



Regal Technologies Guide to Integrating and Using the

RegalTek DevConnect Web Service

Record of Reviews and Versions

VERSION ID	DATE	AUTHOR	DESCRIPTION
2.0.1	5/2/11	Regal Technologies, LLC	Regaltek DevConnect Web Service

This document is the property of Regal Technologies, LLC and contains proprietary unpublished information, designs, algorithms, protocols, innovations, and concepts that are protected under copyright and trade secret laws of the United States. The information contained is confidential and is to be protected from any unauthorized distribution. Reproduction of this document by any means, including photocopying or translation into other languages, is prohibited. While reasonable efforts have been taken in preparation of this document to assure its accuracy, Regal Technologies, LLC assumes no liability resulting from any errors or omissions in this document, or from the use of the information herein. Copyright © Regal Technologies, LLC, All rights reserved.

Disclaimer of Warranties and Limitations of Liabilities

Regal Technologies has taken due care in preparing this document; however, nothing contained herein modifies or alters in any way the standard terms and conditions of the Regal Technologies purchase, lease, or license agreement by which the product was acquired, nor increase in any way Regal Technologies' liability to the customer. In no event shall Regal Technologies be liable for incidental or consequential damages because of information contained in this document or any related materials.

Software Notice

All Regal Technologies software products are licensed to customers in accordance with the terms and conditions of Regal Technologies' standard software license. No title or ownership of Regal Technologies software is transferred, and any use of the software beyond the terms of the aforesaid license, without the written authorization of Regal Technologies, is prohibited.

General Notice

The information and specifications in this document are subject to change without notice. The information on this document is protected by copyright. Except as specifically permitted, no portion of this document may be distributed or reproduced by any means, or in any form, without Regal Technologies' prior written permission.

All Regal Technologies products and publications are commercial in nature. The software, publications, and software documentation available on this web site are "Commercial Items", as that term is defined in 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation", as such terms are defined in 48 C.F.R. §252.227-7014(a)(5) and 48 C.F.R. §252.227-7014(a)(1), and used in 48 C.F.R. §12.212 and 48 C.F.R. 227.7202, as applicable. Pursuant to 48 C.F.R. §12.212, 48 C.F.R. §252.227-7015, 48 C.F.R. §227.7202 through 227.7202-4, 48 C.F.R. §52.227-19, and other relevant sections of the Code of Federal Regulations, as applicable, Regal Technologies' publications, commercial computer software, and commercial computer software documentation are distributed and licensed to United States Government end users with only those rights as granted to all other end users, according to the terms and conditions contained in the license agreements that accompany the products and software documentation, and the terms and conditions herein.

Regal Technologies is a registered trademark of Regal Technologies Limited Liability Corporation.

Regal Technologies, LLC
Postal Box 670
Severna Park, Maryland 21146-0670 USA
Telephone (410) 975-0688
Toll Free (866) 766-1066
Fax (240) 331-9166
www.regaltek.com

Copyright © Regal Technologies, LLC, All rights reserved.

Table of Contents

1.0 Using the RegalTek Payment System.....	5
2.0 Request Values	6
3.0 Response Values	9
4.0 Example SOAP Requests and Responses.....	10
4.1.0 Request to perform a One-Time Credit Card Transaction	10
4.1.1 Response after performing a One-Time Credit Card Transaction	11
4.2.0 Request to perform a One-Time ACH Check Transaction	12
4.2.1 Response after performing a One-Time ACH Check Transaction	13
4.3.0 Request to Create a RecurPay Plan for a Customer	14
4.3.1 Response after performing a RecurPay Plan Creation.....	15
4.4.0 Request to Update a RecurPay Plan for a Customer.....	16
4.5.0 Request to Cancel a RecurPay Plan	18
4.5.1 Response after performing a RecurPay Plan Cancelation.....	19
5.0 Testing	20
5.1 ACH Check Test Payments.....	20
5.2 CREDITCARD Test Payments.....	20
5.3 Testing RecurPay	21
6.0 Integrating With SOAP.....	22
6.1 .NET Languages.....	22
6.2 Java / Sun	22
6.3 PHP	22
6.4 PERL	22

6.5 Python22

6.6 Additional Resources23

1.0 Using the RegalTek Payment System

The RegalTek payment system is a web service designed to run 24 hours a day 7 days a week listening for SOAP requests coming to it. It's purpose is to manage customer information, store payment information and allow RegalTek clients to process ACH and CC transactions with a simple web service call.

