API Guide to Integrating and Using the

DevConnect API Web Service

Record of Reviews and Versions

VERSION ID	DATE	AUTHOR	DESCRIPTION
4.0.0	10/15/13	Dev Team	Added Customer Tokenization Service

Disclaimer of Warranties and Limitations of Liabilities

Regal Technologies/E-Complish 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/E-Complish purchase, lease, or license agreement by which the product was acquired, nor increase in any way Regal Technologies/E-Complish's liability to the customer. In no event shall Regal Technologies/E-Complish be liable for incidental or consequential damages because of information contained in this document or any related materials.

Software Notice

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

All Regal Technologies/E-Complish 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/E-Complish's 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, 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

E-Complish, LLC
Postal Box 926
New Market, Maryland 21774-0926 USA
Telephone (301) 865-7570
Toll Free (888) 847-7744
Fax (240) 331-9188
www.e-complish.com

Table of Contents

General Overview

Purpose of the API

The E-Complish/Regal Technologies DevConnect system is a web service designed to run 24 hours a day 7 days a week listening for SOAP requests coming to it. Its purpose is to provide API access to merchants wishing to securely process ACH, Credit Card, Debit Card and Wire transactions all from a single endpoint and simple web service call.

Web Service Location

The web service can be found 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 and minimize the amount of coding the client must do, there is only one 'operation' built into the web service. The operation is called 'processCommand'. Through this method several different commands and values can be sent to perform the required functionality.

Please review the sections below for more information on which types of processes can be executed.

Processing Transactions

Credit Cards

Processing a Credit Card Transaction

Request Fields

The most common and basic function of the API is to process a single one-time credit card payment. The entry of a credit card payment is essentially an 'Authorization' which will later get automatically settled at the end of the business day (timing can be adjusted, please ask your sales representative). To perform a single credit card payment, the following fields are recommended to post in the API request data.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	TRANSACT	32 CHARs	Required
test	Either TRUE or not. Anything that is sent other than TRUE is interpreted as a non-test transaction.	8 CHARs	Optional – flags to the system if this is a test transaction or not
paymentMethod	CREDITCARD	32 CHARs	Required
paymentSubMethod	One of the following: Visa, MasterCard, Amex, Discover, Unknown	16 CHARs	Required
creditCardNumber	The customer's card number	16 CHARs	Required
expireMonth	The customer's card expiration month	2 CHARs	Required
expireYear	The customer's card expiration year	4 CHARs	Required
cvvCode	The cvv2 aka security code from the back of the card	3-4 CHARs	Optional, but highly recommended
creditCardToken	A previously stored tokenization value obtained outside the API process. (For USAePay clients only)	32 CHARs	This can be passed in **instead of** the creditCardNumber, expireMonth and expireYear.
paymentAmount	The amount to charge, ex: 1.01	12 CHARs	Required
serviceFee	An additional service fee to charge if you like, ex: 0.25	12 CHARs	Optional – will be added to the paymentAmount before processing

billFirstName	The customer's first name	25 CHARs	Required
billLastName	The customer's last name	25 CHARs	Required
billCompany	If the customer is a company and not an individual, you can send the company name here	50 CHARs	Optional. This can be used instead of the billFirstName billLastName but at least one or the other is required
billAddress	The customer's house number and street, ex: 123 Test St	50 CHARs	Required
billCity	The customer's city, ex: Springfield	25 CHARs	Required
billState	The two digit state abbreviation	2 CHARs	Required
billZip	The 5-10 digit zip code for the customer	10 CHARs	Optional but highly recommended
billPhone	The customer's 10 digit phone number	10 CHARs	Optional
billEmail	The customer's email address	50 CHARs	Optional

Other informational fields may be sent in addition to the above. Please see Appendix B (Request Fields) for a listing of all fields in the event you would like to send more customer information for tracking and reporting purposes.

Response Fields

The following response fields are what you may expect from an attempt to process a payment.

Field	Value	Type	Notes
command	The original request command this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
paymentAmount	The amount attempted to charge. Ex: 1.01	12 CHARs	This is echoed from the request
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates if a transaction was approved or declined
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 transaction ID as provided by the back end gateway or processor.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
approvalCode	The 5-6 digit approval code returned from the back end gateway or processor.	6 CHARs	This is not unique but is issued and may be used to verify payment with the processor
trackingNumber	The RegalTek unique tracking number for the transaction.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.

Additionally other fields may be returned depending on what was sent in for the request, please see Appendix C (Response Fields) for a listing of all possible return values.

