="text"/>

Response

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

```
= <RecurringResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/Admin/ws">
```

```
  <CustomerKey>11</CustomerKey>
  <ContractKey />
  <CcInfoKey>26674</CcInfoKey>
  <CheckInfoKey />
  <code>OK</code>
  <error>OK</error>
  <Partner>100</Partner>
  <Vendor>1</Vendor>
  <Username>vital</Username>
</RecurringResult>
```

ManageContract

This web service allows for managing different properties of contracts via integration:

<https://secure.redfinnet.com/admin/ws/recurring.asmx?op=ManageContract>.

Parameter	Value
Username	Required. The username of the admin user.
Password	Required. The password of the admin user
TransType	Required. The type of transaction being performed. The valid values are: ADD UPDATE DELETE
Vendor	Required. The numerical Vendor/Merchant Key.
CustomerKey	Required. The numerical customer key.
ContractKey	Required for TransType UPDATE and DELETE. The numerical contract key.
PaymentInfoKey	Required for Transtype UPDATE and ADD. The numerical information Key. This is dependent for the PaymentType. If you set the Payment Type to CC then the information that needs to be passed in this field is the CCInfoKey or CardInfoKey. Now if CK was set at the PaymentType, then the information that needs to go in this field is the CheckInfoKey. Please make sure that you are passing the right key based on the PaymentType.
PaymentType	Required for TransType ADD and UPDATE. Type of payment: CC for Credit Card and CK for Check
CustomerID	Required. A merchant supplied a unique identifier for a customer.
Customer Name	Required. The customer's name is to be submitted in this field.
FirstName	Optional. The customer's first name.
LastName	Optional. The customer's last name.
Title	Optional. The customer's title.
Department	Optional. The customer' department.
Street1	Optional. The customer's street address 1.
Street2	Optional. The customer's street address 2.
Street3	Optional. The customer's street address3.
City	Optional. The customer's city.
StateID	Optional. The customer's 2 character State Code

Province	Optional. The customer's province if it is outside the USA
Zip	Optional. The customer's zip code if in the USA, postal code if outside the USA
CountryID	Optional. The customer's 3 character country code, for example, USA or CAN
DayPhone	Optional. The customer's day phone.
NightPhone	Optional. The customer's evening phone.
Fax	Optional. The customer's fax number.
Email	Optional. The customer's email address.
Mobile	Optional. The customer's mobile phone.
ContractID	Required. The merchant supplied unique identifier for the contract.
ContractName	Optional. The contract's name.
BillAmt	Optional. The amount to be billed in relation to the contract.
TaxAmt	Optional. The tax amount.
TotalAmt	Required. This is the total amount. $BillAmt + TaxAmt = TotalAmt$.
StartDate	Required. The start date of the contract.
EndDate	Optional. The end date of the contract. If this date is not given, the contract will continue to run until manually cancelled or suspended by the system due to failure of payment
NextBillDt	Required. This is required for TRANSTYPE ADD and UPDATE.
BillingPeriod	Required. Specifies the Billing Period Type, used in conjunction with BillingInterval to compute the next bill date.
BillingInterval	Required. Depending on the billing period, it can mean different things such as DAY = every X number of days, WEEK = number of times per week, MONTH = number of times per month; YEAR = number of times per year.
MaxFailures	Optional. The number of times the system will wait after each retry when a recurring payment fails to process before it puts the contract in suspended mode.

FailureInterval	Optional. Number of days the system will wait after each payment retry when the payment fails.
EmailCustomer	Optional. TRUE/FALSE setting whether to email the customer regarding the status of the recurring payment.
EmailMerchant	Optional. TRUE/FALSE setting whether to email the merchant regarding the status of recurring payment.
EmailCustomerFailure	Optional. TRUE/FALSE setting whether to email the customer when the recurring payment fails.
EmailMerchantFailure	Optional. TRUE/FALSE setting whether to email the merchant when the recurring payment fails.
Status	Optional. Status of the contract.
ExtData	R Optional. Extended Data.

Example

Parameter	Value
Username:	<input type="text" value="vital"/>
Password:	<input type="text" value="1234"/>
TransType:	<input type="text" value="UPDATE"/>
Vendor:	<input type="text" value="1"/>
CustomerKey:	<input type="text" value="11"/>
ContractKey:	<input type="text" value="7"/>
PaymentInfoKey:	<input type="text" value="26607"/>

PaymentType:	<input type="text" value="CC"/>
CustomerID:	<input type="text" value="44444"/>
CustomerName:	<input type="text" value="Fitness Forever"/>
FirstName:	<input type="text" value="John"/>
LastName:	<input type="text" value="Smith"/>
Title:	<input type="text" value="Sales Manager"/>
Department:	<input type="text" value="Sales"/>
Street1:	<input type="text" value="123 Main Street"/>
Street2:	<input type="text"/>