It is located at the following URL:

<https://regaltek.secured-server.biz/RegalPayment/services/ProcessRequest>

and the WSDL is located here:

<https://regaltek.secured-server.biz/RegalPayment/services/ProcessRequest?wsdl>

The system will accept a standard SOAP request and return a SOAP response. To keep things simple, there is only one 'operation' built into the above web service. The operation is called 'processCommand'. Please see the examples section (4.0) for an idea on how the request and response should be structured based on this operation.

The following sections contain details about the request fields and the response values returned by the web service.

2.0 Request Values

The following table describes the field names and values to use when posting a SOAP request to the Regal Technologies system. There are two payment commands labeled 'T' for transact and 'Q' for query that a RegalTek client should use.

Please review the table below for information on all the possible request fields required to send commands. Please see section 4 for examples.

Field	Value	Type	Notes
merchantCode	The authentication token supplied to client by Regal Technologies	50 CHARs alphanumeric	Required
command	The command field tells the API which function you are trying to perform. The following commands are supported: 'transact' – attempt to process a one-time transaction 'query' – attempt to look up historical data for a previously submitted transaction	1 CHAR	Required
test	Either 'true' or anything else (including omitting the field altogether).	8 CHARs	Optional. Defaults to 'false'. Specify 'true' if you DO want this to be a test payment. If this is anything other than 'true', it will be considered a live payment.
paymentMethod	Either ACH or CREDITCARD	16 CHARs	Required for command=transact
paymentSubMethod	One of (Checking, Savings, Business, Visa, MasterCard, Amex, Discover, Unknown)	16 CHARs	Required for command=transact
branchNumber	The merchant's branch number if applicable	16 CHARs	Optional
customerAccountNumber	The end user's customer account number if the merchant uses one.	16 CHARs	Optional
invoiceNumber	An optional invoice number	16 CHARs	Optional
bankRoutingNumber	The customer's bank routing or aba number	9 CHARs	Required for paymentMethod=ACH
bankAccountNumber	The customer's bank account number in cases of ACH	6 CHARs	Required for paymentMethod=ACH
insertType	Either TEL, WEB or RTP. Please speak to your merchant account provider as to what this should be. Defaults to WEB if value provided is blank or anything unrecognized.	3 CHARs	Optional. Defaults to WEB unless TEL is specified. RTP puts the check payment in the 'Ready To Process' queue.
checkDate	The date to process the check in YYYY-MM-DD format	10 CHARs	Required for paymentMethod=ACH
creditCardNumber	The customer's credit card number in cases of CREDITCARD	16 CHARs	Required for paymentMethod=CREDITCARD
expireMonth	The credit card expiry month	2 CHARs	Required for paymentMethod=CREDITCARD
expireYear	The credit card expiry year	4 CHARs	Required for paymentMethod=CREDITCARD
cvvCode	This is the 3 or 4 digit security code	4 CHARs	Optional, but highly recommended.