Example SOAP Request

The following is an example raw SOAP request for an attempt to post a credit card payment to the API. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<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>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </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>2016</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:billCity>Springfield</proc:billCity>
         <proc:billState>CA</proc:billState>
         <proc:billZip>90210</proc:billZip>
         <proc:billPhone>444-555-6666</proc:billPhone>
         <proc:billEmail>support@e-complish.com</proc:billEmail>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
```

Testing

To perform a test credit card payment, simply pass in the field 'test' and set the value to 'TRUE' and use one of the following scenarios listed below.

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 and Reporting functionality, you must have a production token in live mode.

Voiding a Credit Card Transaction

Request Fields

After a credit card transaction has been authorized or approved, it remains in that state until the end of day settlement occurs. Until that happens, you have an opportunity to Void the transaction to remove it from the system without it ever actually charging the customer's card. To process a Void on a credit card transaction, the following fields are recommended for the request.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	VOID	32 CHARs	Required
paymentMethod	CREDITCARD	32 CHARs	Required
trackingNumber	A valid tracking number from a previously authorized payment.	16 CHARs	Required

Response Fields

The following response fields are what you may expect from an attempt to void a payment.

Field	Value	Type	Notes
command	The original request command this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
trackingNumber	Echoed from the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates if a void attempt was approved or declined.
paymentResponseText	The 'response text' returned from attempting to void a payment	120 CHARs	The description of the paymentResponseCode. These will likely be 'Transaction voided successfully' or the reason the transaction could not be voided, most often because it already settled.

Example SOAP Request

The following is an example raw SOAP request for an attempt to void a credit card payment. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>VOID</proc:command>
         <proc:paymentMethod>CREDITCARD</proc:paymentMethod>
         <proc:trackingNumber>1332342565264</proc:trackingNumber>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>VOID</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully processed void.
            </commandResponseText>
            <paymentMethod>CREDITCARD</paymentMethod>
            <trackingNumber>1332342565264</trackingNumber>
            <paymentResponseCode>1</paymentResponseCode>
            <paymentResponseText>Transaction voided successfully.
            </paymentResponseText>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Settling Credit Card Transactions

Credit card transactions are automatically batched and settled at the end of the business day. By default the settlement time is 10PM (Local Time), however you may request from your merchant support manager or sales representative to have a custom time that transactions will settle. Transactions may be 'Voided' (i.e. removed from the system without charging the customer) up until the settlement time, after which, the only way to reverse the payment would be to process a 'Refund'.

Refunding a Credit Card Payment

Request Fields

When a payment has settled during the regular automated end of day batch closing that means the payment is now closed to changes and the funds transfer process is now beginning. The payment can no longer be voided and if it becomes necessary to reverse the payment for whatever reason (perhaps a return of merchandise), the payment will need to be refunded. To process a Refund on a credit card payment, the following fields are recommended for the request to the API.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	REFUND	32 CHARs	Required
paymentMethod	CREDITCARD	32 CHARs	Required
trackingNumber	A valid tracking number from a previously settled payment.	16 CHARs	Required

Response Fields

The following response fields are what you may expect from an attempt to refund a credit card.

Field	Value	Type	Notes
command	The original request command this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
trackingNumber	Echoed from the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
paymentResponseCode	The payment 'response code' returned from the	1 NUM	This is the value that indicates if a refund attempt was approved or

	transaction. In general it's 1 for successful, 2 for declined, 3 for error		declined.
paymentResponseText	The 'response text' returned from attempting to refund a payment	120 CHARs	The description of the paymentResponseCode. This will likely be 'Transaction refunded successfully' or the reason the transaction could not be refunded, most often because it has already been refunded, or has not yet settled.

Example SOAP Request

The following is an example raw SOAP request for an attempt to refund a credit card payment. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>REFUND</proc:command>
         <proc:paymentMethod>CREDITCARD</proc:paymentMethod>
         <proc:trackingNumber>1332342565264</proc:trackingNumber>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>REFUND</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully processed refund.
            </commandResponseText>
            <paymentMethod>CREDITCARD</paymentMethod>
            <trackingNumber>1332342565264</trackingNumber>
            <paymentResponseCode>1</paymentResponseCode>
            <paymentResponseText>Transaction refunded successfully.
            </paymentResponseText>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Debit Cards (Pinless)

Processing a Transaction

Request Fields

Debit Cards (Pinless) are very similar to Credit Cards with the slight adjustment to a few fields. This section details how to send a Debit Card transaction to the API. The following fields are the ones recommended to post.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	TRANSACT	32 CHARs	Required
test	Either TRUE or not. Anything that is sent other than TRUE is interpreted as a non-test transaction.	8 CHARs	Optional – flags to the system if this is a test transaction or not
paymentMethod	DEBITCARD	32 CHARs	Required
paymentSubMethod	Should be sent as: Pinless Debit	16 CHARs	Required
debitCardNumber	The customer's card number	16 CHARs	Required
expireMonth	The customer's card expiration month	2 CHARs	Required
expireYear	The customer's card expiration year	4 CHARs	Required
cvvCode	The cvv2 aka security code from the back of the card	3-4 CHARs	Optional, but highly recommended

debitCardToken	A previously stored tokenization value obtained outside the API process. (For USAePay clients only)	32 CHARs	This can be passed in **instead of** the debitCardNumber, expireMonth and expireYear.
paymentAmount	The amount to charge, ex: 1.01	12 CHARs	Required
serviceFee	An additional service fee to charge if you like, ex: 0.25	12 CHARs	Optional – will be added to the paymentAmount before processing
billFirstName	The customer's first name	25 CHARs	Required
billLastName	The customer's last name	25 CHARs	Required
billCompany	If the customer is a company and not an individual, you can send the company name here	50 CHARs	Optional. This can be used instead of the billFirstName billLastName but at least one or the other is required
billAddress	The customer's house number and street, ex: 123 Test St	50 CHARs	Required
billCity	The customer's city, ex: Springfield	25 CHARs	Required
billState	The two digit state abbreviation	2 CHARs	Required
billZip	The 5-10 digit zip code for the customer	10 CHARs	Optional but highly recommended
billPhone	The customer's 10 digit phone number	10 CHARs	Optional
billEmail	The customer's email address	50 CHARs	Optional

Other informational fields may be sent in addition to the above. Please see Appendix B (Request Fields) for a listing of all fields in the event you would like to send more customer information for tracking and reporting purposes.

Response Fields

The following response fields are what you may expect from an attempt to process a debit card.

Field	Value	Type	Notes
command	The original request command this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted

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 during the attempt to process the request 'command'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
paymentAmount	The amount attempted to charge. Ex: 1.01	12 CHARs	This is echoed from the request
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates if a payment was approved or declined
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 transaction ID as provided by the back end gateway or processor.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
approvalCode	The 5-6 digit approval code returned from the back end gateway or processor.	6 CHARs	This is not unique but is issued and may be used to verify payment with the processor
trackingNumber	The RegalTek unique tracking number for the payment.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.

Additionally other fields may be returned depending on what was sent in for the request, please see Appendix C (Response Fields) for a listing of all possible return values.

Example SOAP Request

The following is an example raw SOAP request for an attempt to post a debit card payment to the API. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<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>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>TRANSACT</proc:command>
         <proc:paymentMethod>DEBITCARD</proc:paymentMethod>
         <proc:paymentSubMethod>Pinless Debit</proc:paymentSubMethod>
         <proc:debitCardNumber>4242424242424242</proc:debitCardNumber>
         <proc:expireMonth>01</proc:expireMonth>
         <proc:expireYear>2016</proc:expireYear>
         <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:billCity>Springfield</proc:billCity>
         <proc:billState>CA</proc:billState>
         <proc:billZip>90210</proc:billZip>
         <proc:billPhone>444-555-6666</proc:billPhone>
         <proc:billEmail>support@e-complish.com</proc:billEmail>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>DEBITCARD</paymentMethod>
            <paymentAmount>1.01</paymentAmount>
            <paymentResponseCode>1</paymentResponseCode>
            <paymentResponseText>(TESTMODE) This transaction has been approved.
            </paymentResponseText>
            <paymentTransactionID>0</paymentTransactionID>
            <trackingNumber>1280892202845</trackingNumber>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Testing

To perform a test debit card payment, simply pass in the field 'test' and set the value to 'TRUE' and use one of the following scenarios listed below.

Creating an APPROVED test debit card payment:

Use a debit card number that passes the Luhn/Mod10 check for credit cards. A good example is:

4242424242424242

Creating a DECLINED test debit card payment:

Use a debit card number that does NOT pass the Luhn/Mod10 check. A good example is:

4242424242424241

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

Voiding a Debit Card Transaction

Request Fields

After a transaction has been approved, it remains in that state until the end of day settlement occurs. Until that happens, you have an opportunity to Void the transaction to remove it from the system without it ever actually debiting the customer's card. To process a Void on a debit card transaction, the following fields are recommended for the request.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	VOID	32 CHARs	Required
paymentMethod	DEBITCARD	32 CHARs	Required
trackingNumber	A valid tracking number from a previously authorized payment.	16 CHARs	Required

Response Fields

The following response fields are what you may expect from an attempt to void a payment.

Field	Value	Type	Notes
command	The original request command this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
trackingNumber	Echoed from the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's	1 NUM	This is the value that indicates if a void attempt was approved or declined.

	1 for successful, 2 for declined, 3 for error		
paymentResponseText	The 'response text' returned from attempting to void a payment	120 CHARs	The description of the paymentResponseCode. These will likely be 'Transaction voided successfully' or the reason the transaction could not be voided, most often because it already settled.

Example SOAP Request

The following is an example raw SOAP request for an attempt to void a debit card payment. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>VOID</proc:command>
         <proc:paymentMethod>DEBITCARD</proc:paymentMethod>
         <proc:trackingNumber>1332342565265</proc:trackingNumber>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>VOID</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully processed void.
            </commandResponseText>
            <paymentMethod>DEBITCARD</paymentMethod>
            <trackingNumber>1332342565265</trackingNumber>
            <paymentResponseCode>1</paymentResponseCode>
            <paymentResponseText>Transaction voided successfully.
            </paymentResponseText>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Settling Debit Card Transactions

Debit card transactions, just like credit card transactions, are automatically batched and settled at the end of the business day. By default the settlement time is 10PM (Local Time), however you may request from your merchant support manager or sales representative to have a custom time that payments will settle. Transactions may be 'Voided' (i.e. removed from the system without charging the customer) up until the settlement time, after which, the only way to reverse the payment would be to process a 'Refund'.

Refunding a Debit Card Payment

Request Fields

When a payment has settled during the regular automated end of day batch closing, it means the payment is now closed to changes and the funds transfer process is now beginning. The payment can no longer be voided and if it becomes necessary to reverse the payment for whatever reason (perhaps a return of merchandise), the payment will need to be refunded. To process a Refund on a debit card payment, the following fields are recommended for the request to the API.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	REFUND	32 CHARs	Required
paymentMethod	DEBITCARD	32 CHARs	Required
trackingNumber	A valid tracking number from a previously settled payment.	16 CHARs	Required

Response Fields

The following response fields are what you may expect from an attempt to refund a debit card.

Field	Value	Type	Notes
command	The original request command this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
trackingNumber	Echoed from the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates if a refund attempt was approved or declined.
paymentResponseText	The 'response text' returned from attempting to refund a payment	120 CHARs	The description of the paymentResponseCode. This will likely be 'Transaction refunded successfully' or the reason the transaction could not be refunded, most often because it has already been refunded, or has not yet settled.

Example SOAP Request

The following is an example raw SOAP request for an attempt to refund a debit card payment. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>REFUND</proc:command>
         <proc:paymentMethod>DEBITCARD</proc:paymentMethod>
         <proc:trackingNumber>1332342565264</proc:trackingNumber>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>REFUND</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully processed refund.
            </commandResponseText>
            <paymentMethod>DEBITCARD</paymentMethod>
            <trackingNumber>1332342565264</trackingNumber>
            <paymentResponseCode>1</paymentResponseCode>
            <paymentResponseText>Transaction refunded successfully.
            </paymentResponseText>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

ACH / Online Checks

Processing an ACH Debit Transaction (ACH Debit)

Request Fields

The most common type of ACH payment (aka ACH Debit) is to process a single payment that debits funds from the customer's bank account and credits the merchant's bank account via the 9 digit ABA number and 4-17 digit bank account number of the customer's bank account. Below are the fields recommended to send to the API to process this type of payment.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	TRANSACT	32 CHARs	Required
test	Either TRUE or not. Anything that is sent other than TRUE is interpreted as a non-test transaction.	8 CHARs	Optional – flags to the system if this is a test transaction or not
paymentMethod	ACH	32 CHARs	Required
paymentSubMethod	One of the following: Checking, Savings, Business	16 CHARs	Required
bankRoutingNumber	The 9 digit ABA or Bank Routing Number located on the bottom left side of a paper check	9 CHARs	Required
bankAccountNumber	The customer's Bank Account Number, usually between 4 and 17 digits, located to the right of their Bank Routing Number on their paper checks	17 CHARs	Required
checkDate	The date to process the check in YYYY-MM-DD format	10 CHARs	Optional. If you don't pass this in, or if you

			send a blank value, we will automatically set it to today's date
paymentAmount	The amount to charge, ex: 1.01	12 CHARs	Required
serviceFee	An additional service fee to charge if you like, ex: 0.25	12 CHARs	Optional – will be added to the paymentAmount before processing
billFirstName	The customer's first name	25 CHARs	Required
billLastName	The customer's last name	25 CHARs	Required
billCompany	If the customer is a company and not an individual, you can send the company name here	50 CHARs	Optional. This can be used instead of the billFirstName billLastName but at least one or the other is required
billAddress	The customer's house number and street, ex: 123 Test St	50 CHARs	Required
billCity	The customer's city, ex: Springfield	25 CHARs	Required
billState	The two digit state or province abbreviation	2 CHARs	Required
billZip	The 5-10 digit zip code for the customer	10 CHARs	Required
billPhone	The customer's 10 digit phone number	10 CHARs	Optional
billEmail	The customer's email address	50 CHARs	Optional

Other informational fields may be sent in addition to the above. Please see Appendix B (Request Fields) for a listing of all fields in the event you would like to send more customer information for tracking and reporting purposes.

Response Fields

The following response fields are what you may expect from an attempt to process an ACH Debit payment.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted

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 '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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
paymentAmount	The amount attempted to charge. Ex: 1.01	12 CHARs	This is echoed from the request
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates if a payment was approved or declined
paymentResponseText	The 'response text' returned from attempting the transaction	120 CHARs	The description of the paymentResponseCode. This will likely be 'Transaction Inserted Successfully' or just 'Success'.
paymentTransactionID	The transaction ID as provided by the back end gateway or processor.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
trackingNumber	The RegalTek unique tracking number for the payment.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.

Additionally other fields may be returned depending on what was sent in for the request, please see Appendix C (Response Fields) for a listing of all possible return values.

Example SOAP Request

The following is an example raw SOAP request for an attempt to post an ACH Debit payment to the API. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<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>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </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>2012-12-25</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:billCity>Springfield</proc:billCity>
         <proc:billState>CA</proc:billState>
         <proc:billZip>90210</proc:billZip>
         <proc:billPhone>444-555-6666</proc:billPhone>
         <proc:billEmail>support@e-complish.com</proc:billEmail>
      </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <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>
```

Testing

To perform a test ACH payment, simply pass in the field 'test' and set the value to 'TRUE' and use one of the following scenarios listed below.

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.

