C3 PLP Documentation

Component of

Documentation Version 35

TABLE OF CONTENTS

- 1. What is PLP?
- 2. Monitor Your Fees!
- 3. Shopping Cart Integration Overview
  - 3.1. Fraud Prevention
    - 3.1.1. CAPTCHA Required Prior to PLP Presentation
    - 3.1.2. What is Card Testing?

        - 3.1.3. What is the impact of Card Testing?
        - 3.1.4. What could make you more susceptible to Card Testing?
        - 3.1.5. Recommendations:
    - 3.2. TIP: How to Develop with PLP
        - 3.2.1. Simulation, Destinations, Responses, and Testing
    - 3.3. Standard Operation Concept
- 4. PCI Security Ramifications of PLP
    - 4.1. PLP Data Flow
        - 4.1.1. CONCEPTUAL DEMO
        - 4.1.2. Actual Demo Screens
- 5. Step 1 - Initialize a transaction session
    - 5.1. Example PLP Source Code Fragment in PHP
- 6. Step 2 - Display the iFrame
    - 6.1. The MODE Parameter
        - 6.1.1. Embedded Mode for iFrame
        - 6.1.2. Redirect Mode for C3 Payment Page
    - 6.2. PLP DEMO Example Source Code
    - 6.3. Example Required JavaScript for PLP
    - 6.4. Example JavaScript for responding to PLP events in an iFrame
- 7. Step 3 - User enters card data and submits transaction
- 8. Step 4 - PLP redirects to the target URL via POST
- 9. Modifying the PLP CSS
    - 9.1. Using the PLP Demo to Work with CSS
- 10. Request Data Fields Definitions
- 11. Response Data Fields Definitions
- 12. MFATAL Codes for (UL) Local Errors
- 13. CURBSTONE API FIELDS - COMPLETE
- 14. Customizable Cosmetic Elements
    - 14.1. Client.ini File
    - 14.2. Header.html File
    - 14.3. Footer.html File
    - 14.4. Form_header.html File
    - 14.5. Form_footer.html File
    - 14.6. Stylesheet.css File
        - 14.6.1. PLP Default Styles
    - 14.7. CSS/HTML Modification Stipulations
- 15. Demo Source Code for Reference
    - 15.1. Demo: CSS Style File
    - 15.2. Demo: config.ini Configuration File
    - 15.3. Demo index.php: Example Checkout Page Source Code
    - 15.4. Demo payment.php: Example Payment Page Source Code
    - 15.5. Demo return_url.php: Example Demo Target Page Source Code
- 16. MAGENTO 2 Extension
- 17. C3 Isolated Card Entry - ICE
    - 17.1. ICE Demo Source - index.php

    - 17.2. ICE Demo Source - payment.php
    - 17.3. ICE Demo Source - return_url.php
- 18. C3 "Check Operation" Utility
- 19. FAQ

---

1. What is PLP?

Curbstone Payment Landing Pages (PLP) is an e-commerce integrator between the Curbstone credit card payment processing engine and any e-commerce platform. PLP provides for secure, PCI-compliant e-commerce transactions by separating the transmission, storage, and processing of credit card transaction data from the e-commerce shopping cart, allowing the shopping cart and its server to be 'out of scope' for PCI compliance. In this way, only the secure PLP with the Curbstone processing engine running on the AS/400 is responsible for transmission, storage, and processing of credit card transaction data and is 'in scope' for PCI compliance. As with all implementations of this type, the e-commerce side would only have to complete the minimal PCI SAQ A audit form that is very short.

PLP can be run on any operating system with any language that can consume web services that are engaged with HTTPS protocol.

2. Monitor Your Fees!

3. Shopping Cart Integration Overview

Often, the shopping cart/e-commerce solution will reside on a different box than the AS/400 (System i) and may run on Windows or Linux. At the point the customer order is complete, the shopping cart will engage PLP so that the credit card transaction can be processed **out of scope of PCI** on the web server. The response is returned to the shopping cart via an embedded return address (URL) provided by the cart. Only the TOKEN for the transaction is returned, and the credit card number is masked except for the last four digits. In this way, the shopping cart software and its server never store or transmit credit card transaction data, allowing them to be **out of scope of PCI**.

The shopping cart that integrates to this PLP can be of any type, including:

1. Local or remote host
2. Any Linux, Windows, Unix, AIX, OS/400, i5/OS, IBM i, or z/OS web server
3. Any web language including PHP, ASP, .Net, C#, VB.net Perl, Cold Fusion, C, C++, Java, Python, Cobol, RPG, or other language
4. All major proprietary and open source shopping carts including Drupal, Magento, Znode, etc...

