PPI PayMover DevConnect API for PHP
Version 3.0.0

Revised: 30 June 2006

Table of Contents

List of Figures

1 License Agreement

THIS AGREEMENT is entered into between you, the legal entity or individual downloading the PPI On-line Payment Processing Software (the Software) for evaluation or for commercial use, and Payment Processing Inc. (PPI), the provider of the downloadable software, in conjunction with PPI's on-line payment services (the Services).

You are required to read and accept the terms of this agreement before downloading and using the software. BY CLICKING THE "I ACCEPT" BUTTON AT THE END OF THIS AGREEMENT, OR BY DOWNLOADING OR USING THE SOFTWARE, YOU AGREE TO BE BOUND BY ALL THE TERMS OF THIS AGREEMENT. If you do not agree to be bound by the terms of this agreement, click the "I decline" button at the end of these terms and do not download or use the software.

IF YOU ARE A RE-SELLER OF PPI'S SERVICES AND YOU ARE DOWNLOADING THE SOFTWARE ON BEHALF OF YOUR MERCHANTS, by clicking the "I accept" button at the end of this agreement or by downloading the software, you acknowledge and agree that (i) you have the authority to accept these terms on behalf of such merchants and bind such merchants to the terms below; or (ii) you have otherwise required such merchants to agree to the terms set forth below.

IF YOU ARE A DEVELOPER DOWNLOADING THE SOFTWARE ON BEHALF OF YOUR CUSTOMERS USING THE SOFTWARE, by clicking the "I accept" button or downloading the software, you acknowledge and agree that (i) you have the authority to accept these terms on behalf of such customers and bind such customers to the terms below; or (ii) you have otherwise passed through the terms below to your customers using the software.

1. LICENSE. Payment Processing Inc. grants to you a non-exclusive and non-transferable license to use the Software for internal evaluation and development for integration with your products and services and solely in conjunction with the Payment Processing Inc. Services.

2. LICENSE RESTRICTIONS. Unless enforcement is prohibited by applicable law, you may not modify, decompile, reverse engineer or otherwise attempt to derive the source code of any Software made available to you. You may not sell, lease, rent or otherwise distribute or commercially exploit the Software. You may not create derivative works of the Software or use any components of the Software separately from the Software. You agree that all intellectual property rights in and to the Software are and shall remain the sole property of Payment Processing Inc. and its licensors, and you shall have no rights beyond those specifically granted in this Agreement. The Software is confidential and protected by copyright laws and treaties. Use of Payment Processing Inc.'s or its licensors' trade marks or names is prohibited except with express prior written approval.

3. WARRANTY DISCLAIMER. PAYMENT PROCESSING INC. AND ITS LICENSORS MAKE NO WARRANTY, EXPRESS, IMPLIED OR STATUTORY WITH RESPECT TO THE SOFTWARE, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS, ALL OF WHICH ARE EXPRESSLY DISCLAIMED. YOU ACKNOWLEDGE THAT PAYMENT PROCESSING INC. HAS NOT REPRESENTED OR WARRANTED THAT THE SERVICES WILL BE UNINTERRUPTED, ERROR FREE OR WITHOUT DELAY OR WITHOUT INFILTRATION OR COMPROMISE OF THE SECURITY SYSTEMS RELATED TO THE SERVICES.

4. LIMITATION OF LIABILITY. IN NO EVENT WILL PAYMENT PROCESSING INC. OR ITS LICENSORS HAVE ANY LIABILITY TO YOU OR ANY OTHER THIRD PARTY FOR ANY DAMAGES SUFFERED AS A RESULT OF USING, MODIFYING, COPYING, DISTRIBUTING OR DOWNLOADING THE SOFTWARE OR FOR ANY LOST OPPORTUNITY OR PROFITS, INJURY TO ANY CUSTOMER RELATIONSHIP, COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES. IN PARTICULAR, PAYMENT PROCESSING INC. WILL HAVE NO LIABILITY FOR ANYINDIRECT, PUNITIVE, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGE, HOWEVER ARISING OUT OF THE USE OF THE SOFTWARE OR THIS AGREEMENT, UNDER ANY CAUSE OF ACTION OR THEORY OF LIABILITY (INCLUDING NEGLIGENCE) AND WHETHER PAYMENT PROCESSING INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THESE LIMITATIONS WILL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY.