Sending an ACH Credit Transaction (ACH Credit)

Request Fields

A second type of ACH transaction can be made which is to debit the funds from the merchant's bank account and send/credit funds to the customer's bank account. This is called an ACH Credit. It is like a refund, but needs no initial transaction to exist. This is sometimes also known as an 'Independent Credit'. Below are the fields recommended to send to the API to process this type of payment.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	TRANSACT	32 CHARs	Required
paymentMethod	ACH	32 CHARs	Required
transactionType	CREDIT	16 CHARs	Required – if this is not present or is blank, then DEBIT is assumed.
test	Either TRUE or not. Anything that is sent other than TRUE is interpreted as a non-test transaction.	8 CHARs	Optional – flags to the system if this is a test transaction or not
paymentSubMethod	One of the following: Checking, Savings, Business	16 CHARs	Required
bankRoutingNumber	The 9 digit ABA or Bank Routing Number located on the bottom left side of a paper check	9 CHARs	Required
bankAccountNumber	The customer's Bank Account Number, usually between 4 and 17	17 CHARs	Required

	digits, located to the right of their Bank Routing Number on their paper checks		
checkDate	The date to process the check in YYYY-MM-DD format	10 CHARs	Optional. If you don't pass this in, or if you send a blank value, we will automatically set it to today's date
paymentAmount	The amount to charge, ex: 1.01	12 CHARs	Required
serviceFee	An additional service fee to charge if you like, ex: 0.25	12 CHARs	Optional – will be added to the paymentAmount before processing
billFirstName	The customer's first name	25 CHARs	Required
billLastName	The customer's last name	25 CHARs	Required
billCompany	If the customer is a company and not an individual, you can send the company name here	50 CHARs	Optional. This can be used instead of the billFirstName billLastName but at least one or the other is required
billAddress	The customer's house number and street, ex: 123 Test St	50 CHARs	Required
billCity	The customer's city, ex: Springfield	25 CHARs	Required
billState	The two digit state or province abbreviation	2 CHARs	Required
billZip	The 5-10 digit zip code for the customer	10 CHARs	Required
billPhone	The customer's 10 digit phone number	10 CHARs	Optional
billEmail	The customer's email address	50 CHARs	Optional

Other informational fields may be sent in addition to the above. Please see Appendix B (Request Fields) for a listing of all fields in the event you would like to send more customer information for tracking and reporting purposes.

Response Fields

The following response fields are what you may expect from an attempt to process an ACH Credit.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 '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'.
transactionType	This is set to CREDIT as per the request post	16 CHARs	If this is not present or blank, it can be assumed that the response refers to a DEBIT transaction
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
paymentAmount	The amount attempted to charge. Ex: 1.01	12 CHARs	This is echoed from the request
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates if a payment was approved or declined
paymentResponseText	The 'response text' returned from attempting the transaction	120 CHARs	The description of the paymentResponseCode. This will likely be 'Transaction Inserted Successfully' or just 'Success'.
paymentTransactionID	The transaction ID as provided by the back end gateway or processor.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
trackingNumber	The RegalTek unique	16 CHARs	This is a unique ID with a 1

	tracking number for the payment.	to 1 relationship per transaction attempt.

Additionally other fields may be returned depending on what was sent in for the request, please see Appendix C (Response Fields) for a listing of all possible return values.