In this way, Curbstone PLP is agnostic to the details of the shopping cart engine and supports integration with any shopping cart across all platforms with little or no customization required. A common checkout process for most commercially available shopping carts occurs as a 4-step process carried out on the e-commerce server:

Order Summary > Shipping > Payment > Confirmation

Where:

1. Order Summary page allows the customer to inspect their order before completing checkout
2. Shipping page form collects bill to and ship to information
3. Payment page collects payment credit card transaction information (Curbstone PLP)
4. Confirmation page provides order confirmation information.

3.1. Fraud Prevention

3.1.1. CAPTCHA Required Prior to PLP Presentation

Curbstone **REQUIRES** the implementation of the latest version of **CAPTCHA** on the page immediately prior to the PLP iFrame. That page must NOT contain the iFrame HTML code, even if it is non-displayed or “hidden”. In most cases, CAPTCHA will not present any challenge to legitimate visitors.

WARNING - CAPTCHA REQUIRED
Curbstone customers found to not have implemented CAPTCHA immediately prior to the presentation of the PLP iFrame will be shut down immediately.

To deter or prevent card testing, phishing attacks, and/or bot attacks, you ***ARE REQUIRED TO*** add a CAPTCHA, to present and then satisfy it BEFORE the PLP iFrame is presented. The page that contains the CAPTCHA cannot also have the HTML code for the iFrame embedded, even if it is "hidden" or set to non-display. CAPTCHA cannot be loaded into our iFrame, by you or by us, that is not an option. The CAPTCHA authentication has to happen from the hosting page that CONTROLS ACCESS to the iFrame.

The goal is to prevent non-CAPTCHA-authenticated users from accessing - clicking through to - the card entry in the iFrame, and the above accomplishes that effectively. This is the Best Practice standard for embedded iFrames.

3.1.2. What is Card Testing?

Card Testing is a trial-and-error method used by fraudsters to obtain, within seconds, payment card information such as an account number, card expiration date or Card Verification Value 2 (CVV2), as well as a user password for online account access.

In this attack, automated software commonly known as a “botnet” is used as a downloader or a credential-collection tool that generates a large volume of consecutive guesses of account data. A fraudster can continue to run credit card numbers through merchant websites until the authorization response comes back approved.

3.1.3. What is the impact of Card Testing?

Card Testing can cause excessive authorization fees to be charged to the merchant's account for each attempt when not dealt with properly.

It is the responsibility of each software developer put proactive measures in place to prevent this type of activity for their merchants. Authorization fees for Card Testing attack can quickly accumulate as these types of attacks tend to involve several cards so the fraudster can gather as much information as possible.

3.1.4. What could make you more susceptible to Card Testing?

Any application that enables online payments and has not implemented online payment best practices, including implementing CAPTCHA, is at risk for a Card Testing attack.

3.1.5. Recommendations:

Note that Curbstone REQUIRES that the latest version of CAPTCHA be implemented on the page immediately prior to the presentation of our PLP iFrame. The rest of the items below are recommendations.

We strongly recommend that you consider the implementing the following best practices to help mitigate card testing and other fraudulent attacks:

- Captcha controls and Three-Domain Secure (3DS) authentication
    - This may help to prevent automated transaction initiation by robots or scripts (for example, five authorizations from one IP address or card).
- Use a layered validation approach
    - CVV2 and Address Verification Service (AVS)
    - Monitor IP Addresses
        - Include IP address with multiple failed card payment data in a fraud detection's black-list database for automated or manual review
        - Look for logins for a single card account coming from many IP addresses.
    - Velocity Checks
        - Use for small and large transactions as well as authorization-only transactions.
    - Throttling
        - Throttling injects random pauses when checking an account to slow brute force attacks that are dependent on time.
    - Monitor Processing Patterns
        - Excessive usage and bandwidth consumption from a single user.
        - Multiple tracking elements in a purchase linked to the same device. (Example, multiple transactions with different cards using the same e-mail address and sane device ID)
    - Monitor Login attempts
        - Lock out an account if a user guesses the user name / password
        - Lock out any account authentication data incorrectly on "x" number of login attempts.
    - Behavioral Biometrics
        - Takes into consideration a variety of elements. Some of the Card Brands have developed these programs that are available at a cost.