5. TERMINATION. This Agreement is effective until terminated. You may terminate this Agreement at any time by destroying all copies of the Software. The license granted in section 1 above will terminate immediately and without notice from Payment Processing Inc. in the event you fail to comply with any restriction contained in section 2 above or any other obligation contained in this Agreement. Upon such termination, all rights granted under this Agreement terminate and you are required to destroy all copies of the Software.

6. EXPORT. You acknowledge and agree that you shall not import, export or re-export directly or indirectly, any commodity, including the Software, to any country in violation of the laws and regulations of any applicable jurisdiction. This restriction expressly includes, without limitation, the export regulations of Canada and the United States. Specifically, you agree that the Software is not being acquired for, shipped, transferred or re-exported, directly or indirectly, to proscribed or embargoed countries or their nationals. Proscribed countries include those set forth in the U.S. Export Administration Regulations. Countries subject to embargo are: Cuba, Iran, Iraq, Libya, North Korea, Serbia, Syria, the Sudan, and the Taliban-controlled areas of Afghanistan. This list is subject to change without further notice from Payment Processing Inc. and you must comply with the list as it in fact exists at any time. You certify that you are not on the U.S. Department of Commerce's Entity List, Denied Persons List or affiliated lists, or on the U.S. Department of Treasury's Specially Designated Nationals List . You agree to comply strictly with all applicable Canadian and U.S. export laws and assume sole responsibility for obtaining licenses to export or re-export as may be required.

7. JURISDICTION. This Agreement will be governed by and construed in accordance with the laws of the Province of British Columbia and those of Canada applicable therein without giving effect to any principles of conflicts of laws. All disputes or claims arising out of this Agreement will be submitted to the courts of the Province of British Columbia. You submit and attorn to the exclusive jurisdiction of the courts of the Province of British Columbia to finally adjudicate or determine any suit, action or proceeding arising out of or in connection with this Agreement.

2 Introduction

This document is intended for merchants who wish to integrate the PPI PayMover DevConnect API for PHP into their payment process. The PPI PayMover DevConnect API for PHP connects PHP applications and web pages to the PPI PayMover Payment Service. It provides a simple API that can be integrated into an application to enable credit card processing and Automated Clearing House (ACH) processing.

In addition to ACH and simple credit card transaction processing, the API also provides the ability to perform credit card payer authentication for programs such as Verified by Visa, MasterCard SecureCode and JCB J/Secure. The API can also be used to set up periodic recurring payments for credit cards, and to settle and query the contents of daily batches.

Readers of this document should be familiar with the contents of the *PPI PayMover Payment Service User Guide*. It covers request and response field requirements and important concepts for payment processing.

The PPI PayMover Payment Service provides merchants with secure, real-time credit card and ACH processing. This service includes reporting and administration interfaces, and has been integrated with many major e-commerce platforms for merchants of all types and sizes.

The PPI PayMover API for PHP uses Secure Sockets Layer (SSL) to communicate with the PPI PayMover Payment Service.

3 Requirements

- An account on the payment service. (This is not required to do test transactions).
- PHP 4 or newer (http://www.php.net).
- Ability to develop in your environment.
- If you will be accepting credit card information over the Internet, you must have an SSL certified secure web server.
- A web server configured to process PHP documents.
- OpenSSL (http://www.openssl.org).
- cURL module for PHP (http://curl.haxx.se). *Note that even if you are using the PHP cURL extension module (`php_curl.dll`), you will still have to get the non-compiled module in order to get the `ca-bundle.crt` file.
- Open port 443 for bi-directional https TCP traffic.

4 Installation

The payment API for PHP consists of several `.php` files along with example code demonstrating how to use them.

Unzip the distribution file to a new directory and take a moment to examine its contents. You should find:

Contents	Description
`*.php`	Payment API files for PHP.
`doc/`	Directory containing this document and the Payment Processing Inc license agreement. By using the payment API for PHP, you are bound by this license agreement.
`doc/apidocs/`	Directory containing documentation describing the methods and data types available in the PHP API.
`samples/`	Directory containing the sample transaction processing pages.
`samples/ACH/`	Directory containing the Automated Clearing House (ACH) sample transaction processing pages.
`samples/Batch/`	Directory containing the Batch sample transaction processing pages.
`samples/CreditCard/`	Directory containing the Credit Card sample transaction processing pages.
`samples/PayerAuthentication/`	Directory containing the Payer Authentication sample transaction processing pages.
`samples/Recurring/`	Directory containing the Recurring sample transaction processing pages.