Example SOAP Request

The following is an example raw SOAP request for an attempt to post an ACH Credit to the API. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<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>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>TRANSACT</proc:command>
         <proc:paymentMethod>ACH</proc:paymentMethod>
         <proc:transactionType>CREDIT</proc:transactionType>
         <proc:paymentSubMethod>Checking</proc:paymentSubMethod>
         <proc:bankRoutingNumber>123456780</proc:bankRoutingNumber>
         <proc:bankAccountNumber>123456</proc:bankAccountNumber>
         <proc:checkDate>2012-12-25</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:billCity>Springfield</proc:billCity>
         <proc:billState>CA</proc:billState>
         <proc:billZip>90210</proc:billZip>
         <proc:billPhone>444-555-6666</proc:billPhone>
         <proc:billEmail>support@e-complish.com</proc:billEmail>
      </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>TRANSACT</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully processed transaction.
            </commandResponseText>
            <transactionType>CREDIT</transactionType>
            <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>
```

Testing

To perform a test ACH payment, simply pass in the field 'test' and set the value to 'TRUE' and use one of the following scenarios listed below.

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.

Updating an ACH Transaction

Request Fields

The API provides the ability to modify or update an existing ACH transaction that is in the system, prior to closing/settling the ACH batch. The first four fields below are required and the remaining fields are suggestions as to what you may wish to update about the payment.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	UPDATE	32 CHARs	Required
paymentMethod	ACH	32 CHARs	Required
trackingNumber	The trackingNumber of the previously authorized payment you wish to update.	16 CHARs	Required
* The remaining fields below are all optional and only listed as suggestions of information that you may wish to update about a payment *			
transactionType	Whether this should be a CREDIT or	16 CHARs	Optional

	DEBIT type of transaction		
paymentSubMethod	One of the following: Checking, Savings, Business	16 CHARs	Optional
bankRoutingNumber	The 9 digit ABA or Bank Routing Number located on the bottom left side of a paper check	9 CHARs	Optional
bankAccountNumber	The customer's Bank Account Number, usually between 4 and 17 digits, located to the right of their Bank Routing Number on their paper checks	17 CHARs	Optional
paymentAmount	The amount to charge, ex: 1.01	12 CHARs	Optional
billFirstName	The customer's first name	25 CHARs	Optional
billLastName	The customer's last name	25 CHARs	Optional
billAddress	The customer's house number and street, ex: 123 Test St	50 CHARs	Optional
billCity	The customer's city, ex: Springfield	25 CHARs	Optional
billState	The two digit state or province abbreviation	2 CHARs	Optional
billZip	The 5-10 digit zip code for the customer	10 CHARs	Optional
billPhone	The customer's 10 digit phone number	10 CHARs	Optional
billEmail	The customer's email address	50 CHARs	Optional

Other informational fields may be sent in addition to the above. Please see Appendix B (Request Fields) for a listing of all fields in the event you would like to send more customer information for tracking and reporting purposes.

Response Fields

The following response fields are what you may expect from an attempt to update an ACH payment.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
commandResponseCode	The response to attempting	1 NUM	This is the overall response

	the 'command' 1 for successful, 2 for declined, 3 for error		to the '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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
trackingNumber	The tracking number for the payment as posted in the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
transactionType	If the resulting payment is a CREDIT payment, this will be set to CREDIT	16 CHARs	If this is not present or blank, it can be assumed that the response refers to a DEBIT transaction

Additionally other fields may be returned depending on what was sent in for the request, please see Appendix C (Response Fields) for a listing of all possible return values.

Example SOAP Request

The following is an example raw SOAP request for an attempt to update the customer's name, bank account type, and payment amount for a previously inserted ACH transaction. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>UPDATE</proc:command>
         <proc:paymentMethod>ACH</proc:paymentMethod>
         <proc:trackingNumber>1280893546734</proc:trackingNumber>
         <proc:billFirstName>John</proc:billFirstName>
         <proc:billLastName>Doe</proc:billLastName>
         <proc:paymentSubMethod>Savings</proc:paymentSubMethod>
         <proc:paymentAmount>2.02</proc:paymentAmount>
      </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>UPDATE</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully processed update.
            </commandResponseText>
            <paymentMethod>ACH</paymentMethod>
            <trackingNumber>1280893546734</trackingNumber>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Deleting an ACH Transaction

Request Fields

In the event that you wish to delete or void an ACH transaction before the settlement process occurs, the following fields are recommended for the request.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	VOID	32 CHARs	Required
paymentMethod	ACH	32 CHARs	Required
trackingNumber	A valid tracking number from a previously authorized payment.	16 CHARs	Required

Response Fields

The following response fields are what you may expect from an attempt to void a transaction.

Field	Value	Type	Notes
command	The original request command this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
commandResponseCode	The response to attempting the 'command' 1 for successful, 2 for declined, 3	1 NUM	This is the overall response to the request 'command' field. Do not confuse this with the

	for error		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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
trackingNumber	Echoed from the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates if a void (delete) attempt was approved or declined.
paymentResponseText	The 'response text' returned from attempting to void (delete) a payment	120 CHARs	The description of the paymentResponseCode. These will likely be 'Transaction voided successfully' or the reason the transaction could not be voided, most often because it already settled.

Example SOAP Request

The following is an example raw SOAP request for an attempt to void (delete) an ACH transaction. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>VOID</proc:command>
         <proc:paymentMethod>ACH</proc:paymentMethod>
         <proc:trackingNumber>1332342565265</proc:trackingNumber>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>VOID</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully processed void.
            </commandResponseText>
            <paymentMethod>ACH</paymentMethod>
            <trackingNumber>1332342565265</trackingNumber>
            <paymentResponseCode>1</paymentResponseCode>
            <paymentResponseText>Transaction voided successfully.
            </paymentResponseText>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Settling ACH Transactions

Request Fields

The API optionally provides the ability to invoke a 'settle' command against your own currently open batch of ACH transactions. This is useful for merchants who want to be in control of their own timing and limits for which transactions get sent and processed as a batch. The default approach is to have the merchant manually process their open ACH batches before 5:00PM EST daily using the ACHNow .Net ACH Processor interface. Alternatively, the merchant can use a process called AutoSend that will automatically process any open ACH batch at specified time of day (usually before 5PM EST daily). Please contact your merchant support manager or sales representative to have a custom time that ACH transactions will settle should you choose to use AutoSend.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	SETTLE_BATCH	32 CHARs	Required
paymentMethod	ACH	32 CHARs	Required

Response Fields

The following response fields are what you may expect from an attempt to settle the current ACH batch.

Field	Value	Type	Notes
command	The original request command this response is	32 CHARs	Echoed from request – This is the command that was attempted

	referring to.		
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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing

Example SOAP Request

The following is an example raw SOAP request for an attempt to settle an ACH batch. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>SETTLE_BATCH</proc:command>
         <proc:paymentMethod>ACH</proc:paymentMethod>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>SETTLE_BATCH</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully settled batch.
            </commandResponseText>
            <paymentMethod>ACH</paymentMethod>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Refunding an ACH Payment

Request Fields

When an ACH payment has settled and can no longer be deleted or modified, it is sometimes needed to reverse or refund the payment for whatever reason (perhaps a return of merchandise). To process a Refund on an ACH Debit payment, the following fields should be sent to the API.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	REFUND	32 CHARs	Required
paymentMethod	ACH	32 CHARs	Required
trackingNumber	A valid tracking number from a previously settled payment.	16 CHARs	Required

* Note: For security and good business practice reasons, this method only works on payments that were originally ACH Debits, i.e. situations where the merchant has collected funds from the customer and now wants to refund those funds.

Response Fields

The following response fields are what you may expect from an attempt to refund an ACH Debit payment.

Field	Value	Type	Notes
command	The original request command this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
trackingNumber	Echoed from the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's	1 NUM	This is the value that indicates if a refund attempt was approved or

	1 for successful, 2 for declined, 3 for error		declined.
paymentResponseText	The 'response text' returned from attempting to refund a payment	120 CHARs	The description of the paymentResponseCode. This will likely be 'Transaction refunded successfully' or the reason the transaction could not be refunded, most often because it has already been refunded, or has not yet settled.

Example SOAP Request

The following is an example raw SOAP request for an attempt to refund an ACH Debit payment. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>REFUND</proc:command>
         <proc:paymentMethod>ACH</proc:paymentMethod>
         <proc:trackingNumber>1332342565264</proc:trackingNumber>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>REFUND</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully processed refund.
            </commandResponseText>
            <paymentMethod>ACH</paymentMethod>
            <trackingNumber>1332342565264</trackingNumber>
            <paymentResponseCode>1</paymentResponseCode>
            <paymentResponseText>Transaction refunded successfully.
            </paymentResponseText>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Wire Transfers

Posting a Wire Transfer

The API provides the ability to send an electronic bank-to-bank Wire Transfer. This is similar to the ACH functions mentioned above in that the destination Bank Routing Number and Bank Account Number are required, but has an additional benefit of transferring funds directly in real-time instead of going through the ACH settlement process and potentially longer waiting cycle.

Request Fields

To post a Wire Transfer, the following request fields should be sent to the API.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	TRANSACT	32 CHARs	Required
paymentMethod	WIRE	32 CHARs	Required
paymentAmount	The amount of funds to transfer in dollars and cents, ex: 1.01	12 CHARs	Required
serviceFee	An additional service fee to charge if you like, ex: 0.25	12 CHARs	Optional – will be added to the paymentAmount before processing
bankRoutingNumber	The receivers bank 9 digit ABA or Bank Routing Number.	9 CHARs	Required
bankAccountNumber	The receivers Bank Account Number, between 4 and 17 digits.	17 CHARs	Required
billFirstName	The receivers first name	25 CHARs	Required
billLastName	The receivers last name	25 CHARs	Required
billAddress	The receivers street address which should match the address on file at the receivers bank account	35 CHARs	Required
billCity	The receivers city	35 CHARs	Required
billState	The two digit state or province abbreviation	2 CHARs	Required
billZip	The 5-10 digit zip code for the customer	10 CHARs	Required

billPhone	The customer's 10 digit phone number	10 CHARs	Optional but recommended
description	An optional field to provide information to the receiver from the sender. Ex: "Monthly Payment for June"	120 CHARs	Optional but recommended

Other informational fields may be sent in addition to the above. Please see Appendix B (Request Fields) for a listing of all fields in the event you would like to send more customer information for tracking and reporting purposes.

Response Fields

The following response fields are what you may expect from an attempt to process a Wire transfer.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 '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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
paymentAmount	The amount attempted to charge. Ex: 1.01	12 CHARs	This is echoed from the request
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates if the wire request was properly formatted and is being submitted to the Fed
paymentResponseText	The 'response text' returned from attempting the transaction	120 CHARs	The description of the paymentResponseCode. This will likely be 'Transaction Inserted

			Successfully' or just 'Success'.
trackingNumber	The RegalTek unique tracking number for the payment.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.

Additionally other fields may be returned depending on what was sent in for the request, please see Appendix C (Response Fields) for a listing of all possible return values.

Example SOAP Request

The following is an example raw SOAP request for an attempt to post a Wire Transfer to the API. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>TRANSACT</proc:command>
         <proc:paymentMethod>WIRE</proc:paymentMethod>
         <proc:paymentAmount>1.01</proc:paymentAmount>
         <proc:bankRoutingNumber>123456780</proc:bankRoutingNumber>
         <proc:bankAccountNumber>123456</proc:bankAccountNumber>
         <proc:billFirstName>Mary</proc:billFirstName>
         <proc:billLastName>Smith</proc:billLastName>
         <proc:billAddress>123 test st</proc:billAddress>
         <proc:billCity>Springfield</proc:billCity>
         <proc:billState>CA</proc:billState>
         <proc:billZip>90210</proc:billZip>
         <proc:billPhone>444-555-6666</proc:billPhone>
         <proc:description>Payment for November Services</proc:description>
      </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>TRANSACT</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully processed transaction.</commandResponseText>
            <paymentMethod>WIRE</paymentMethod>
            <paymentAmount>1.01</paymentAmount>
            <paymentResponseCode>1</paymentResponseCode>
            <paymentResponseText>Transaction Inserted Successfully.</paymentResponseText>
            <trackingNumber>1280893546734</trackingNumber>
```

```
        </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Settling Wire Transfers

Wire transfers are batched and sent to the Federal Reserve for approval on an hourly basis. There is no 'end of day' settlement process or manual settlement process that occurs. For this reason, merchants should be sure that they want the wire transfer to occur (submitted to the Fed for approval) because once it's submitted, there is no ability to void or cancel the transfer. For a more flexible (but slower) process, the ACH functions should be used.

Scheduling Transactions

Future Transactions

Inserting a Future Transaction

The API also has the ability to hold on to a transaction for a specified period of time before posting it to the appropriate processor for processing. This type of transaction we refer to as a FuturePay transaction, and it works for all payment methods listed above (CREDITCARD, DEBITCARD, ACH and WIRE).

To specify that you would like to insert a transaction for processing at a later date, simply send a field named 'paymentDate' in the format of YYYY-MM-DD along with the other fields for that transaction. If the paymentDate is received by the system and that date is greater than today, the system will automatically recognize that this should be marked as a Future Transaction.

Bear in mind however that you will not get a definitive answer of whether the transaction is approved or declined because the transaction will not have been sent to the processor yet. The system does, however, verify the incoming data to make sure the card numbers, bank numbers, etc., conform to the standard numbering algorithms and verify as much as we can that the transaction is a good candidate for approval.

In addition, for credit card and pin-less debit card payments, we will run a 1 cent pre-authorization against the card to verify that the card number, expiration date and CVV code are actually correct before inserting the FuturePay transaction. The 1 cent pre-authorization drops off at the end of the business day and is never actually charged nor reflected on the customer's card statement.

Below is example SOAP request and response for sending in a single one-time Credit Card payment to be processed on a future date.

Request Fields

For Request Fields needed for Future transactions, please see the appropriate previous section that matches the which type of transaction you would like to submit, then make sure to add the 'paymentDate' field to the request.

Response Fields

For Response Fields returned for Future transactions, please see the appropriate previous section that matches the which type of transaction you are submitting, however, please note that the response will not be indicative of whether the funds will actually process or not. It will only indicate if the transaction has been successfully scheduled for processing.

To follow up later and get the response value from the processor, you will need to use the online reporting tools, or Query functionality of the API as described in the Querying and Reporting section of this document.

Example SOAP Request for a Credit Card Future Payment

Here is an example SOAP request to insert a future dated credit card payment to illustrate how it works.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
```

```
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>TRANSACT</proc:command>
         <proc:paymentMethod>CREDITCARD</proc:paymentMethod>
         <proc:paymentDate>2014-01-01</proc:paymentDate>
         <proc:paymentSubMethod>Visa</proc:paymentSubMethod>
         <proc:creditCardNumber>4242424242424242</proc:creditCardNumber>
         <proc:expireMonth>01</proc:expireMonth>
         <proc:expireYear>2016</proc:expireYear>
         <proc:paymentAmount>1.01</proc:paymentAmount>
         <proc:serviceFee>0.25</proc:serviceFee>
         <proc:billFirstName>test</proc:billFirstName>
         <proc:billLastName>test</proc:billLastName>
         <proc:billAddress>123 test st</proc:billAddress>
         <proc:billCity>Springfield</proc:billCity>
         <proc:billState>CA</proc:billState>
         <proc:billZip>90210</proc:billZip>
         <proc:billPhone>444-555-6666</proc:billPhone>
         <proc:billEmail>support@e-complish.com</proc:billEmail>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>TRANSACT</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully processed transaction.
            </commandResponseText>
            <paymentMethod>CREDITCARD</paymentMethod>
            <paymentAmount>1.01</paymentAmount>
            <paymentResponseCode>1</paymentResponseCode>
            <paymentResponseText>Future payment inserted successfully.
</paymentResponseText>
            <trackingNumber>1280892202844</trackingNumber>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Updating a Future Transaction

To modify / update a Future transaction, you will need to pass in the standard required authentication values along with the trackingNumber and fields you would like updated.

Request Fields

The first four fields below are required and the remaining fields are suggestions as to what you may wish to update about the transaction. The fields you choose are likely going to vary based on the type of Future transaction you are modifying.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	UPDATE	32 CHARs	Required
paymentMethod	One of ACH, CREDITCARD, DEBITCARD or WIRE	32 CHARs	Required
trackingNumber	The trackingNumber of the previously inserted future payment.	16 CHARs	Required
*** The remaining fields below are all optional and only listed as suggestions of information that you may wish to update about a payment ***			
paymentDate	The date to process this payment in YYYY-MM-DD format	10 CHARs	Optional
paymentAmount	The amount to charge, ex: 1.01	12 CHARs	Optional
billFirstName	The customer's first name	25 CHARs	Optional
billLastName	The customer's last name	25 CHARs	Optional

Other informational fields may be sent in addition to the above. Please see Appendix B (Request Fields) for a listing of all fields in the event you would like to send more customer information for tracking and reporting purposes.

Response Fields

The following response fields are what you may expect from an attempt to update a Future transaction.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 '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’.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
trackingNumber	The tracking number for the payment as posted in the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.

Additionally other fields may be returned depending on what was sent in for the request, please see Appendix C (Response Fields) for a listing of all possible return values.

Example SOAP Request

The following is an example raw SOAP request for an attempt to update the payment date and amount of a previously scheduled FuturePay ACH payment.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>UPDATE</proc:command>
         <proc:paymentMethod>ACH</proc:paymentMethod>
         <proc:trackingNumber>1280893546734</proc:trackingNumber>
         <proc:paymentDate>2014-02-02</proc:paymentDate>
         <proc:paymentAmount>2.02</proc:paymentAmount>
      </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>UPDATE</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Future payment successfully updated.
            </commandResponseText>
            <paymentMethod>ACH</paymentMethod>
            <trackingNumber>1280893546734</trackingNumber>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

* Note: The system will automatically know if you are trying to update a future transaction, or for example, trying to modify an existing ACH authorization that hasn't settled yet, even though the command and paymentMethod would be the same for both of these operations. We can tell which you are trying to do based on the current status of the transaction in our system.

Cancelling a Future Transaction

Request Fields

In the event that you wish to cancel (or void) a previously inserted Future transaction before processing occurs, the following fields should be sent in the request data.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	VOID	32 CHARs	Required
paymentMethod	One of ACH, CREDITCARD, DEBITCARD or WIRE	32 CHARs	Required
trackingNumber	A valid tracking number from a previously scheduled payment.	16 CHARs	Required

Response Fields

The following response fields are what you may expect from an attempt to cancel a future transaction.

Field	Value	Type	Notes
command	The original request command this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted

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'.
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing
trackingNumber	Echoed from the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.

Example SOAP Request

The following is an example raw SOAP request for an attempt to cancel / void a future payment. The following is only for illustration and troubleshooting purposes only.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>VOID</proc:command>
         <proc:paymentMethod>ACH</proc:paymentMethod>
         <proc:trackingNumber>1332342565265</proc:trackingNumber>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>VOID</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully cancelled future payment.
            </commandResponseText>
            <paymentMethod>ACH</paymentMethod>
            <trackingNumber>1332342565265</trackingNumber>
         </processCommandReturn>
      </processCommandResponse>
```

```
   </soapenv:Body>
</soapenv:Envelope>
```