Street3:	<input type="text"/>
City:	<input type="text" value="Atlanta"/>
StateID:	<input type="text" value="GA"/>
Province:	<input type="text"/>
Zip:	<input type="text" value="40490"/>
CountryID:	<input type="text" value="USA"/>
Email:	<input type="text" value="johns@hotmail.com"/>
DayPhone:	<input type="text" value="912-333-4444"/>
NightPhone:	<input type="text" value="912-888-9999"/>
Fax:	<input type="text"/>
Mobile:	<input type="text" value="912-777-9999"/>
ContractID:	<input type="text" value="22222"/>
ContractName:	<input type="text" value="Fitness4ever"/>
BillAmt:	<input type="text" value="19.00"/>
TaxAmt:	<input type="text" value="2.00"/>
TotalAmt:	<input type="text" value="21.00"/>
StartDate:	<input type="text" value="02/13/2007"/>
EndDate:	<input type="text" value="04/13/2007"/>

NextBillDt:	<input type="text" value="2/13/2007"/>
BillingPeriod:	<input type="text" value="DAY"/>
BillingInterval:	<input type="text" value="2"/>
MaxFailures:	<input type="text" value="1"/>
FailureInterval:	<input type="text" value="1"/>
EmailCustomer:	<input type="text" value="TRUE"/>
EmailMerchant:	<input type="text" value="TRUE"/>
EmailCustomerFailure:	<input type="text" value="TRUE"/>
EmailMerchantFailure:	<input type="text" value="TRUE"/>
Status:	<input type="text"/>
ExtData:	<input type="text"/>

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

```
= <RecurringResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/Admin/ws">
```

```
  <CustomerKey>11</CustomerKey>
  <ContractKey>7</ContractKey>
  <CcInfoKey>26607</CcInfoKey>
  <CheckInfoKey />
  <code>OK</code>
  <error>OK</error>
  <Partner>100</Partner>
  <Vendor>1</Vendor>
  <Username>vital</Username>
</RecurringResult>
```

ManageCustomer

This webservice allows for the management of customer information:

<https://secure.redfinnet.com/admin/ws/recurring.aspx?op=ManageCustomer>.

Parameter	Value
Username	Required. The username of the admin user.
Password	Required. The password of the admin user
TransType	Required. The type of transaction being performed. The valid values are: ADD UPDATE DELETE
Vendor	Required. The numerical Vendor/Merchant Key.
CustomerKey	Required. The numerical customer key.
CustomerID	Required. A merchant supplied a unique indentifier for a customer.
Customer Name	Required. The customer's name is to be submitted in this field.
FirstName	Optional. The customer's first name.
LastName	Optional. The customer's last name.
Title	Optional. The customer's title.
Department	Optional. The customer' department.
Street1	Optional. The customer's street address 1.
Street2	Optional. The customer's street address 2.
Street3	Optional. The customer's street address3.
City	Optional. The customer's city.
StateID	Optional. The customer's 2 character State Code
Province	Optional. The customer's province if it is outside the USA
Zip	Optional. The customer's zip code if in the USA, postal code if outside the USA
CountryID	Optional. The customer's 3 character country code, for example, USA or CAN
DayPhone	Optional. The customer's day phone.

NightPhone	Optional. The customer's evening phone.
Fax	Optional. The customer's fax number.
Email	Optional. The customer's email address.
Mobile	Optional. The customer's mobile phone.
Status	Optional. Status of the contract.
ExtData	R Optional. Extended Data.

Example

Parameter	Value
Username:	vital
Password:	1234
TransType:	UPDATE
Vendor:	1
CustomerKey:	11
CustomerID:	44444
CustomerName:	Fitness Forever
FirstName:	John
LastName:	Smith
Title:	Sales Manager
Department:	Sales
Street1:	123 Main Street
Street2:	
Street3:	
City:	Atlanta
StateID:	GA
Province:	

Zip:	<input type="text" value="40490"/>
CountryID:	<input type="text" value="USA"/>
Email:	<input type="text" value="JSmith@hotmail.com"/>
DayPhone:	<input type="text" value="912-333-4444"/>
NightPhone:	<input type="text" value="912-888-9999"/>
Fax:	<input type="text"/>
Mobile:	<input type="text" value="912-777-8888"/>
Status:	<input type="text"/>
ExtData:	<input type="text"/>
<input type="button" value="Invoke"/>	

Response

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

```
= <RecurringResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/Admin/ws">
```

```
<CustomerKey>11</CustomerKey>
```

```
<code>OK</code>
```

```
<error>OK</error>
```

```
<Partner />
```

```
<Vendor>1</Vendor>
```

```
<Username>vital</Username>
```

```
</RecurringResult>
```

ManageContractAddDaysToNextBillDt

This web service allows for adding days to the next billing:

<https://secure.redfinnet.com/admin/ws/recurring.aspx?op=ManageContractAddDaysToNextBillDt>.