	from the back of the customer's payment card.		
paymentAmount	The amount to process	12 CHARs	Required for command=transact
serviceFee	The service fee to process if applicable	12 CHARs	Optional – will be added to the paymentAmount before processing
billFirstName	The customer's first name	25 CHARs	Required for command=transact
billLastName	The customer's last name	25 CHARs	Required for command=transact
billCompany	The customer's company if applicable	50 CHARs	Optional
billAddress	The customer's street address	50 CHARs	Required for command=transact
billCity	The customer's city	25 CHARs	Required for command=transact
billState	The two digit state abbreviation	2 CHARs	Required for command=transact
billZip	The 5 digit customer zip code	5 CHARs	Required for command=transact
billCountry	The customer's country	2 CHARs	Optional
billPhone	The customer's 10 digit phone number	10 CHARs	Optional
billEmail	The customer's email address	50 CHARs	Optional
billDriversLicenseNumber	The drivers license number if applicable for ACH transactions	24 CHARs	Optional
billDriversLicenseState	The drivers license state if applicable for ACH transactions	2 CHARs	Optional
shipFirstName	The customer's ship to first name	25 CHARs	Optional
shipLastName	The customer's ship to last name	25 CHARs	Optional
shipCompany	The customer's ship to company	50 CHARs	Optional
shipAddress	The customer's shipping address	50 CHARs	Optional
shipCity	The customer's shipping city	25 CHARs	Optional
shipState	The customer's shipping state	2 CHARs	Optional
shipZip	The customer's shipping zip	5 CHARs	Optional
shipCountry	The customer's shipping country	2 CHARs	Optional
shipPhone	The customer's ship to phone	10 CHARs	Optional
shipEmail	The customer's shipping email	50 CHARs	Optional
description	An optional payment description	255 CHARs	Optional
trackingNumber	The trackingNumber of a previous transaction. Used to retrieve previous transaction's status.	16 CHARs	Required for command=query
recurringAmount	For recurpay accounts, it's the amount to charge.	12 CHAR	Required for command = CREATE_RECURPAY_CUSTOMER
recurringStartDate	The date to start a recurpay payment plan for this customer in YYYY-MM-DD format	10 CHAR	Required for command = CREATE_RECURPAY_CUSTOMER
recurringFrequency	In the case of setting up a recurpay customer, this is the frequency they should be billed. Values can be W for Weekly, B for Bi-Weekly, M for Monthly and T for Twice-Monthly	1 CHAR	Required for command = CREATE_RECURPAY_CUSTOMER
recurringWeekDay	In the case of Weekly or Bi-Weekly plans, this is a two letter day abbreviation of MO, TU, WE, TH, FR, SA or SU	2 CHAR	Required for command = CREATE_RECURPAY_CUSTOMER when recurringFrequency=W or B
recurringDateOfMonth1	In the case of Monthly or Twice Monthly, this is the 2 digit date of the month to bill on. Example 16	2 CHAR	Required for command = CREATE_RECURPAY_CUSTOMER when recurringFrequency=M or T
recurringDateOfMonth2	In the case of Twice Monthly this is the second monthly payment date, Example 30. Note that for shorter months the payment occurs on the last day of the month, so in Feb a payment due on the 30 th would be processed on the 28 th .	2 CHAR	Required for command = CREATE_RECURPAY_CUSTOMER when recurringFrequency=T
numberOfPayments	The number of payments that should	6 CHAR	Required for command =

	be charged in a recurring plan. Example 12.		CREATE_RECURPAY_CUSTOMER
--	--	--	--------------------------

3.0 Response Values

The following values represent the different parts of a response that can come back as a result of running a request using values in section 2.0.

Field	Value	Type	Notes
command	Original command this response is referring to	1 CHAR	Echoed from request
commandResponseCode	The response to attempting the 'command' 1 for successful, 2 for declined, 3 for error	1 NUM	This is the overall response to the request 'command' field. Do not confuse this with the paymentResponseCode
commandResponseText	The text describing the commandResponseCode	120 CHARs	A description of the declined message or error that occurred by attempting to process the request 'command'.
paymentAmount	The amount attempted to charge.	12 CHARs amount	For command=transact or command=query.
paymentResponseCode	The payment 'response code' returned from the transaction	1 NUM	1=approved, 2=declined, 3=error.
paymentResponseText	The 'response text' returned from attempting a transaction	120 CHARs	The description of the paymentResponseCode. This will likely be 'Transaction Inserted Successfully' or just 'Success'.
paymentTransactionID	The ACH of CC transaction ID for 'T' commands. This can be used to lookup transactions in the Regal reporting tools. It is provided by the back end bank provider.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
trackingNumber	The RegalTek tracking number for the payment.	16 CHARs	This is unique to all transactions.
approvalCode	The 5-6 digit approval code returned for CC transactions from the back end bank provider.	6 CHARs	This is not unique but is issued and may be used to verify payment with the credit card companies.
customerAccountNumber	An echo back of the customerAccountNumber sent in the request.	16 CHARs	This is mostly for clarity when creating, updating and cancelling recurpay customers as the customerAccountNumber is the unique key to the customer's recurpay set up.