Recurring Payment Plans

In addition to real-time single payments and future single payments, the system has the ability to create recurring payment plans (that we call RecurPay) that occur on a frequency and duration that you specify.

Creating a RecurPay Plan

Request Fields

Listed below are the required and recommended fields that should be sent to the API for creating a RecurPay plan for a customer.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	CREATE_RECURPAY_CUSTOMER	32 CHARs	Required
test	Either TRUE or not. Anything that is sent other than TRUE is interpreted as a non-test transaction.	8 CHARs	Optional – flags to the system if this is a test transaction or not
customerAccountNumber	The merchant generated unique customer account number to identify this RecurPay customer	32 CHARs	Required – this is used to identify the customer throughout the system and in UPDATE and CANCEL commands
recurringStartDate	The date to begin or activate this recurring payment plan. Format should be MM/DD/YYYY	10 CHARs	Required
recurringFrequency	One of M (for monthly), T (for twice a month), B (for bi-weekly) or W (for weekly)	1 CHAR	Required
recurringDateOfMonth1	A month day in the form of DD	2 CHARs	Required for frequencies of M or T
recurringDateOfMonth2	A second month day in the form of DD	2 CHARs	Required for frequency of T
recurringWeekDay	A week day abbreviation, one of	2 CHARs	Required for frequencies

	MO, TU, WE, TH, FR, SA, SU		of B or W
recurringAmount	The amount to bill on the recurringFrequency specified	12 CHARs	Required
numberOfPayments	The number of payments remaining in this payment plan until the plan is completed	6 CHARs	Required – Note this automatically counts down each time a payment is attempted
paymentMethod	CREDITCARD, DEBITCARD or ACH	32 CHARs	Required
paymentSubMethod	One of the following: Visa, MasterCard, Amex, Discover, Pinless Debit, Checking, Savings, Business, Unknown	16 CHARs	Required
creditCardNumber	The customer's credit card number	16 CHARs	Required for paymentMethod = CREDITCARD
debitCardNumber	The customer's debit card number	16 CHARs	Required for paymentMethod = DEBITCARD
expireMonth	The customer's card expiration month	2 CHARs	Required for paymentMethod = CREDITCARD or DEBITCARD
expireYear	The customer's card expiration year	4 CHARs	Required for paymentMethod = CREDITCARD or DEBITCARD
cvvCode	The cvv2 also known as security code from the back of the card	3-4 CHARs	Optional, but highly recommended. Used for CREDITCARD paymentMethod.
bankRoutingNumber	The 9 digit Bank Routing Number in the case of ACH payments	12 CHARs	Required
bankAccountNumber	The 4-17 digit Bank Account Number in the case of ACH payments	17 CHARs	Optional – will be added to the paymentAmount before processing
billFirstName	The customer's first name	25 CHARs	Required
billLastName	The customer's last name	25 CHARs	Required
billCompany	If the customer is a company and not an individual, you can send the	50 CHARs	Optional. This can be used instead of the

	company name here		billFirstName billLastName but at least one or the other is required
billAddress	The customer's house number and street, ex: 123 Test St	50 CHARs	Required
billCity	The customer's city, ex: Springfield	25 CHARs	Required
billState	The two digit state abbreviation	2 CHARs	Required
billZip	The 5-10 digit zip code for the customer	10 CHARs	Optional but highly recommended
billPhone	The customer's 10 digit phone number	10 CHARs	Optional
billEmail	The customer's email address	50 CHARs	Optional

Response Fields

The following response fields are what you may expect from an attempt to create a RecurPay plan.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 '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'.
customerAccountNumber	Echoed from the original request	32 CHARs	This is the unique identifier for this RecurPay customer. The merchant sends this in during the request to create the plan
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing

paymentAmount	The recurring amount of the payment plan	12 CHARs	This is echoed from the recurringAmount sent in with the request fields

Example SOAP Request

The following illustrates an example SOAP request to create a recurring credit card payment plan in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:test>TRUE</proc:test>
         <proc:command>CREATE_RECURPAY_CUSTOMER</proc:command>
         <proc:customerAccountNumber>123456789012</proc:customerAccountNumber>
         <proc:recurringStartDate>01/17/2013</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>2014</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>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>CREATE_RECURPAY_CUSTOMER</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully Created RecurPay Customer.
            </commandResponseText>
            <customerAccountNumber>123456789012</customerAccountNumber>
            <paymentAmount>19.95</paymentAmount>
            <paymentMethod>CREDITCARD</paymentMethod>
         </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.

Updating a RecurPay Plan

Request Fields

Listed below are the required and recommended fields that should be sent to the API for updating a RecurPay plan for a customer.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	UPDATE_RECURPAY_CUSTOMER	32 CHARs	Required
test	Either TRUE or not. Anything that is sent other than TRUE is interpreted as a non-test transaction.	8 CHARs	Optional – flags to the system if this is a test transaction or not
customerAccountNumber	The merchant generated unique customer account number to identify this RecurPay customer	32 CHARs	Required – this is used to identify the customer. It was previously sent during a CREATE RecurPay call.
* The fields below are all optional and just an example of which fields you may wish to update.			
paymentMethod	CREDITCARD (for example) Can also be DEBITCARD or ACH	32 CHARs	Optional in this situation
paymentSubMethod	One of the following: Visa, MasterCard, Amex, Discover, Pinless Debit, Checking, Savings, Business, Unknown	16 CHARs	Optional in this situation
creditCardNumber	The customer's new credit card number	16 CHARs	Optional in this situation
expireMonth	The customer's new card expiration month	2 CHARs	Optional in this situation
expireYear	The customer's new card expiration year	4 CHARs	Optional in this situation
cvvCode	The new cvv2 also known as security code from the back of the card	4 CHARs	Optional in this situation

Response Fields

The following response fields are what you may expect from an attempt to UPDATE a RecurPay plan.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 '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'.
customerAccountNumber	Echoed from the original request	32 CHARs	This is the unique identifier for this RecurPay customer. The merchant sends this in during the request to create the plan
paymentMethod	An echo of the paymentMethod sent in the request originally when the plan was created	32 CHARs	This varies by type of operation you are processing
paymentAmount	The recurring amount of the payment plan	12 CHARs	This is echoed from the recurringAmount sent in originally when the plan was created.

Example SOAP Request

The following shows an example SOAP request to update a RecurPay customer's credit card in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </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>2015</proc:expireYear>
         <proc:cvvCode>123</proc:cvvCode>
      </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>UPDATE_RECURPAY_CUSTOMER</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully Updated RecurPay Customer.
            </commandResponseText>
            <customerAccountNumber>123456789012</customerAccountNumber>
            <paymentAmount>19.95</paymentAmount>
            <paymentMethod>CREDITCARD</paymentMethod>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Cancelling a RecurPay Plan

Request Fields

Listed below are the fields that should be sent to the API for cancelling a RecurPay plan for a customer.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	CANCEL_RECURPAY_CUSTOMER	32 CHARs	Required
test	Either TRUE or not. Anything that is sent other than TRUE is interpreted as a non-test transaction.	8 CHARs	Optional – flags to the system if this is a test transaction or not
customerAccountNumber	The merchant generated unique customer account number to identify this RecurPay customer	32 CHARs	Required – this is used to identify the customer. It was previously sent during a CREATE RecurPay call.

Response Fields

The following response fields are what you may expect from an attempt to CANCEL a RecurPay plan.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 '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'.
customerAccountNumber	Echoed from the original request	32 CHARs	This is the unique identifier for this RecurPay customer. The merchant sends this in during the request to create the plan
paymentMethod	An echo of the paymentMethod sent in the request originally when the plan was created	32 CHARs	This varies by type of operation you are processing
paymentAmount	The recurring amount of the payment plan	12 CHARs	This is echoed from the recurringAmount sent in originally when the plan was created.

Example SOAP Request

The following shows an example SOAP request to update a RecurPay customer's credit card in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
<soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </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>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>CANCEL_RECURPAY_CUSTOMER</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully Canceled RecurPay Customer</commandResponseText>
            <customerAccountNumber>123456789012</customerAccountNumber>
            <paymentAmount>19.95</paymentAmount>
            <paymentMethod>CREDITCARD</paymentMethod>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

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 in the one-time credit card testing section.

Customer Tokenization

In addition to recurring and future dated payments, the DevConnect API also supports the use of Customer Tokenization. This means that you will be able to post us the payment information for a customer, we will securely store the information, and you can then process payments on demand as you see fit by simply posting us a 'token' or an identifier representing that customer and by using the TRANSACT command from section 1.