Parameter	Value
Username	Required. The username of the admin user.
Password	Required. The password of the admin user
Vendor	Required. The numerical Vendor/Merchant Key.
CustomerKey	Required. The numerical customer key.
ContractKey	Required for TransType UPDATE and DELETE. The numerical contract key.
NumOfDays	The number of days to be added.
ExtData	Optional. Extended Data.

Example

Parameter	Value
Username:	<input type="text" value="vital"/>
Password:	<input type="text" value="1234"/>
Vendor:	<input type="text" value="1"/>
CustomerKey:	<input type="text" value="11"/>
ContractKey:	<input type="text" value="7"/>
NumOfDays:	<input type="text" value="15"/>
ExtData:	<input type="text"/>

Response

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

```
= <RecurringResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinnet.com/Admin/ws">
```

<CustomerKey>**11**</CustomerKey>

<ContractKey>**7**</ContractKey>

<CcInfoKey />

<CheckInfoKey />

<code>**OK**</code>

<error>**NextBillDate=2/28/2007**</error>

<Partner />

<Vendor />

<Username>**vital**</Username>

</RecurringResult>

GetNetworkID

This web service allows for returning the debit network ID if the debit card number matches any of these network's bin ranges. If there is a match, the card can likely be used as a debit card and processed through the Debit Network. This BIN range is stored in %EPSROOT%\xmlfiles\CardBin.txt which needs to be periodically updated MANUALLY probably every month or so.

<http://secure.redfinnet.com/smartpayments/validate.aspx?op=GetNetworkID>.

Note: This was tested with a live Debit card number.

Example

Parameter	Value
Username	Required. The username of the admin user.
Password	Required. The password of the admin user
CardNumber	Required. The customer's debit card number.

```
POST /smartpayments/validate.aspx/GetNetworkID HTTP/1.1
Host: secure.redfinnet.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
```

UserName=string&Password=string&CardNumber=string

Parameter	Value
UserName:	<input type="text" value="test"/>
Password:	<input type="text" value="123"/>
CardNumber:	<input type="text" value="XXXXXXXXXXXX2557"/>

Response

<?xml version="1.0" encoding="utf-8" ?>

```

=<Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://secure.redfinet.com/SmartPayments/">

  <Result>0</Result>

  <RespMSG>Interlink</RespMSG>

  <Message>Y</Message>

  <Message1>NetworkID=ILK,NetworkAuthorizerNum=48</Message1>

  <Message2>OfflineSupported=True</Message2>

  <AuthCode>ILK</AuthCode>

  <HostCode>48</HostCode>

</Response>

```

BIN Management Network Table Values

Debit Network	Network ID	Network Authorization Number
Accel	ACL	69
AFFN	AFN	68
Alaska Option	AKO	61
CU24	C24	85
Interlink	ILK	48
Jeanie	JEN	86
Star Northeast (MAC)	MAC	17
Maestro	MAE	40
Nets	NET	83
NYCE	NYC	28

Pulse	PUL	06
Star Southeast	SES	07
Shazam	SHZ	58
Star West	STX	23
TYME	TYM	78

Web Service Response Field

Transact.asmx

Response Field	Data Type	Value Description	Remarks
AuthCode	A string value up to 50 characters	Returns the transaction result code from the payment processor	This value can be either an approval code, for approved transactions, or an error code, for declined transactions
ExtData	A string value up to 500 characters	Returns extra data from the processed transaction	The value of ExtData will be in a specific format. The format typically consists of the name of the data field, an equal sign, and then the value for the data field. Multiple data fields are separated with a comma. See the "Web Service ExtData Response Field Data Elements" for full description of data elements that can be returned. The following is an example of the format: <i>ExtName1=ExtValue1,ExtName2=ExtValue2</i>
GetAVSResult	A string value up to 1 character	Returns the overall address verification result code from the payment processor	When programmatically validating an AVS Result, this value should ALWAYS be used instead of any formatted response message describing the result
GetAVSResultTXT	A string value up to 25 characters	Returns the formatted response message when address verification is performed	Do NOT use this when programmatically validating a transaction's AVS result; please see GetAVSResult field
GetCommercialCard	A string value representing a Boolean value	Returns the payment processor's response indicator that specifies if the card is a commercial card	This value is only applicable to credit card transactions. The card verification number is typically printed on the back of the card and not embossed on the front. It is used as an extra authentication method for "card not present" transactions. When programmatically validating a CV Result, this value should ALWAYS be used instead of any formatted response message describing the result
GetCVResult	A string value up to 1 character	Returns the card verification result code from the payment processor	This value is only applicable to credit card transactions. The card verification number is typically printed on the back of the card and not embossed on the front. It is used as an extra authentication method for "card not present" transactions. When programmatically validating a CV Result, this value should ALWAYS be used instead of any formatted response message describing the result
GetCVResultTXT	A string value up to 25 characters	Returns the formatted response message when card verification is performed	This value is only applicable to credit card transactions. Do NOT use this when programmatically validating a transaction's CV result; please see GetCVResult field
GetStreetMatchTXT	A string value up to 25 characters	Returns the formatted response message when street number address	This value will typically be "Match", for correctly matching the street address, or "No Match", for an incorrect street address