In order to use the API, you must have the cURL module for PHP installed.

To begin using the API from within a PHP application, copy the payment API files to your web server and add the following line to your PHP application:

```
include("Paygateway.php");
```

Then you can make use of the PHP API's methods to perform credit card transaction processing. Refer to the sample applications and the API docs for examples of how to use the API.

5 Using the Sample Applications

The payment API for PHP is distributed with several sample applications: one demonstrates credit card transactions, one the use of the batch transactions, one the use of recurring transactions, one the use of payer authentication transactions, and the other demonstrates the use of the Automated Clearing House (ACH) transactions.

The sample application includes one `.html` file and one `.php` file. The HTML file accepts parameters to pass to the payment API. The HTML file posts to the `.php` file which process the form data, instantiates the API and performs the required transaction.

For information about the required fields for each transaction type, refer to the *Payment Service User Guide*.

> **Note**
>
> If you are receiving the error message: "*SSL certificate problem, verify that the CA cert is OK*", you may need to set the location of your `ca-bundle.crt` file. This can be done by using the `setCABundle()` method. Alternatively, you can set the environment variable `CURL_CA_BUNDLE` to achieve the same result.

5.1 Credit Card Request Sample Application

To run the sample credit card pay page, copy the files from the sample application directory, along with the payment API files to a directory on your web server's document path and navigate to `demopaypage.html` within a browser.

You should see a page similar to the one shown below:

Figure 5.1: Sample Credit Card Transaction Pay Page

When you submit the form, you'll see a results page:

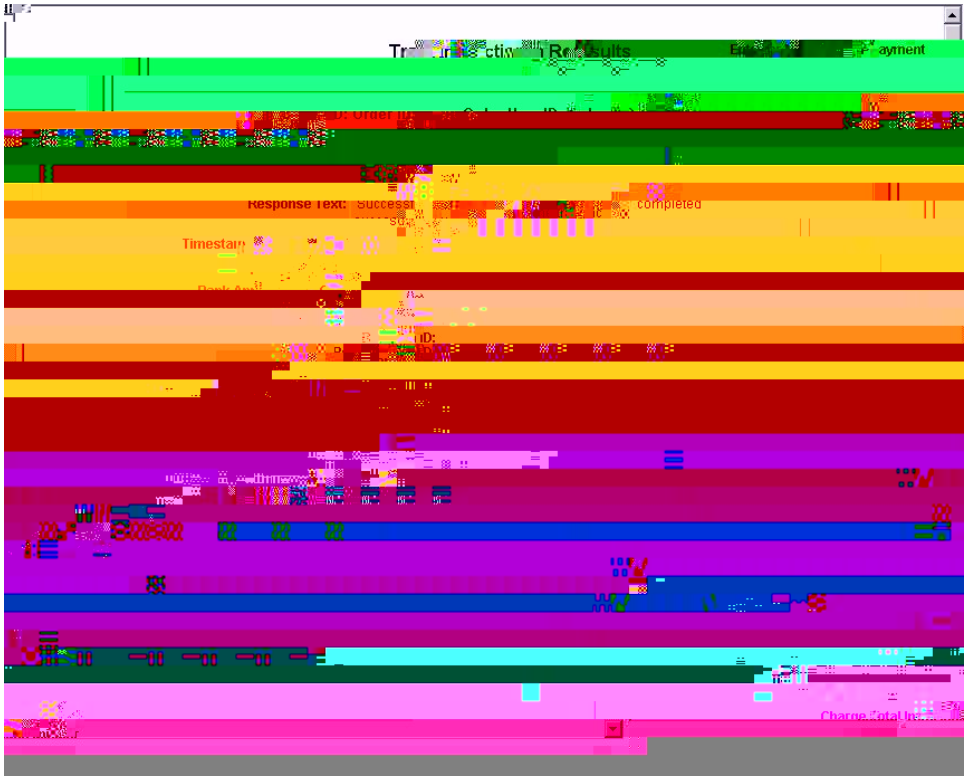


Figure 5.2: Sample Credit Card Transaction Results Page

To test various response codes, refer to the penny value tests described in the *Payment Service User Guide*.