Storing (or Tokenizing) a Customer

Request Fields

Listed below are the required and optional fields that should be sent to the API for creating a customer payment record in our system.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	CREATE_TOKENIZED_CUSTOMER	32 CHARs	Required
test	Either TRUE or not. Anything other than TRUE is interpreted as a production item.	8 CHARs	Optional – flags to the system if this is a test transaction or not
customerAccountNumber	The merchant generated unique customer account number to identify this customer. **** This will be the 'token' or 'identifier' that you pass to us later when processing ****	32 CHARs	Required – this is used to identify the customer throughout the system and in UPDATE and CANCEL commands
paymentMethod	CREDITCARD, DEBITCARD or ACH	32 CHARs	Required
paymentSubMethod	One of the following: Visa, MasterCard, Amex, Discover, Pinless Debit, Checking, Savings, Business, Unknown	16 CHARs	Required
creditCardNumber	The customer's credit card number	16 CHARs	Required for paymentMethod = CREDITCARD
debitCardNumber	The customer's debit card number	16 CHARs	Required for paymentMethod = DEBITCARD
expireMonth	The customer's card expiration	2 CHARs	Required for

	month		paymentMethod = CREDITCARD or DEBITCARD
expireYear	The customer's card expiration year	4 CHARs	Required for paymentMethod = CREDITCARD or DEBITCARD
cvvCode	The cvv2 also known as security code from the back of the card	3-4 CHARs	Optional, but highly recommended. Used for CREDITCARD paymentMethod.
bankRoutingNumber	The 9 digit Bank Routing Number in the case of ACH payments	12 CHARs	Required
bankAccountNumber	The 4-17 digit Bank Account Number in the case of ACH payments	17 CHARs	Optional – will be added to the paymentAmount before processing
billFirstName	The customer's first name	25 CHARs	Required
billLastName	The customer's last name	25 CHARs	Required
billCompany	If the customer is a company and not an individual, you can send the company name here	50 CHARs	Optional. This can be used instead of the billFirstName billLastName but at least one or the other is required
billAddress	The customer's house number and street, ex: 123 Test St	50 CHARs	Required
billCity	The customer's city, ex: Springfield	25 CHARs	Required
billState	The two digit state abbreviation	2 CHARs	Required
billZip	The 5-10 digit zip code for the customer	10 CHARs	Optional but highly recommended
billPhone	The customer's 10 digit phone number	10 CHARs	Optional
billEmail	The customer's email address	50 CHARs	Optional

Response Fields

The following response fields are what you may expect from an attempt to create a tokenized customer.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 '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'.
customerAccountNumber	Echoed from the original request	32 CHARs	This is the unique identifier for this customer. The merchant sends this in during the request to create the customer
paymentMethod	An echo of the paymentMethod sent in the request	32 CHARs	This varies by type of operation you are processing

Example SOAP Request

The following illustrates an example SOAP request to store a tokenized customer in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:test>TRUE</proc:test>
         <proc:command>CREATE_TOKENIZED_CUSTOMER</proc:command>
         <proc:customerAccountNumber>123456789012</proc:customerAccountNumber>
         <proc:paymentMethod>CREDITCARD</proc:paymentMethod>
         <proc:paymentSubMethod>Visa</proc:paymentSubMethod>
         <proc:creditCardNumber>4242424242424242</proc:creditCardNumber>
         <proc:expireMonth>01</proc:expireMonth>
         <proc:expireYear>2014</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>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>CREATE_TOKENIZED_CUSTOMER</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully Created Tokenized Customer.
            </commandResponseText>
            <customerAccountNumber>123456789012</customerAccountNumber>
            <paymentMethod>CREDITCARD</paymentMethod>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

*Note: Customers are identified by the 'customerAccountNumber' field. Subsequent transaction attempts, customer updates and customer cancels must use the 'customerAccountNumber' field to identify the customer.

Updating a Tokenized Customer

Request Fields

Listed below are the required and recommended fields that should be sent to the API for updating a customer.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	UPDATE_TOKENIZED_CUSTOMER	32 CHARs	Required
test	Either TRUE or not. Anything that is sent other than TRUE is interpreted as a non-test transaction.	8 CHARs	Optional – flags to the system if this is a test transaction or not
customerAccountNumber	The merchant generated unique customer account number to identify this customer	32 CHARs	Required – this is used to identify the customer. It was previously sent during a CREATE call.
* The fields below are all optional and just an example of which fields you may wish to update.			

paymentMethod	CREDITCARD (for example) Can also be DEBITCARD or ACH	32 CHARs	Optional in this situation
paymentSubMethod	One of the following: Visa, MasterCard, Amex, Discover, Pinless Debit, Checking, Savings, Business, Unknown	16 CHARs	Optional in this situation
creditCardNumber	The customer's new credit card number	16 CHARs	Optional in this situation
expireMonth	The customer's new card expiration month	2 CHARs	Optional in this situation
expireYear	The customer's new card expiration year	4 CHARs	Optional in this situation
cvvCode	The new cvv2 also known as security code from the back of the card	4 CHARs	Optional in this situation

Response Fields

The following response fields are what you may expect from an attempt to UPDATE a Customer.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 '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'.
customerAccountNumber	Echoed from the original request	32 CHARs	This is the unique identifier for this customer. The merchant sends this in during the original request to create the customer record
paymentMethod	An echo of the paymentMethod sent in	32 CHARs	This varies by type of operation you are

	the request originally when the record was created	processing

Example SOAP Request

The following shows an example SOAP request to update a customer's credit card in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
   <soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:test>TRUE</proc:test>
         <proc:command>UPDATE_TOKENIZED_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>2015</proc:expireYear>
         <proc:cvvCode>123</proc:cvvCode>
      </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>UPDATE_TOKENIZED_CUSTOMER</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully Updated Tokenized Customer.
            </commandResponseText>
            <customerAccountNumber>123456789012</customerAccountNumber>
            <paymentMethod>CREDITCARD</paymentMethod>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Cancelling a Tokenized Customer

Request Fields

Listed below are the fields that should be sent to the API for cancelling or removing a customer from storage.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	CANCEL_TOKENIZED_CUSTOMER	32 CHARs	Required
test	Either TRUE or not. Anything that is sent other than TRUE is interpreted as a non-test transaction.	8 CHARs	Optional – flags to the system if this is a test transaction or not
customerAccountNumber	The merchant generated unique customer account number to identify this customer	32 CHARs	Required – this is used to identify the customer. It was previously sent during a CREATE call.

Response Fields

The following response fields are what you may expect from an attempt to cancel a customer.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 'command' field. Do not confuse this with 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'.
customerAccountNumber	Echoed from the original request	32 CHARs	This is the unique identifier for this customer. The merchant sends this in during the original request to create the customer record
paymentMethod	An echo of the paymentMethod sent in the request originally when the record was created	32 CHARs	This varies by type of operation you are processing

Example SOAP Request

The following shows an example SOAP request to cancel or remove a customer's record from the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
<soapenv:Header/>
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:test>TRUE</proc:test>
         <proc:command>CANCEL_TOKENIZED_CUSTOMER</proc:command>
         <proc:customerAccountNumber>123456789012</proc:customerAccountNumber>
      </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>CANCEL_TOKENIZED_CUSTOMER</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully Canceled Tokenized Customer.
            </commandResponseText>
            <customerAccountNumber>123456789012</customerAccountNumber>
            <paymentMethod>CREDITCARD</paymentMethod>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Performing Transactions on Stored Customers

Request Fields

Listed below are the fields that should be sent to the API for performing a transaction on a previously stored (and thus tokenized) Customer. This is very similar to the regular TRANSACT command for regular customers, except that you need to pass the customerAccountNumber and you do not need to pass any of the customers payment information.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required

command	TRANSACT	32 CHARs	Required
test	Either TRUE or not. Anything that is sent other than TRUE is interpreted as a non-test transaction.	8 CHARs	Optional – flags to the system if this is a test transaction or not
customerAccountNumber	The previously stored customer's account number. This will allow us to pull up and use all the other details needed to perform a transaction.	32 CHARs	Required – this is used to identify the customer. It was previously sent during a CREATE call.
useTokenization	This indicates that you intend to do a tokenization transaction. This must be set to TRUE.	6 CHARs	Required and must be set to TRUE.
paymentAmount	The amount to charge, ex: 1.01	12 CHARs	Required
serviceFee	An additional service fee to charge if you like, ex: 0.25	12 CHARs	Optional – will be added to the paymentAmount before processing

Response Fields

The following response fields are what you may expect from an attempt to process a payment on a previously stored customer.

Field	Value	Type	Notes
command	The original request command this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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'.
paymentMethod	An echo of the paymentMethod that was used for the	32 CHARs	This depends on what was previously stored in the customer record

	payment		
paymentAmount	The amount attempted to charge. Ex: 1.01	12 CHARs	This is echoed from the request
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates if a transaction was approved or declined
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 transaction ID as provided by the back end gateway or processor.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
approvalCode	The 5-6 digit approval code returned from the back end gateway or processor.	6 CHARs	This is not unique but is issued and may be used to verify payment with the processor
trackingNumber	The RegalTek unique tracking number for the transaction.	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.

Example SOAP Request

The following is an example raw SOAP request for an attempt to perform a credit card payment on a previously stored customer. Please use the built in or third party SOAP libraries for your programming language to post payments. The following is only for illustration and troubleshooting purposes only.

```
<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>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>TRANSACT</proc:command>
         <proc:customerAccountNumber>123456789012</proc:customerAccountNumber>
         <proc:useTokenization>TRUE</proc:useTokenization>
         <proc:paymentAmount>1.01</proc:paymentAmount>
         <proc:serviceFee></proc:serviceFee>
       </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
```

Testing Tokenization

To test the Customer Tokenization process, it is necessary to first call the CREATE_TOKENIZED_CUSTOMER command, with the 4242424242424242 creditCardNumber and test=TRUE and then test subsequent transactions, updates and cancels on the previously created customer.

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

Querying and Reporting

Querying Transactions

The API provides the ability for a merchant to query for previously posted transactions based on the unique trackingNumber or on any of the other request fields.