4.0 Example SOAP Requests and Responses

Below you will find example SOAP post requests and responses to communicate with the API. It is generally up to you to determine which library or method to use with your programming language of choice to communicate to a SOAP based API, but all modern languages have common and popular methods of integration for this purpose. A few useful links can be found in the 'Integrating With Soap' section for the most common languages.

4.1.0 Request to perform a One-Time Credit Card Transaction

The following is an example request to the API to signal that a credit card transaction should be performed in real time and the response should be returned.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
  <soapenv:Header/>
  <soapenv:Body>
    <proc:processCommand>
      <proc:test>TRUE</proc:test>
    <proc:merchantCode>K3L45JL3K45J34KL435JJK45JLK345JLK543LKJ543JLKUII</proc:merchantCode>
    <proc:command>TRANSACT</proc:command>
    <proc:paymentMethod>CREDITCARD</proc:paymentMethod>
    <proc:paymentSubMethod>Visa</proc:paymentSubMethod>
    <proc:creditCardNumber>4242424242424242</proc:creditCardNumber>
    <proc:expireMonth>01</proc:expireMonth>
    <proc:expireYear>2012</proc:expireYear>
    <proc:cvvCode>123</proc:cvvCode>
    <proc:paymentAmount>1.01</proc:paymentAmount>
    <proc:serviceFee></proc:serviceFee>
    <proc:billFirstName>test</proc:billFirstName>
    <proc:billLastName>test</proc:billLastName>
    <proc:billAddress>123 test st</proc:billAddress>
    <proc:billZip>90210</proc:billZip>
    <proc:billEmail>support@e-complish.com</proc:billEmail>
    <proc:billState>CA</proc:billState>
    <proc:billPhone>444-555-6666</proc:billPhone>
    <proc:billCity>90210</proc:billCity>
  </proc:processCommand>
</soapenv:Body>
</soapenv:Envelope>
```

4.1.1 Response after performing a One-Time Credit Card Transaction

And here is the response that one might expect from the example in 4.1.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <soapenv:Body>
    <processCommandResponse xmlns="http://processor">
      <processCommandReturn>
        <approvalCode>000000</approvalCode>
        <command>TRANSACT</command>
        <commandResponseCode>1</commandResponseCode>
        <commandResponseText>Successfully processed transaction.</commandResponseText>
        <paymentMethod>CREDITCARD</paymentMethod>
        <paymentAmount>1.01</paymentAmount>
        <paymentResponseCode>1</paymentResponseCode>
        <paymentResponseText>(TESTMODE) This transaction has been
approved.</paymentResponseText>
        <paymentTransactionID>0</paymentTransactionID>
        <trackingNumber>1280892202844</trackingNumber>
      </processCommandReturn>
    </processCommandResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

4.2.0 Request to perform a One-Time ACH Check Transaction

The following is an example request to the API to signal that a check (ACH) transaction should be performed in real time and the response should be returned.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
  <soapenv:Header/>
  <soapenv:Body>
    <proc:processCommand>
      <proc:test>TRUE</proc:test>
    <proc:merchantCode>K3L45JL3K45J34KL435JJK45JLK345JLK543LKJ543JLKUII</proc:merchantCode>
    <proc:command>TRANSACT</proc:command>
    <proc:paymentMethod>ACH</proc:paymentMethod>
    <proc:paymentSubMethod>Checking</proc:paymentSubMethod>
    <proc:bankRoutingNumber>123456780</proc:bankRoutingNumber>
    <proc:bankAccountNumber>123456</proc:bankAccountNumber>
    <proc:checkDate>2011-12-25</proc:checkDate>
    <proc:paymentAmount>1.01</proc:paymentAmount>
    <proc:serviceFee></proc:serviceFee>
    <proc:billFirstName>test</proc:billFirstName>
    <proc:billLastName>test</proc:billLastName>
    <proc:billAddress>123 test st</proc:billAddress>
    <proc:billZip>90210</proc:billZip>
    <proc:billEmail>support@e-complish.com</proc:billEmail>
    <proc:billState>CA</proc:billState>
    <proc:billPhone>444-555-6666</proc:billPhone>
    <proc:billCity>90210</proc:billCity>
  </proc:processCommand>
</soapenv:Body>
</soapenv:Envelope>
```