		verification is performed	
GetZipMatchTXT	A string value up to 25 characters	Returns the formatted response message when zip code address verification is performed	This value will typically be "Match", for correctly matching the zip code, or "No Match", for an incorrect zip code
HostCode	A string value up to 30 characters	Typically returns a number which uniquely identifies the transaction in the payment processor	This value may not be returned for all payment processors
Message	A string value up to 50 characters	Returns a formatted response message concerning the processed transaction	This value will typically be "APPROVAL", for approved transactions, or an error message, for declined transactions. Do NOT use this when programmatically validating a transaction's result; please see Result field below
Message1	A string value up to 50 characters	Returns an extra formatted response message giving more information about the processed transaction	The Payment Server will only populate this field when there is applicable information from the payment processor to return
Message2	A string value up to 50 characters	Returns an extra formatted response message giving more information about the processed transaction	The Payment Server will only populate this field when there is applicable information from the payment processor to return
PNRef	A string value representing a signed 32-bit integer	Returns a number which uniquely identifies the transaction in the payment gateway	
RespMSG	A string value up to 50 characters	Returns the response message concerning the processed transaction	This value is typically either Approved or Declined. Do NOT use this when programmatically validating a transaction's result; please see Result field below
Result	A string value representing a signed 32-bit integer	Returns the transaction result code from the payment gateway which signifies the result of the transaction (i.e. approved, decline, etc.)	When programmatically validating a transaction's result, this value should ALWAYS be used instead of any response message describing the result. See the "Result Response Fields Definitions" section for a full list of result values and descriptions

Transact.asmx - ExtData Response Field Data Elements

Data Element Name	Value Description	Remarks
BatchNum	Returns the current batch number, returned by the payment processor, for transactions, settlement, and batch inquiries	Not all payment processors support returning this data element
CardType	Returns the credit card type (VISA, MASTERCARD, etc), payment method (Debit, EBT, or EGC) for card-based payments	This value is not returned for Check/ACH payments
InvNum	Returns the same invoice number for the transaction that was originally sent in the request to the Payment Server	

TrxDetail.asmx

Response Field	Data Type	Value Description	Remarks
Account_Type_CH	A string value up to 10 characters	Returns the card type of the transaction, e.g. VISA, Diners, EBT	
AccountNum_VC	A string value up to 200 characters	Returns the check account number	This field will be masked out with asterisk (*) characters except for the last 4 digits if the System Security Level of the user is set to 1
Acct_Num_CH	A string value up to 200 characters	Returns credit card number	This field will be masked out with asterisk (*) characters except for the last 4 digits if the System Security Level of the user is set to 1
Amount_MN	A string value representing a signed 64-bit real number	Returns the check's total amount	
Approval_Code_CH	A string value up to 50 characters	Returns the response code from the payment processor	
Auth_Amt_MN	A string value representing a signed 64-bit real number	Returns the authorized amount of a card transaction	
Authorization	A string value representing a signed 64-bit real number	Returns the dollar amount of all Authorization (PreAuth) transactions	
Authorization_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all Authorization (PreAuth) transactions	
AVS_Resp_CH	A string value up to 1 character	Returns the address verification result code from the payment processor	
AVS_Resp_Txt_VC	A string value up to 25 characters	Returns the formatted response message when address verification is performed	
Batch_Number	A string value up to 10 characters	Returns the batch number for the transaction that was returned by the payment processor	Not all payment processors support returning this data element

Capture	A string value representing a signed 64-bit real number	Returns the dollar amount of all Capture transactions	This value will always return "0"
Capture_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all Capture transactions	This value will always return "0"
Card_Info_Key	A string value representing a signed 32-bit integer	Returns the primary key of the CC_Info_T table in the database	
Cash_Back_Amt_MN	A string value representing a signed 64-bit real number	Returns the cash back amount for a debit or EBT transaction	
CheckNum_CH	A string value up to 10 characters	Returns the check number	
Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all transactions	
CustomerID	A string value up to 50 characters	Returns the Customer ID of a customer to which the transaction belongs to	
CV_Resp_CH	A string value up to 1 character	Returns the card verification result code from the payment processor	
CV_Resp_Txt_VC	A string value up to 25 characters	Returns the formatted response message when card verification is performed	
Date_DT	A string value representing a date and time	Returns the date on which the transaction is first made	
ERROR	A string value up to 200 characters	Returns an error message when a problem occurs during the transaction processing	
Exp_CH	A string value up to 10 characters	Returns the credit card expiration date	
ForceCapture	A string value representing a signed 64-bit real number	Returns the dollar amount of all ForceCapture (ForceAuth) transactions	
ForceCapture_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all ForceCapture (ForceAuth) transactions	
Host_Date_CH	A string value up to 10	Returns the payment	