Retrieving a Single Transaction

Request Fields

To retrieve a single status update of a single transaction, the most straightforward query is to do a lookup by trackingNumber. Listed below are the fields that should be sent to the API for this type of query.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	QUERY	32 CHARs	Required
trackingNumber	The unique number for this transaction, generated by Regal and previously returned to the client.	16 CHARs	Required for this type of query

Response Fields

The following response fields are what you may expect from a request to query a single transaction.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 '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'

paymentMethod	An echo of the paymentMethod sent in the request originally when the plan was created	32 CHARs	This varies by type of operation you are processing
customerAccountNumber	The customer account number for this payment	32 CHARs	This is posted by the merchant at the time the payment is created
approvalCode	The 5-6 digit processor approval code for credit card or debit card payments	6 CHARs	This is returned by the processor for approved payments
trackingNumber	Echoed from the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates whether the last payment action was successful or not
paymentResponseText	The current 'response text' for the payment which describes the response code	120 CHARs	The description of the paymentResponseCode. This will be a clearer description of the state of the payment
paymentTransactionID	The processor transaction id if present	16 CHARs	Issued on approvals and some types of declines as well
paymentAmount	The amount of the payment	12 CHARs	This was submitted during the original payment post to us
serviceFee	The service fee of the payment	12 CHARs	This was submitted during the original payment post to us
status	This can be 'Authorized' 'FailedProcessing', 'Retrying', 'Paid', 'Refunded', 'Voided', 'Cancelled', 'Future', 'Declined', 'Returned', 'Unknown'	16 CHARs	A single word describing the 'status' of the payment in the system. This is useful to see if a payment is being queued for re-processing due to a processor failure (Retrying), or to see if a payment has been settled (Paid) vs. unsettled (Authorized).

Example SOAP Request

The following shows an example SOAP request to query for a single transaction in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
<soapenv:Header/>
   <soapenv:Body>
```

```
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>QUERY</proc:command>
         <proc:trackingNumber>1280892202845</proc:trackingNumber>
      </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above.

```
<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>
            <command>QUERY</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully queried transaction.</commandResponseText>
            <paymentMethod>CREDITCARD</paymentMethod>
            <customerAccountNumber>555544433</customerAccountNumber>
            <approvalCode>000000</approvalCode>
            <trackingNumber>1280892202845</trackingNumber>
            <paymentResponseCode>1</paymentResponseCode>
            <paymentResponseText>(TESTMODE) This transaction has been approved.
            </paymentResponseText>
            <paymentTransactionID>0</paymentTransactionID>
            <paymentAmount>1.01</paymentAmount>
            <serviceFee>0.00</serviceFee>
            <status>Authorized</status>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Retrieving a Group of Transactions

Request Fields

To retrieve a group of transactions, such as all payments for a specific day, or all declined debit cards for the week, there please use the following fields.

Field	Value	Type	Notes
merchantCode	Assigned to you by Regal	49 CHARs	Required
command	QUERY	32 CHARs	Required
* Note the remaining fields below are all optional based on what type of payments you are searching for.			
fromDate	A date/time in the format of YYYY-	19 CHARs	Optional – this is the From date

	MM-DD hh:mm:ss		for the query
toDate	A date/time in the format of YYYY-MM-DD hh:mm:ss	19 CHARs	Optional – this is the To date for the query
trackingNumber	The unique number for this transaction, generated by Regal and previously returned to the client.	16 CHARs	Optional
paymentMethod	Can be one of 'CREDITCARD', 'DEBITCARD', 'ACH', 'WIRE'	16 CHARs	Optional
paymentSubMethod	Can be one of 'Checking', 'Savings', 'Business', 'Pinless Debit', 'Visa', 'MasterCard', 'Amex', 'Discover', 'Unknown'	16 CHARs	Optional
paymentAmount	The amount of the payment ex: 1.01	12 CHARs	Optional
creditCardToken	The tokenized number sent in for a credit card payment (USAePay Only)	32 CHARs	Optional
status	The status of the payment (see Appendix for 'Payment Statuses')	16 CHARs	Optional

* Or any other appropriate request field from Appendix B can be submitted to the query

Response Fields

The following response fields are what you may expect from a request to query a group of transactions. Note that for this type of query (where several transactions are returned), the results will be wrapped in <transaction> tags to show the separation between results.

Field	Value	Type	Notes
command	The original 'command' this response is referring to.	32 CHARs	Echoed from request – This is the command that was attempted
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 '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'
paymentMethod	An echo of the	32 CHARs	This varies by type of operation you

	paymentMethod sent in the request originally when the plan was created		are processing
customerAccountNumber	The customer account number for this payment	32 CHARs	This is posted by the merchant at the time the payment is created
approvalCode	The 5-6 digit processor approval code for credit card or debit card payments	6 CHARs	This is returned by the processor for approved payments
trackingNumber	Echoed from the request	16 CHARs	This is a unique ID with a 1 to 1 relationship per transaction attempt.
paymentResponseCode	The payment 'response code' returned from the transaction. In general it's 1 for successful, 2 for declined, 3 for error	1 NUM	This is the value that indicates whether the last payment action was successful or not
paymentResponseText	The current 'response text' for the payment which describes the response code	120 CHARs	The description of the paymentResponseCode. This will be a clearer description of the state of the payment
paymentTransactionID	The processor transaction id if present	16 CHARs	Issued on approvals and some types of declines as well
paymentAmount	The amount of the payment	12 CHARs	This was submitted during the original payment post to us
serviceFee	The service fee of the payment	12 CHARs	This was submitted during the original payment post to us
status	This can be 'Authorized' 'FailedProcessing', 'Retrying', 'Paid', 'Refunded', 'Voided', 'Cancelled', 'Future', 'Declined', 'Returned', 'Unknown'	16 CHARs	A single word describing the 'status' of the payment in the system. This is useful to see if a payment is being queued for re-processing due to a processor failure (Retrying), or to see if a payment has been settled (Paid) vs. unsettled (Authorized).

Example SOAP Request

The following shows an example SOAP request to query for all declined CREDITCARD payments between 1pm and 2pm on New Years Day 2012.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:proc="http://processor">
<soapenv:Header/>
```

```
   <soapenv:Body>
      <proc:processCommand>
         <proc:merchantCode>K3L45JL3K45J34KL435JJLK45JLK345JLK543LKJ543JLKUII
         </proc:merchantCode>
         <proc:command>QUERY</proc:command>
         <proc:paymentMethod>CREDITCARD</proc:paymentMethod>
         <proc:fromDate>2012-01-01 13:00:00</proc:fromDate>
         <proc:toDate>2012-01-01 14:00:00</proc:toDate>
         <proc:status>Declined</proc:status>
      </proc:processCommand>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Response

And here is an example of a raw SOAP response you might expect to receive from the request shown above. Notice the results came back as a list wrapped in <transaction> tags.

```
<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>
            <command>QUERY</command>
            <commandResponseCode>1</commandResponseCode>
            <commandResponseText>Successfully queried transaction.</commandResponseText>
            <transaction>
                <paymentDate>2012-01-01 13:05:02</paymentDate>
                <paymentMethod>CREDITCARD</paymentMethod>
                <paymentSubMethod>Visa</paymentSubMethod>
                <customerAccountNumber>555544433</customerAccountNumber>
                <trackingNumber>1280892202845</trackingNumber>
                <paymentResponseCode>2</paymentResponseCode>
                <paymentResponseText>This transaction is declined.</paymentResponseText>
                <paymentTransactionID>0</paymentTransactionID>
                <paymentAmount>1.01</paymentAmount>
                <serviceFee>0.00</serviceFee>
                <status>Declined</status>
            </transaction>
            <transaction>
                <paymentDate>2012-01-01 13:34:28</paymentDate>
                <paymentMethod>CREDITCARD</paymentMethod>
                <paymentSubMethod>MasterCard</paymentSubMethod>
                <customerAccountNumber>123456</customerAccountNumber>
                <trackingNumber>1280892202765</trackingNumber>
                <paymentResponseCode>2</paymentResponseCode>
                <paymentResponseText>This card has expired.</paymentResponseText>
                <paymentTransactionID>0</paymentTransactionID>
                <paymentAmount>2.02</paymentAmount>
                <serviceFee>0.00</serviceFee>
                <status>Declined</status>
            </transaction>
            <transaction>
                <paymentDate>2012-01-01 13:51:29</paymentDate>
                <paymentMethod>CREDITCARD</paymentMethod>
                <paymentSubMethod>Amex</paymentSubMethod>
                <customerAccountNumber>44400</customerAccountNumber>
                <trackingNumber>1280892203287</trackingNumber>
                <paymentResponseCode>2</paymentResponseCode>
                <paymentResponseText>Invalid card number.</paymentResponseText>
                <paymentTransactionID>0</paymentTransactionID>
                <paymentAmount>3.03</paymentAmount>
                <serviceFee>0.00</serviceFee>
```

```
                <status>Declined</status>
            </transaction>
         </processCommandReturn>
      </processCommandResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

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 web service. 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.

.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

Java / Sun

Examples and explanations of building SOAP clients in JAVA:

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

PHP

Coding SOAP clients using PHP:

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

PERL

Using SOAP with PERL:

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

Python

How to build SOAP clients in PYTHON:

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

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

Appendices

Appendix A - Command Reference

The following is a list of possible 'command' values that can be sent in to the API and a brief description of what each one is used for. They are listed in alphabetical order.

Value	Description
CANCEL_RECURPAY_CUSTOMER	This is used to deactivate a recurring payment plan from the system.
CREATE_RECURPAY_CUSTOMER	This is used to insert a recurring payment plan (payments occurring on a repeating basis) into the system for processing.
QUERY	This is used to pull transaction information or updated details for several transactions based on search parameters you choose.
REFUND	This is used to reverse, i.e. credit, a previously posted payment that has already settled.
SETTLE_BATCH	This is used to manually settle an ACH batch of payments.
TRANSACT	This is used to insert a payment, wire transfer, or future payment for processing.
UPDATE	This is used to update or modify an ACH payment before it has settled, or it can also modify a Future payment (of any type) before it gets processed.
UPDATE_RECURPAY_CUSTOMER	This is used to update or modify the details of a customer's active recurring payment plan.
VOID	This is used to stop a payment before it is settled. In the case of Authorized payments (real time payments that have occurred recently), it changes their status to Voided. In the case of Future based payments, it changes their status to Cancelled.

Appendix B - Request Fields in Alphabetical Order

Following is a list of all Request Fields that can be submitted to the API and a brief description of each.