4.2.1 Response after performing a One-Time ACH Check Transaction

And here is the response that one might expect from the example in 4.3.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <soapenv:Body>
    <processCommandResponse xmlns="http://processor">
      <processCommandReturn>
        <approvalCode xsi:nil="true"/>
        <command>TRANSACT</command>
        <commandResponseCode>1</commandResponseCode>
        <commandResponseText>Successfully processed transaction.</commandResponseText>
        <paymentMethod>ACH</paymentMethod>
        <paymentAmount>1.01</paymentAmount>
        <paymentResponseCode>1</paymentResponseCode>
        <paymentResponseText>(TESTMODE) Transaction Inserted
Successfully.</paymentResponseText>
        <paymentTransactionID>1280893546734</paymentTransactionID>
        <trackingNumber>1280893546734</trackingNumber>
      </processCommandReturn>
    </processCommandResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

4.3.0 Request to Create a RecurPay Plan for a Customer

The following is an example post to create a recurpay plan for a customer. After the plan is created, it will be activated on the recurringStartDate and automatically every recurringFrequency for a total of numberOfPayments times.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
  <soapenv:Header/>
  <soapenv:Body>
    <proc:processCommand>
<proc:merchantCode>K3L45JL3K45J34KL435JJK45JLK345JLK543LKJ543JLKUII</proc:merchantCode>
      <proc:test>TRUE</proc:test>
      <proc:command>CREATE_RECURPAY_CUSTOMER</proc:command>
      <proc:customerAccountNumber>123456789012</proc:customerAccountNumber>
      <proc:recurringStartDate>2011-01-01</proc:recurringStartDate>
      <proc:recurringFrequency>M</proc:recurringFrequency>
      <proc:recurringDateOfMonth1>01</proc:recurringDateOfMonth1>
      <proc:recurringAmount>19.95</proc:recurringAmount>
      <proc:numberOfPayments>12</proc:numberOfPayments>
      <proc:paymentMethod>CREDITCARD</proc:paymentMethod>
      <proc:paymentSubMethod>Visa</proc:paymentSubMethod>
      <proc:creditCardNumber>4242424242424242</proc:creditCardNumber>
      <proc:expireMonth>01</proc:expireMonth>
      <proc:expireYear>2012</proc:expireYear>
      <proc:cvvCode>123</proc:cvvCode>
      <proc:billFirstName>Jonathan</proc:billFirstName>
      <proc:billLastName>Doe</proc:billLastName>
      <proc:billAddress>123 test st</proc:billAddress>
      <proc:billCity>testville</proc:billCity>
      <proc:billZip>90210</proc:billZip>
      <proc:billState>CA</proc:billState>
      <proc:billPhone>555-666-7777</proc:billPhone>
      <proc:billEmail>jdoe@example.com</proc:billEmail>
    </proc:processCommand>
  </soapenv:Body>
</soapenv:Envelope>
```

4.3.1 Response after performing a RecurPay Plan Creation

And here is the response one might expect after creating a recurring plan for a customer:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <soapenv:Body>
    <processCommandResponse xmlns="http://processor">
      <processCommandReturn>
        <approvalCode xsi:nil="true"/>
        <command>CREATE_RECURPAY_CUSTOMER</command>
        <commandResponseCode>1</commandResponseCode>
        <commandResponseText>Successfully Created RecurPay
Customer.</commandResponseText>
        <customerAccountNumber>123456789012</customerAccountNumber>
        <paymentAmount>19.95</paymentAmount>
        <paymentMethod>CREDITCARD</paymentMethod>
        <paymentResponseCode>1</paymentResponseCode>
        <paymentResponseText>Successfully Created RecurPay
Customer.</paymentResponseText>
        <paymentTransactionID/>
        <trackingNumber/>
      </processCommandReturn>
    </processCommandResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

****Note: RecurPay customers are identified by the 'customerAccountNumber' field. Subsequent updates and cancels must use the 'customerAccountNumber' field to identify the customer.**

4.4.0 Request to Update a RecurPay Plan for a Customer

The following is an example request that one might post to update the RecurPay customer previously created:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
  <soapenv:Header/>
  <soapenv:Body>
    <proc:processCommand>
<proc:merchantCode>K3L45JL3K45J34KL435JJK45JLK345JLK543LKJ543JLKUII</proc:merchantCode>
      <proc:test>TRUE</proc:test>
      <proc:command>UPDATE_RECURPAY_CUSTOMER</proc:command>
      <proc:customerAccountNumber>123456789012</proc:customerAccountNumber>
      <proc:paymentMethod>CREDITCARD</proc:paymentMethod>
      <proc:paymentSubMethod>MasterCard</proc:paymentSubMethod>
      <proc:creditCardNumber>5454545454545454</proc:creditCardNumber>
      <proc:expireMonth>01</proc:expireMonth>
      <proc:expireYear>2012</proc:expireYear>
      <proc:cvvCode>123</proc:cvvCode>
    </proc:processCommand>
  </soapenv:Body>
</soapenv:Envelope>
```