	characters	processor's date on which the transaction is performed	
Host_Ref_Num_CH	A string value up to 30 characters	Returns a number which uniquely identifies the transaction for the payment processor	
Host_Time_CH	A string value up to 10 characters	Returns the payment processor's time at which the transaction was performed	
Invoice_ID	A string value up to 100 characters	Returns the transaction's Invoice number	
IP_VC	A string value up to 15 characters	Returns the IP address of the client machine from which the transaction was processed	
Last_Update_DT	A string value representing a date and time	Returns the date and time on which the transaction is last modified	
Manual	A string value representing a Boolean value	Returns the card was swiped or not	
Merchant_Key	A string value representing a signed 32-bit integer	Returns a number which uniquely identifies a merchant	
Name_on_Card_VC	A string value up to 25 characters	Returns the name of the cardholder	
NameOnCheck_VC	A string value up to 25 characters	Returns the check payer's name on the check	
Orig_TRX_HD_Key	A string value representing a signed 32-bit integer	Returns the TRX_HD_Key on which the current transaction is based	
Payment_Type_ID	A string value up to 10 characters	Returns the payment type, e.g. ECHECK	
PostAuth	A string value representing a signed 64-bit real number	Returns the dollar amount of all PostAuth transactions	
PostAuth_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all PostAuth transactions	
Processor_ID	A string value up to 10 characters	Returns the name the payment processor, e.g.	

		Vital	
Receipt	A string value representing a signed 64-bit real number	Returns the dollar amount of all transactions with a Receipt	This value will always return "0"
Receipt_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all transactions with a Receipt	This value will always return "0"
Ref_Number_CH	A string	Not currently used	This field is not the unique transaction identifier (also called PNRef) of the Payment Server. See the field TRX_HD_Key for the PNRef value
Register_Number_CH	A string value up to 10 characters	Returns the register number of a transaction	
RepeatSale	A string value representing a signed 64-bit real number	Returns the dollar amount of all RepeatSale (Recurring Billing/Installment) transactions	
RepeatSale_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all RepeatSale (Recurring Billing/Installment) transactions	
Reseller_Key	A string value representing a signed 32-bit integer	Returns the primary key of the Reseller_T table in the database	
Result_CH	A string value up to 50 characters	Returns the transaction processing result, e.g. 0, 12. "0" for approval, "12" for decline	
Result_Msg_VC	A string value up to 50 characters	Returns the check transaction's processing result	
Result_Msg1_VC	A string value up to 50 characters	Returns an extra formatted response message giving more information about the processed transaction	
Result_Msg2_VC	A string value up to 50 characters	Returns an extra formatted response message giving more information about the processed transaction	
Result_Txt_VC	A string value up to 150 characters	Returns the text message of either approval or decline for	

		the transaction processing result	
Return	A string value representing a signed 64-bit real number	Returns the dollar amount of all Return (Credit) transactions	
Return_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all Return (Credit) transactions	
Sale	A string value representing a signed 64-bit real number	Returns the dollar amount of all Sale transactions	
Sale_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all Sale transactions	
Settle_Date_DT	A string value representing a date and time	Returns the date on which the transaction is settled	
Settle_Flag_CH	A string value representing a Boolean value	Returns if the transaction is settled or not	
StateCode_CH	A string value up to 10 characters	Returns the state code	
Street_CH	A string value up to 25 characters	Returns the billing street address of the credit card	
SureCharge_Amt_MN	A string value representing a signed 64-bit real number	Returns the sure charge amount of a transaction	
Tip_Amt_MN	A string value representing a signed 64-bit real number	Returns the tip amount of a transaction	
Trans_Type_ID	A string value up to 20 characters	Returns the transaction type, e.g. Sale, Credit	
Transport_Method	A string	Returns the Transportation Method	Only for use with Dial-up transactions
Transport_EndPoint	A string	Returns the Transportation's Ending Destination	Only for use with Dial-up transactions
TransitNum_VC	A string value up to 200 characters	Returns the transit/routing number	This field will be masked out with asterisk (*) characters except for the last 4 digits if the System Security Level of the user is set to 1
TRX_Card_Key	A string value representing a signed	Returns the primary key of the TRX_Card_T table	