Field	Type	Description

bankAccountNumber	17 CHARs	The customer's bank account number in cases of ACH or WIRE
bankRoutingNumber	9 CHARs	The customer's bank ABA number for ACH or WIRE payments
billAddress	50 CHARs	The customer's street address
billCity	25 CHARs	The customer's city
billCompany	50 CHARs	The company to charge. Can be used instead of billFirstName and billLastName. Note for ACH and WIRE transactions: The processor has a 35 character limit on this value. We will continue to store the full value you send, but on the processor end, it will be cut off if it's longer than 35 characters.
billCountry	25 CHARs	The customer's country
billDriversLicense	16 CHARs	The customer's drivers license number
billDriversLicenseState	2 CHARs	The customer's drivers license state
billEmail	50 CHARs	The customer's email address
billFirstName	25 CHARs	The customer's first name Note for ACH and WIRE transactions: The billFirstName and billLastName get combined together to form a full name. The processor has a 35 character limit on this full name. We will continue to store the full values you send, but on the processor end, it will be cut off if it's longer than 35 characters.
billLastName	25 CHARs	The customer's last name Note for ACH and WIRE transactions: The billFirstName and billLastName get combined together to form a full name. The processor has a 35 character limit on this full name. We will continue to store the full values you send, but on the processor end, it will be cut off if it's longer than 35 characters.
billPhone	10 CHARs	The customer's phone number
billState	2 CHARs	The customer's US state abbreviation
billZip	10 CHARs	The customer's 5-10 digit zip code
branchNumber	6 CHARs	The merchant company's branch number if segregated into branches
checkDate	10 CHARs	The date to clear the check (on the ACH processor side). We recommend using 'paymentDate' instead as it applies to all transactions and gives more control

command	16 CHARs	The 'command' or operation you are trying to accomplish with a post to the API
creditCardNumber	16 CHARs	The 15-16 digit credit card number, no spaces or dashes, just straight numbers
creditCardToken	32 CHARs	The 16-19 digit USAePay credit card token (USAePay customers only)
customerAccountNumber	16 CHARs	The customer's account or company file number at the merchant company. This should be unique, but isn't required to be.
cvvCode	4 CHARs	The 3-4 digit security code from the customer's credit card
debitCardNumber	16 CHARs	The customer's debit card number
debitCardToken	32 CHARs	The 16-19 digit USAePay debit card token (USAePay customers only)
description	120 CHARs	A description of the payment. Recommended especially for WIRE payments
expireMonth	2 CHARs	The credit card or debit card expiration month
expireYear	4 CHARs	The credit card or debit card expiration year
fromDate	19 CHARs	The date to begin searching from in case of a 'QUERY' command
insertType	3 CHARs	The type of ACH transaction to insert for ACH transactions. Can be WEB, TEL, RTP, etc.
invoiceNumber	16 CHARs	The merchant's invoice or receipt number for this payment. Not required, it's for convenience only to assist the merchant in tracking the order
merchantCode	49 CHARs	The Regal issue encryption key to identify a merchant to the API
numberOfPayments	6 CHARs	The number of payments remaining on a RecurPay plan
paymentAmount	12 CHARs	The payment amount to process
paymentChannel	3 CHARs	The payment channel, i.e. the mechanism that the payment came in by. Can be WEB (for customer initiated online payments), IVR (for automated phone payments), CSR (for representative initiated online payments), MOB (for mobile phone payments)

paymentDate	10 CHARs	The date the transaction was processed on, or should be processed on in cases of Future payments
paymentMethod	16 CHARs	The overall type of payment that was used. Can be CREDITCARD, DEBITCARD, ACH or WIRE
paymentSubMethod	16 CHARs	The sub-type of the paymentMethod. Can be Visa, MasterCard, Amex, Discover, Pinless Debit, Checking, Savings, Business or Unknown
paymentTransactionID	16 CHARs	The processor-issued transaction id after a payment has been sent for processing
recurringAmount	12 CHARs	The amount to charge on a recurring basis in a RecurPay plan
recurringDateOfMonth1	2 CHARs	The 2 digit date of month to process monthly or twice-monthly RecurPay plans
recurringDateOfMonth2	2 CHARs	The second 2 digit date of month to process twice-monthly RecurPay plans
recurringFrequency	1 CHARs	The frequency on which to charge the customer. Can be M (for monthly), T (for twice-monthly), B (for bi-weekly) or W (for weekly)
recurringStartDate	10 CHARs	The date on which to activate or begin the plan
recurringWeekDay	2 CHARs	The 2 digit week day to process weekly or bi-weekly RecurPay plans
serviceFee	12 CHARs	A service fee that the merchant may be charging to the customer for this payment. It gets added to the paymentAmount and the sum total of these two fields is what gets charged.
shipAddress	50 CHARs	The customer's shipping address in cases of physical products
shipCity	25 CHARs	The customer's shipping city
shipCompany	50 CHARs	The customer's shipping company
shipCountry	25 CHARs	The customer's shipping country
shipEmail	50 CHARs	The customer's shipping email
shipFirstName	25 CHARs	The customer's shipping name in cases of a gift or third party
shipLastName	25 CHARs	The customer's shipping last name

shipPhone	10 CHARs	The customer's shipping phone number
shipState	2 CHARs	The customer's shipping US State abbreviation
shipZip	10 CHARs	The customer's shipping zip code
status	16 CHARs	The current state of a payment. Can be one of Authorized, Declined, Voided, Cancelled, Future, Paid, Refunded, Retrying, Returned, FailedProcessing, Unknown (See 'Payment Statuses' in Appendix)
test	8 CHARs	Indicator if the request should be treated as a test or not. Can be 'TRUE' or anything else (anything other than 'TRUE' is interpreted as a non-test payment)
toDate	19 CHARs	The date to search until in cases of a QUERY command
trackingNumber	16 CHARs	The unique identifier for a payment
transactionType	16 CHARs	In cases of ACH payments, this is used to indicate that a CREDIT type of operation is desired. If not present in the request, DEBIT is assumed.

Appendix C - Response Fields in Alphabetical Order

Below are the possible response fields that can be returned by the API. They are listed in alphabetical order for your reference.

Field	Type	Description
approvalCode	6 CHARs	The 5-6 digit approval code as issued by the processor for authorized credit card or debit card payments
command	16 CHARs	The 'command' value from the request, echoed back for clarity
commandResponseCode	1 CHARs	The response of whether the overall 'command' operation was received and attempted. Can be 1 for successful, 2 for declined or 3 for error
commandResponseText	32 CHARs	The response text detailing the reason for the commandResponseCode above
customerAccountNumber	16 CHARs	The customer's account number as entered by the merchant
paymentAmount	12 CHARs	The amount of the payment
paymentDate	19 CHARs	The date the payment has been or will be processed

paymentMethod	16 CHARs	The type of payment, can be ACH, CREDITCARD, DEBITCARD or WIRE
paymentResponseCode	1 CHARs	The code indicating approval or denial of payment type operations. Can be 1 for approved, 2 for declined or 3 for error. See examples under the processing sections
paymentResponseText	256 CHARs	The response text describing the reason for the paymentResponseCode
paymentSubMethod	16 CHARs	The sub-type of payment. Can be 'Visa', 'MasterCard', 'Amex', 'Discover', 'Pinless Debit', 'Checking', 'Savings', 'Business' or 'Unknown'
paymentTransactionID	16 CHARs	The transaction id as issued by the processor
serviceFee	12 CHARs	An echo of the service fee that the merchant sent in. For clarity. It is added to the payment amount at time of processing
status	16 CHARs	The current state of a payment. Can be one of Authorized, Declined, Voided, Cancelled, Future, Paid, Refunded, Retrying, Returned, FailedProcessing, Unknown (See 'Payment Statuses' in Appendix)
trackingNumber	16 CHARs	The unique Regal transaction identification number. All transactions will be issued one of these.
transactionType	16 CHARs	In the case this is a credit type operation this will say CREDIT

Appendix D - Payment Statuses

The following is a list of possible 'status' values to be used in querying and to be observed in the responses for other types of commands. They are what we use internally to track the current state of payments and in general they sum up what has happened with a payment. They are the 'values' to the field named 'status' (see Request Fields or Response Fields in appendices above).

Value	Description
Authorized	The payment has recently been posted for approval and is approved. It has not yet settled, i.e. it is in an open batch and at the end of the day will change to 'Paid'.
Voided	This payment was authorized earlier in the day, but has since been 'Voided' i.e., removed from the batch so that it will not settle nor charge the customer's credit card or bank account.

Paid	The payment is in a settled status and funds have been transferred or are en-route to being transferred.
Refunded	This payment was in a 'Paid' status but the merchant has decided to reverse the payment and return the funds back to the customer, thus the payment becomes 'Refunded'.
Future	This payment has not yet processed. When it's 'paymentDate' arrives it will be processed and will change to either an 'Authorized' or 'Declined' status.
Cancelled	This payment used to be in the 'Future' status but has been cancelled by the merchant and will remain in that state and not get processed on its 'paymentDate'.
Declined	The processor declined this payment.
Returned	This is an ACH payment that after a period of time the ACH processor has notified us that it was returned to the customer, such as when there are insufficient funds, or a dispute and the customers bank decides to refund the customer on their behalf.
FailedProcessing	This status is the original status of all payments and is used internally to indicate and catch issues and errors. We are generally notified about such types of payments and in most cases resolve these if they occur without need of the merchant intervening.
Retrying	This status indicates there was a processor issue or timeout failure or other issue and the payment is now queued up for resubmission. These are fairly rare and when they do occur they usually correct themselves and correctly go into an Authorized or Declined state within 30 minutes or so.
Unknown	This status is the last, bottom of the logic tree, if all else fails, status. It exists so that there will never be a blank status for a transaction. This status is the rarest of all, in fact it has never occurred.

Appendix E - Configuration Requirements for API Functions

API Function	Processing / Configuration Requirements
paymentMethod = WIRE	The merchant must have submitted their merchant bank ABA number, bank code and merchant bank account number to us so that we may configure these values in the WIRE transfer module as being the originating bank that the funds will be drawn from.
debitCardToken or creditCardToken	These values are reserved for merchants using the USAePay tokenization system of storing card numbers. To use this method of payment, the merchant needs to provide us their ueSecurityToken so that we may configure the application on our

	end to successfully post these values through to USAePay.
paymentMethod = ACH	To use real time, future date, or RecurPay ACH functionality, the merchant must have an established Regal ACH account or provide their ACH UserID and API Token from ATI / Affirmative for us to configure the merchant properly in our system