4.4.1 Response after performing a RecurPay Plan Update

And here is the response one might expect after updating a recurring plan for a customer:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <soapenv:Body>
    <processCommandResponse xmlns="http://processor">
      <processCommandReturn>
        <approvalCode xsi:nil="true"/>
        <command>UPDATE_RECURPAY_CUSTOMER</command>
        <commandResponseCode>1</commandResponseCode>
        <commandResponseText>Successfully Updated RecurPay
Customer.</commandResponseText>
        <customerAccountNumber>123456789012</customerAccountNumber>
        <paymentAmount>19.95</paymentAmount>
        <paymentMethod>CREDITCARD</paymentMethod>
        <paymentResponseCode>1</paymentResponseCode>
        <paymentResponseText>Successfully Updated RecurPay
Customer.</paymentResponseText>
        <paymentTransactionID/>
        <trackingNumber/>
      </processCommandReturn>
    </processCommandResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

4.5.0 Request to Cancel a RecurPay Plan

Here is an example post to cancel a RecurPay customer:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
  <soapenv:Header/>
  <soapenv:Body>
    <proc:processCommand>
<proc:merchantCode>K3L45JL3K45J34KL435JJK45JLK345JLK543LKJ543JLKUII</proc:merchantCode>
      <proc:test>TRUE</proc:test>
      <proc:command>CANCEL_RECURPAY_CUSTOMER</proc:command>
      <proc:customerAccountNumber>123456789012</proc:customerAccountNumber>
    </proc:processCommand>
  </soapenv:Body>
</soapenv:Envelope>
```

4.5.1 Response after performing a RecurPay Plan Cancelation

And here is the response one might expect after canceling a recurring plan for a customer:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <soapenv:Body>
    <processCommandResponse xmlns="http://processor">
      <processCommandReturn>
        <approvalCode xsi:nil="true"/>
        <command>CANCEL_RECURPAY_CUSTOMER</command>
        <commandResponseCode>1</commandResponseCode>
        <commandResponseText>Successfully Canceled RecurPay
Customer</commandResponseText>
        <customerAccountNumber>123456789012</customerAccountNumber>
        <paymentAmount>19.95</paymentAmount>
        <paymentMethod>CREDITCARD</paymentMethod>
        <paymentResponseCode>1</paymentResponseCode>
        <paymentResponseText>Successfully Canceled RecurPay Customer
</paymentResponseText>
        <paymentTransactionID/>
        <trackingNumber/>
      </processCommandReturn>
    </processCommandResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

5.0 Testing

There are different ways to test the API. The first and easiest is to use the demo merchant code found in the example SOAP posts in section 4. For reference, the demo merchant code is:

K3L45JL3K45J34KL435JLK45JLK345JLK543LKJ543JLKUII

A second way to test is, after installing your live production merchant code, to pass the value of 'true' for the field named 'test' (see Request Values table for more information).

For testing approved payments versus declined payments, you must first be in TEST mode.

5.1 ACH Check Test Payments

Creating an APPROVED test check payment:

Use the 'bankAccountNumber' value of 123456