3.2. TIP: How to Develop with PLP

This is likely new tech for you. Having overseen many implementations of our different technologies and APIs, we can make one very strong suggestion. We provide a DEMO site for PLP listed in these docs. Think through your plans to use PLP on your web site. Write down the transaction types and their field contents in the same sequence you intend to use them in your site.

Start with the default DEMO as presented and play with it. Then start PROTOTYPING your specific transactions in the DEMO screens. Walk through them one by one and SEE what they look like in the DEMO screens. Once you have confidence in the transactions, their types, and the field contents, then start using bad data.

Note that you also get to see the EXACT returned responses that your program will see. Test your sequence of operations and see every possible screen, response, and error that can happen.

See documentation file **C3_Master_Reference_xx.pdf** on the support site at **https://curb911.com** which covers all of the Curbstone fields.

Once you know how your intended transactions will work in the DEMO screens, start to develop your code for your site. You should know what to expect, as the results should mimic what you prototyped.

3.2.1. Simulation, Destinations, Responses, and Testing

Use the DEMO as a proofing tool. Your code will ALWAYS work the same as the DEMO. We have the demos now running against both the LIVE Portal and the SANDBOX Portal. We prefer you to work against the SANDBOX first. Once you are comfortable, and things are working, move to the LIVE Portal. You should see no difference in operation. We also provide TWO test destinations for the transactions on both destinations, MFMRCH 99998 and 97777. The first is a SIMULATOR that provides dummy responses directly from the Portal. The latter connects to the TSYS test server that they graciously allow use, and you, to test against at any time. If you will be using another network than TSYS, the fields MFRCVV, MFRAVS, and MFRREF may return slightly different values. These are three fields that we pass through without abstracting. This allows you direct access to their field values without our interpretation.

See documentation file **C3_Master_Reference_xx.pdf** on the support site at **https://curb911.com** which covers all of the Curbstone fields.

3.3. Standard Operation Concept

In the preferred mode of operation, the payment page generated from the C3 Portal is embedded in an iFrame within the shopping cart payment page.

In the case of iFrames (i.e. field *mode* = "embedded", default if the "mode" parameter is not set by the shopping cart), PLP leverages the child-parent relationship which exists due to the fact that the PLP application is embedded in the shopping cart via iFrame. PLP will "bust out" of the iFrame upon successful completion of the transaction, and securely POSTs the results to the "target" URL specified by the merchant - the "return address" - Curbstone field MPTRGT. This URL on the merchant's side just needs to accept and process such a POST, which includes an array of fields, only some of which may be important to the Shopping Cart program. These fields include the bank responses as to the validity of the account, the match result on the Address Verification, and the match result of the security code (CVV), and the Approval or Decline (MFRTRN=UG or UN respectively).

The target URL/script would then redirect to a confirmation page after processing what parts of the card response they care about.

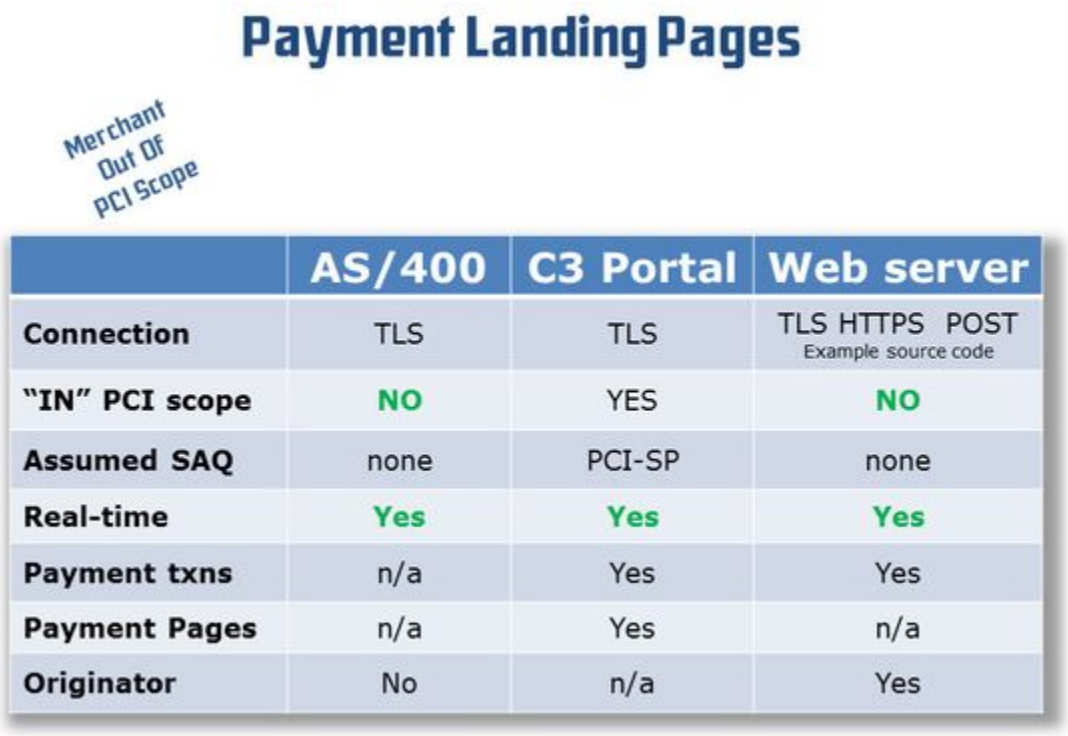
The cart might even be able to combine the processing of the POST and the redirection to the confirmation page into a single page-rendering script, depending on how their web application framework is structured/works. This is easy to do in PHP.