5.2 Recurring Request Sample Application

The recurring client example shows how to create recurring transactions for the recurring billing system. The recurring billing system is an optional feature available with your account. The recurring billing system has a user-friendly web-based interface for performing periodic transactions. As an automated alternative to this interface, you can use the PHP API to enter transactions. For information on the recurring billing system, see the *Payment Service User Guide*.

To run the sample recurring pay page, copy the files from the sample application directory, along with the payment API files to a directory on your web server's document path and navigate to `demorecurringpaypage.html` within a browser.

For information on the available fields for recurring transactions, see the *Payment Service User Guide*.

5.3 Batch Request Sample Application

The batch client demonstrates how to use the PHP API to settle batches of authorized credit card transactions. This is entirely optional. Batches settlement is automatically done daily by default, so you only need to use batch requests if you want to automate batch closure yourself.

To run the sample batch request, copy the files from the sample application directory, along with the payment API files to a directory on your web server's document path and navigate to `demorecurringpaypage.html` within a browser.

For information on the available fields for batch transactions, see the *Payment Service User Guide*.

5.4 Payer Authentication Sample Application

The payer authentication sample application demonstrates how to use the payment API to decrease fraud and merchant liability by accepting cards that are enrolled in Verified by Visa, MasterCard SecureCode and JCB J/Secure.

The payer authentication sample application is a web-based application that uses PHP pages.

To run the sample payer authentication application, you need to adjust the settings in the `BeginPayerAuthentication.php` file first. The following properties are required:

Property	Value
`AccountToken`	This is your unique identifier for your account on the payment service. By default a test token is provided that will not charge any credit cards.
`TransactionURL`	The fully qualified URL for the web page that will complete the credit card transaction after the buyer has entered their user name and password.

> **Note**
>
> The URL used for the `TransactionURL` must have the same domain that was used to navigate to the pay page with.

Once you have configured the sample application by editing `BeginPayerAuthentication.php`, you can navigate to the sample pay page and perform a test transaction. Enter the URL to the `PayPage.html` file into your browser. The URL will be:

```
http://<hostname>/PayerAuthentication/PayPage.html
```

You must not use `localhost` in the URL in order to facilitate the redirects required for payer authentication transactions. Use your IP address or machine name instead; the domain should match the domain you use in the `TransactionURL` key in `BeginPayerAuthentication.php`. You should see a pay page similar to the one shown below.

Figure 5.3: Sample Payer Authentication pay page

Fill in the required fields to perform a test transaction. To perform a simple test, use the credit card number *4000000000000002* and a charge total of *1.01*. These are test values that result in a successful VBV authentication and a successful credit card transaction. Refer to the *Payment Service User Guide* for other test credit card numbers and penny values.

After clicking *Submit*, the buyer will be presented with an authentication page prompting for a password.

Figure 5.4: Payer Authentication test page

This is a test authentication page. When your account is in test mode, or the word *TEST* is prepended to your account token, you will see this sample page. On real transactions, the authentication page will come from the issuing bank and appear slightly different. Click *Submit* to continue.

After the authentication page had been submitted, the transaction will continue. You should see a results page similar to the one shown below.