	32-bit integer	in the database	
TRX_Check_Key	A string value representing a signed 32-bit integer	Returns the primary key of the TRX_Check_T table in the database	
TRX_HD_Key	A string value representing a signed 32-bit integer	Returns the primary key of the TRX_Header_T table in the database	This field is the unique transaction identifier (also called PNRRef) of the Payment Server. Use its value when submitting transactions based on a previous transaction (i.e. Voids) through the Transact.asmx Web Service
TRX_Settle_Key	A string value representing a signed 32-bit integer	Returns the primary key of the TRX_Settle_T table in the database	
TRX_Settle_Msg_VC	A string value up to 25 characters	Returns the transaction's settlement message	
Type_CH	A string value up to 10 characters	Returns the credit card type, e.g. VISA, MASTERCARD	
User_Name_VC	A string value up to 25 characters	Returns the username, under which the transactions were made	
Void_Flag_CH	A string value representing a Boolean value	Returns the transaction is voided or not	
Zip_CH	A string value up to 10 characters	Returns the billing zip code of the credit card	
Auth_Amt_MN	A string value representing a signed 64-bit real number	Returns the authorized amount of a card transaction	
Authorization	A string value representing a signed 64-bit real number	Returns the dollar amount of all Authorization (PreAuth) transactions	
Authorization_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all Authorization (PreAuth) transactions	
AVS_Resp_CH	A string value up to 1 character	Returns the address verification result code from the payment processor	
AVS_Resp_Txt_VC	A string value up to 25 characters	Returns the formatted response message when	

		address verification is performed	
Batch_Number	A string value up to 10 characters	Returns the batch number for the transaction that was returned by the payment processor	Not all payment processors support returning this data element
Capture	A string value representing a signed 64-bit real number	Returns the dollar amount of all Capture transactions	This value will always return "0"
Capture_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all Capture transactions	This value will always return "0"
Card_Info_Key	A string value representing a signed 32-bit integer	Returns the primary key of the CC_Info_T table in the database	
Cash_Back_Amt_MN	A string value representing a signed 64-bit real number	Returns the cash back amount for a debit or EBT transaction	
CheckNum_CH	A string value up to 10 characters	Returns the check number	
Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all transactions	
CustomerID	A string value up to 50 characters	Returns the Customer ID of a customer to which the transaction belongs to	
CV_Resp_CH	A string value up to 1 character	Returns the card verification result code from the payment processor	
CV_Resp_Txt_VC	A string value up to 25 characters	Returns the formatted response message when card verification is performed	
Date_DT	A string value representing a date and time	Returns the date on which the transaction is first made	
ERROR	A string value up to 200 characters	Returns an error message when a problem occurs during the transaction processing	
Exp_CH	A string value up to 10 characters	Returns the credit card expiration date	
ForceCapture	A string value representing a signed	Returns the dollar amount of all	

	64-bit real number	ForceCapture (ForceAuth) transactions	
ForceCapture_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all ForceCapture (ForceAuth) transactions	
Host_Date_CH	A string value up to 10 characters	Returns the payment processor's date on which the transaction is performed	
Host_Ref_Num_CH	A string value up to 30 characters	Returns a number which uniquely identifies the transaction for the payment processor	
Host_Time_CH	A string value up to 10 characters	Returns the payment processor's time at which the transaction was performed	
Invoice_ID	A string value up to 100 characters	Returns the transaction's Invoice number	
IP_VC	A string value up to 15 characters	Returns the IP address of the client machine from which the transaction was processed	
Last_Update_DT	A string value representing a date and time	Returns the date and time on which the transaction is last modified	
Manual	A string value representing a Boolean value	Returns the card was swiped or not	
Merchant_Key	A string value representing a signed 32-bit integer	Returns a number which uniquely identifies a merchant	
Name_on_Card_VC	A string value up to 25 characters	Returns the name of the cardholder	
NameOnCheck_VC	A string value up to 25 characters	Returns the check payer's name on the check	
Orig_TRX_HD_Key	A string value representing a signed 32-bit integer	Returns the TRX_HD_Key on which the current transaction is based	
Payment_Type_ID	A string value up to 10 characters	Returns the payment type, e.g. ECHECK	
PostAuth	A string value	Returns the dollar	

	representing a signed 64-bit real number	amount of all PostAuth transactions	
PostAuth_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all PostAuth transactions	
Processor_ID	A string value up to 10 characters	Returns the name the payment processor, e.g. Vital	
Receipt	A string value representing a signed 64-bit real number	Returns the dollar amount of all transactions with a Receipt	This value will always return "0"
Receipt_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all transactions with a Receipt	This value will always return "0"
Ref_Number_CH	A string	Not currently used	This field is not the unique transaction identifier (also called PNRef) of the Payment Server. See the field TRX_HD_Key for the PNRef value
Register_Number_CH	A string value up to 10 characters	Returns the register number of a transaction	
RepeatSale	A string value representing a signed 64-bit real number	Returns the dollar amount of all RepeatSale (Recurring Billing/Installment) transactions	
RepeatSale_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all RepeatSale (Recurring Billing/Installment) transactions	
Reseller_Key	A string value representing a signed 32-bit integer	Returns the primary key of the Reseller_T table in the database	
Result_CH	A string value up to 50 characters	Returns the transaction processing result, e.g. 0, 12. "0" for approval, "12" for decline	
Result_Msg_VC	A string value up to 50 characters	Returns the check transaction's processing result	
Result_Msg1_VC	A string value up to 50 characters	Returns an extra formatted response message giving more information about the processed transaction	