So, in summary:

1) "***embedded***" (i.e. iFrame) mode returns results via busting the iFrame and POSTing to a specified MPTRGT URL.

2) "***redirect***" mode entails your shopping cart redirect completely to a Curbstone (3rd party vendor) website, at this time, https://c3plp.net. This is similar functionality to that employed by sites integrating with PayPal. At the end of the payment, the results are also sent to the "target URL" using HTTPS POST from the Curbstone Portal.

4. PCI Security Ramifications of PLP

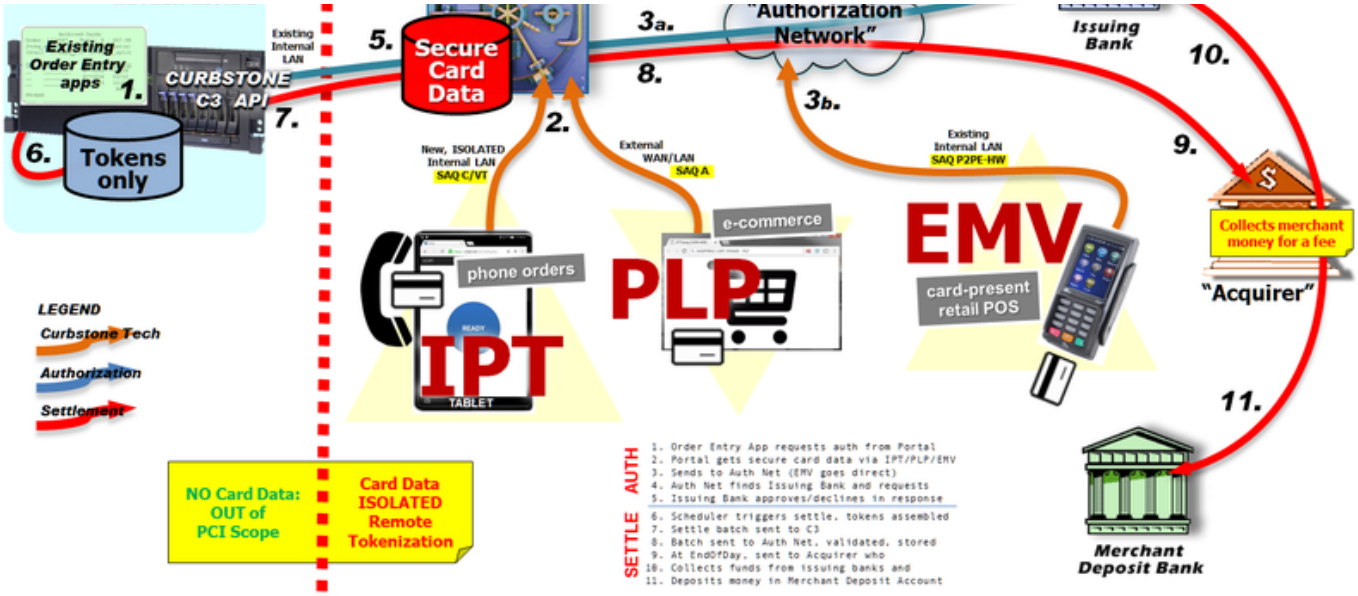


	AS/400	C3 Portal	Web server
Connection	TLS	TLS	TLS HTTPS POST Example source code
"IN" PCI scope	**NO**	YES	**NO**
Assumed SAQ	none	PCI-SP	none
Real-time	**Yes**	**Yes**	**Yes**
Payment txns	n/a	Yes	Yes
Payment Pages	n/a	Yes	n/a
Originator	No	n/a	Yes