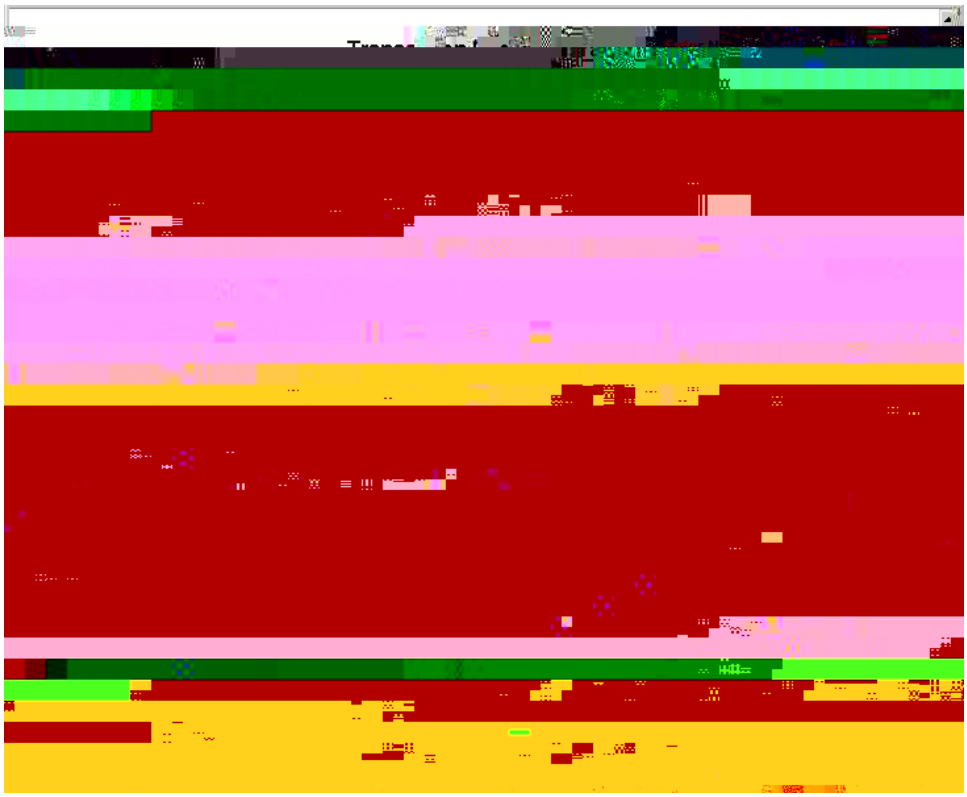
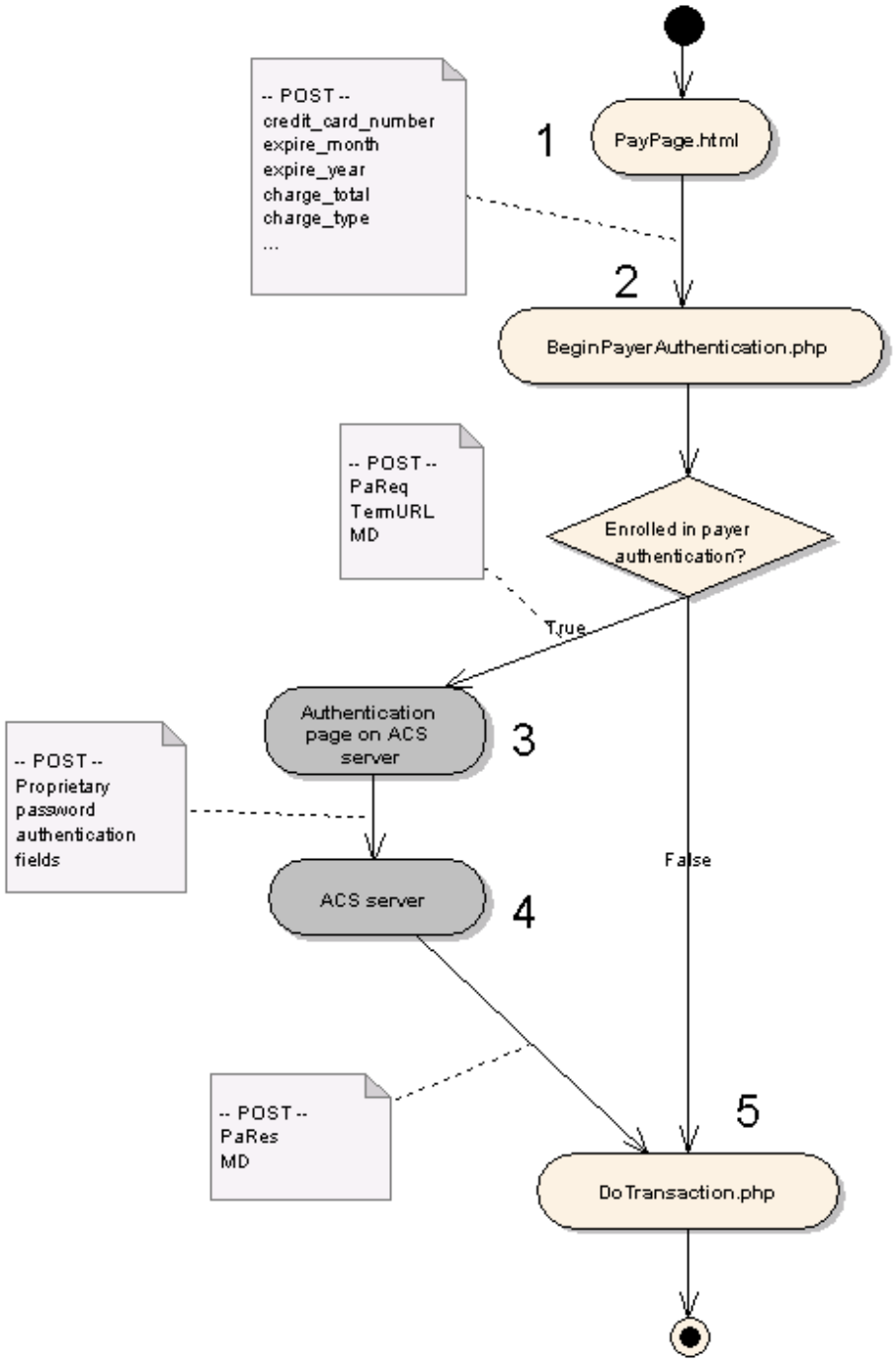