Creating a DECLINED test check payment:

Use any 'bankAccountNumber' other than 123456

** Note: If you encounter the message "bankRoutingNumber does not conform to ABA test", you can use 123456780 as a test bankRoutingNumber as it does successfully conform to the ABA/Mod 10 algorithm.

5.2 CREDITCARD Test Payments

Creating an APPROVED test credit card payment:

Use a credit card number that passes the Luhn/Mod10 check. Good examples are:
4242424242424242 for Visa
5454545454545454 for MasterCard

Creating a DECLINED test credit card payment:

Use a card number that does NOT pass the Luhn/Mod10 check. Good examples are:
4242424242424241 for Visa
5454545454545453 for MasterCard

** Note: Test payments DO NOT get stored in the database and they do not get sent to the processor. For testing the 'QUERY' command, you must have a production token in live mode.

5.3 Testing RecurPay

To test the RecurPay process, it is advisable to first call the CREATE_RECURPAY_CUSTOMER command, with 4242424242424242 creditCardNumber and test=TRUE and then test subsequent updates and cancels on the previously created customer.

During a RecurPay customer creation, the credit card number is validated using the same process that one time payments use during testing. Thus to get an approved created recurpay plan, please use the testing details found above in the one-time credit card testing.

6.0 Integrating With SOAP

SOAP (Simple Object Access Protocol) is a web services standard protocol that is compatible and convenient for all programming languages to access. Thus every modern web enabled programming language has either built in or readily available libraries and example code you can find online for integrating your software with a SOAP webservice. The following is a helpful list of libraries and source code that may help you integrate with SOAP services if they are new to you or you need assistance.

6.1 .NET Languages

Adding a SOAP Web Service client to a Microsoft project using Visual Studio (.NET languages):

<http://msdn.microsoft.com/en-us/library/tydxdyw9.aspx>

MSDN class documentation to serialize a .NET data object into SOAP format (examples given in VB, C#, C++, F# and Jscript):

<http://msdn.microsoft.com/en-us/library/system.runtime.serialization.formatters.soap.soapformatter.aspx>

6.2 Java / Sun

Examples and explanations of building SOAP clients in JAVA:

<http://java.sun.com/developer/technicalArticles/WebServices/SOAP/>

6.3 PHP

Coding SOAP clients using PHP:

<http://php.net/manual/en/book.soap.php>

6.4 PERL

Using SOAP with PERL:

<http://users.skynet.be/pascalbotte/rcx-ws-doc/soaplite.htm>

6.5 Python

How to build SOAP clients in PYTHON:

<http://users.skynet.be/pascalbotte/rcx-ws-doc/python.htm>

6.6 Additional Resources

Additionally, there are several standalone programs that allow to you interface directly with a SOAP API from your desktop and view the SOAP envelope as well as the raw data getting passed back and forth. One that comes in handy when debugging and testing manually against a SOAP web service is called soapUI by Eviware. It's completely free and can be downloaded here:

<http://www.eviware.com/soapUI/soapui-products-overview.html>