Result_Msg2_VC	A string value up to 50 characters	Returns an extra formatted response message giving more information about the processed transaction	
Result_Txt_VC	A string value up to 150 characters	Returns the text message of either approval or decline for the transaction processing result	
Return	A string value representing a signed 64-bit real number	Returns the dollar amount of all Return (Credit) transactions	
Return_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all Return (Credit) transactions	
Sale	A string value representing a signed 64-bit real number	Returns the dollar amount of all Sale transactions	
Sale_Cnt	A string value representing a signed 32-bit integer	Returns the transaction count of all Sale transactions	
Settle_Date_DT	A string value representing a date and time	Returns the date on which the transaction is settled	
Settle_Flag_CH	A string value representing a Boolean value	Returns if the transaction is settled or not	
StateCode_CH	A string value up to 10 characters	Returns the state code	
Street_CH	A string value up to 25 characters	Returns the billing street address of the credit card	
SureCharge_Amt_MN	A string value representing a signed 64-bit real number	Returns the sure charge amount of a transaction	
Tip_Amt_MN	A string value representing a signed 64-bit real number	Returns the tip amount of a transaction	
Total_Amt_MN	A string value representing a signed 64-bit real number	Returns the total amount of a transaction	
Trans_Type_ID	A string value up to 20 characters	Returns the transaction type, e.g. Sale, Credit	
Transport_Method	A string	Returns the Transportation Method	Only for use with Dial-up transactions
Transport_EndPoint	A string	Returns the	Only for use with Dial-

		Transportation's Ending Destination	up transactions
TransitNum_VC	A string value up to 200 characters	Returns the transit/routing number	This field will be masked out with asterisk (*) characters except for the last 4 digits if the System Security Level of the user is set to 1
TRX_Card_Key	A string value representing a signed 32-bit integer	Returns the primary key of the TRX_Card_T table in the database	
TRX_Check_Key	A string value representing a signed 32-bit integer	Returns the primary key of the TRX_Check_T table in the database	
TRX_HD_Key	A string value representing a signed 32-bit integer	Returns the primary key of the TRX_Header_T table in the database	This field is the unique transaction identifier (also called PNRRef) of the Payment Server. Use its value when submitting transactions based on a previous transaction (i.e. Voids) through the Transact.asmx Web Service
TRX_Settle_Key	A string value representing a signed 32-bit integer	Returns the primary key of the TRX_Settle_T table in the database	
TRX_Settle_Msg_VC	A string value up to 25 characters	Returns the transaction's settlement message	
Type_CH	A string value up to 10 characters	Returns the credit card type, e.g. VISA, MASTERCARD	
User_Name_VC	A string value up to 25 characters	Returns the username, under which the transactions were made	
Void_Flag_CH	A string value representing a Boolean value	Returns the transaction is voided or not	
Zip_CH	A string value up to 10 characters	Returns the billing zip code of the credit card	

Response Values

Result Response Field Definitions (Error Codes)

The list below contains result codes returned in the Result response field of the XMLPayResponse when using a transaction processing Transact.asmx web service operation (i.e. ProcessCreditCard, ProcessCheck, etc). A decline returned by the payment processor for this response field is value twelve (12) or thirteen (13). An approval is value zero (0). Any other value is an error code, which is returned by the payment gateway and not by the payment processor. Please note that when programmatically validating a transaction's result, this value should be used instead of any response message describing the result. I.e. do *not* use RespMSG or Message response fields, as these values may vary. Please note that this list is subject to change without prior notice.

Value	Description
-100	Transaction NOT Processed; Generic Host Error
0	Approved
1	User Authentication Failed
2	Invalid Transaction
3	Invalid Transaction Type
4	Invalid Amount
5	Invalid Merchant Information
7	Field Format Error
8	Not a Transaction Server
9	Invalid Parameter Stream
10	Too Many Line Items
11	Client Timeout Waiting for Response
12	Decline
13	Referral
14	Transaction Type Not Supported In This Version
19	Original Transaction ID Not Found
20	Customer Reference Number Not Found
22	Invalid ABA Number

23	Invalid Account Number
24	Invalid Expiration Date
25	Transaction Type Not Supported by Host
26	Invalid Reference Number
27	Invalid Receipt Information
28	Invalid Check Holder Name
29	Invalid Check Number
30	Check DL Verification Requires DL State
40	Transaction did not connect (to NCN because SecureNCIS is not running on the web server)
50	Insufficient Funds Available
99	General Error
100	Invalid Transaction Returned from Host
101	Timeout Value too Small or Invalid Time Out Value
102	Processor Not Available
103	Error Reading Response from Host
104	Timeout waiting for Processor Response
105	Credit Error
106	Host Not Available
107	Duplicate Suppression Timeout
108	Void Error
109	Timeout Waiting for Host Response
110	Duplicate Transaction
111	Capture Error
112	Failed AVS Check
113	Cannot Exceed Sales Cap
1000	Generic Host Error
1001	Invalid Login
1002	Insufficient Privilege or Invalid Amount
1003	Invalid Login Blocked
1004	Invalid Login Deactivated

1005	Transaction Type Not Allowed
1006	Unsupported Processor
1007	Invalid Request Message
1008	Invalid Version
1010	Payment Type Not Supported
1011	Error Starting Transaction
1012	Error Finishing Transaction
1013	Error Checking Duplicate
1014	No Records To Settle (in the current batch)
1015	No Records To Process (in the current batch)

AVS Response Codes

The following table contains the possible response values returned for address verification (AVS).