Figure 5.5: Successful test transaction results page

The results page shows two sets of response fields. The first set is the results of the financial transaction. The second set is the *Payer Authentication Response Fields*. When payer authentication is used, these fields show the results of the authentication. A *Response Code* of 1 indicates success.

5.4.1 Transaction Flow

During the transaction flow, there are two messages passed between the payment API and the payment service. The first message is a `AuthenticationRequest` which is used to see if the cardholder is enrolled in a payer authentication service. The second message is a `TransactionRequest` which is used to perform the actual financial transaction. If the response from the `AuthenticationResponse` indicated that the cardholder was enrolled in payer authentication, then the merchant site will redirect the buyer to a URL found in the response object in order to perform authentication. The following diagram describes the transaction process and shows what data fields are posted between the web pages. This diagram is best understood by following along in the source code for the php files to see how each step is implemented.



1. The transaction flow begins on the pay page. In a real merchant application, this page will be dynamically generated and will likely display the products being ordered and the charge total to be billed. The buyer is prompted to enter their contact information and credit card number and expiry date. Upon submitting this form, the information is posted to `BeginPayerAuthentication.php`.

2. `BeginPayerAuthentication.php` creates a `AuthenticationRequest` with the action set to lookup. This request is sent to the payment service to determine if the buyer's card is enrolled in a payer authentication program. If they are enrolled, the response will include a URL that will be used to perform the payer authentication. If the card is not enrolled, then control will pass to `DoTransaction.php` and the transaction will continue normally (without any authentication).

3. After verifying that the credit card is enrolled in payer authentication, the browser is re-directed to the ACS URL returned by the `AuthenticationResponse`. This step is shown in grey to indicate that control has now been passed away from the merchant's web site. The merchant has no control here and will never be able to access the cardholder's password.

4. The password is submitted to the ACS server for processing (control is still outside of the merchant application). The data is encrypted and posted back to the merchant site at the URL that was provided in the *TermURL* field that was posted to the ACS page previously.

5. By arriving at Step 5, the buyer has either entered their payer authentication password, or they are not enrolled in payer authentication. To perform a financial transaction, a `TransactionRequest` object is created and all of the billing and transaction information is set from the session data. If the posted variable *PaRes* contains data, we know they are enrolled and have already entered their password. To complete the payer authentication, the authentication payload field in the `CreditCardRequest` object is set to the value of *PaRes*, and the authentication transaction ID is set to the value returned by the lookup request in Step 2. Finally, the transaction condition code must be set to the value of the constant: `TCC_CARDHOLDER_NOT_PRESENT_PAYER_AUTHENTICATION`. The credit card request can then be processed. If all the required payer authentication fields were present, the `TransactionResponse` object will contain the results of the authentication. Otherwise, the transaction proceeds without payer authentication.

> **Note**
>
> To use payer authentication, the `TransactionRequest` must have the transaction condition code set to `TCC_CARDHOLDER_NOT_PRESENT_PAYER_AUTHENTICATION`. The actual transaction condition code sent to the gateway varies depending on the authentication response in the `TransactionResponse` object.

Information about the transaction will be displayed to standard output. This will be visible in the console window that the web server was started in. Following this output will help to understand the transaction flow.

5.5 Automated Clearing House (ACH) Transactions

To run the sample ACH page, copy the files from the sample application directory, along with the payment API files to a directory on your web server’s document path and navigate to `demoachpage.html` within a browser. Fill in the information for the required fields, press submit and you will get a page showing the result of the transaction.

For information on the available fields for ACH transactions, see the *Payment Service User Guide*.