Value	Description
X	Exact: Address and nine-digit Zip match
Y	Yes: Address and five-digit Zip match
A	Address: Address matches, Zip does not
Z	5-digit Zip: 5-digit Zip matches, address doesn't
W	Whole Zip: 9-digit Zip matches, address doesn't
N	No: Neither address nor Zip matches
U	Unavailable: Address information not available
G	Unavailable: Address information not available for international transaction
R	Retry: System unavailable or time-out
E	Error: Transaction unintelligible for AVS or edit error found in the message that prevents AVS from being performed
S	Not Supported: Issuer doesn't support AVS service
B	* Street Match: Street addresses match for international transaction, but postal code doesn't
C	* Street Address: Street addresses and postal code not verified for international transaction
D	* Match: Street addresses and postal codes match for international transaction
I	* Not Verified: Address Information not verified for International transaction
M	* Match: Street addresses and postal codes match for international transaction
P	* Postal Match: Postal codes match for international transaction, but street address doesn't
0	** No response sent
5	Invalid AVS response

* These values are Visa specific.

** These values are returned by the Payment Server and not the processor.

CV Response Codes

The following table contains the possible response values returned for a CVV2/CVC2/CID check.

Value	Description
M	CVV2/CVC2/CID Match
N	CVV2/CVC2/CID No Match
P	Not Processed
S	Issuer indicates that the CV data should be present on the card, but the merchant has indicated that the CV data is not present on the card.
U	Unknown / Issuer has not certified for CV or issuer has not provided Visa/MasterCard with the CV encryption keys.
X	Server Provider did not respond

Valid Parameter Input Characters

The table below displays all allowable characters (unless otherwise noted) that are accepted by the Payment Server. Characters are displayed in Courier New font. All other characters may cause undesirable results.

Table 1. Valid Data Characters

DEC	HEX	Character	DEC	HEX	Character	DEC	HEX	Character
32	20	Space	63	3F	?	96	60	`
33	21	!	64	40	@	97	61	a
34	22	"	65	41	A	98	62	b
35	23	#	66	42	B	99	63	c
36	24	\$	67	43	C	100	64	d
37	25	%	68	44	D	101	65	e
38	26	&	69	45	E	102	66	f
39	27	'	70	46	F	103	67	g
40	28	(71	47	G	104	68	h
41	29)	72	48	H	105	69	i
42	2A	*	73	49	I	106	6A	j
43	2B	+	74	4A	J	107	6B	k
44	2C	,	75	4B	K	108	6C	l
45	2D	-	76	4C	L	109	6D	m
46	2E	.	77	4D	M	110	6E	n
47	2F	/	78	4E	N	111	6F	o
48	30	0	79	4F	O	112	70	p
49	31	1	80	50	P	113	71	q
50	32	2	81	51	Q	114	72	r
51	33	3	82	52	R	115	73	s
52	34	4	83	53	S	116	74	t
53	35	5	84	54	T	117	75	u
54	36	6	85	55	U	118	76	v
55	37	7	86	56	V	119	77	w
56	38	8	87	57	W	120	78	x
57	39	9	88	58	X	121	79	y
58	3A	:	89	59	Y	122	7A	z
59	3B	;	90	5A	Z	123	7B	{
60	3C	<	92	5C	\	124	7C	
61	3D	=	94	5E	^	125	7D	}
62	3E	>	95	5F	_	126	7E	~

Character Removal

The table below displays all acceptable characters that must be removed by the Payment Server before submitting information to the Web Service operations. See each input parameter for each Web Service operation in order to know which input parameters will

have these characters removed. This character removal ensures that the Payment Servers' internal XML parsers can properly read the information of the Web Service operation. Characters in the table are displayed in Courier New font.

Many XML Parsers will encode these characters for you. In this case, the characters *will not* be converted back to their proper values by the Payment Server; they will be taken literally. Also, if you pass the encoded character through an input parameter that removes the characters listed in the table below, then certain characters may be removed (see examples below). However, if you are not using a parser, or if the parser does not handle this encoding, then the characters in the table listed below may still be removed, depending on the input parameter for the Web Service operation you are using.

Table 2. XML Character Removal

Character	XML Parser Encoding
<	<
>	>
&	&
'	'
"	"

Example

The following example shows how characters would be removed if the data was passed through the NameOnCard parameter of the ProcessCreditCard operation.

Valid: *John James*

Invalid: *John & James becomes John James*

Invalid: *John & James becomes John amp; James*