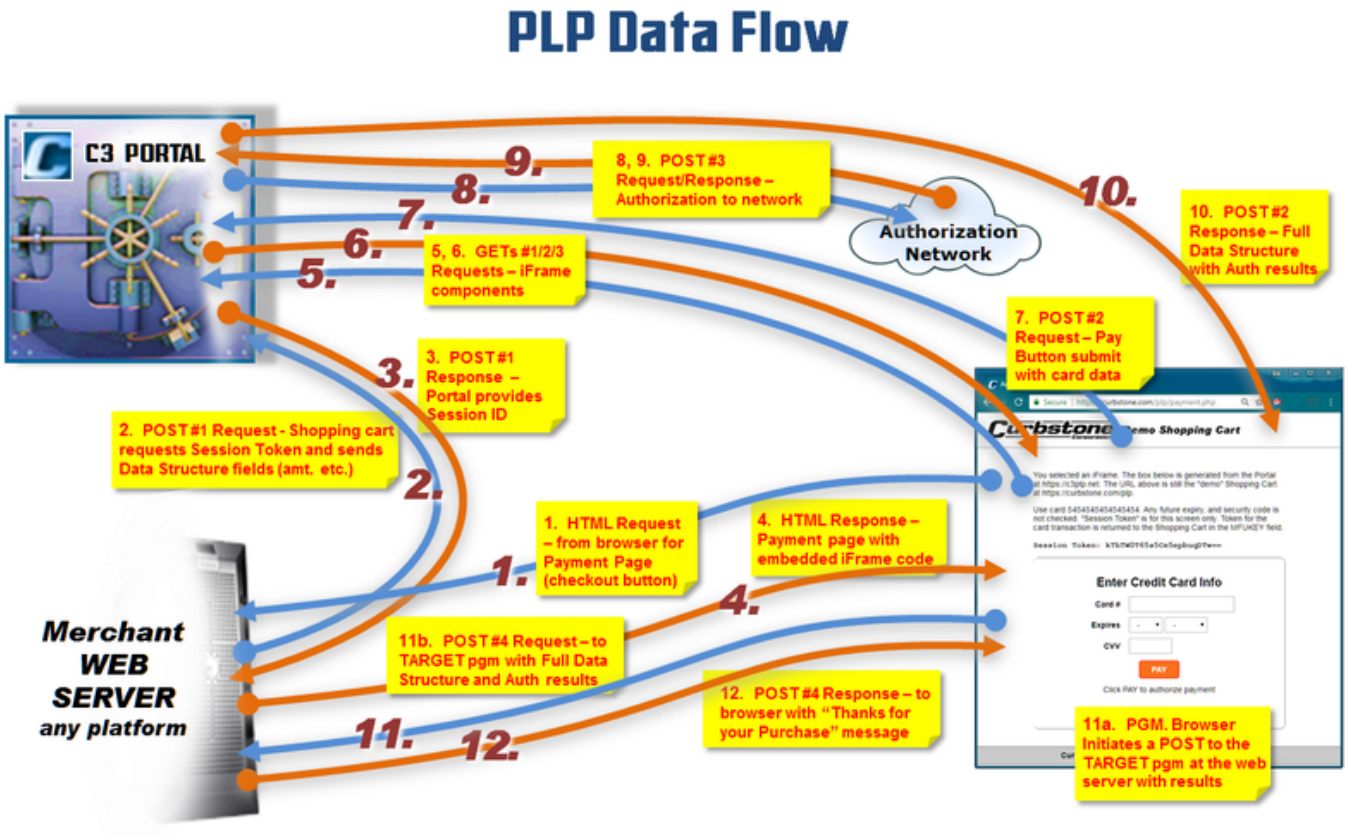
The decision as to what PCI compliance requirements affect your organization can ONLY be determined by a Qualified Security Assessor (QSA). Curbstone provides tools and technology that can take your systems out of PCI scope, however the actual effectiveness of your implementation can only be determined by a third-party QSA.

4.1. PLP Data Flow

Overview:



Specifics:



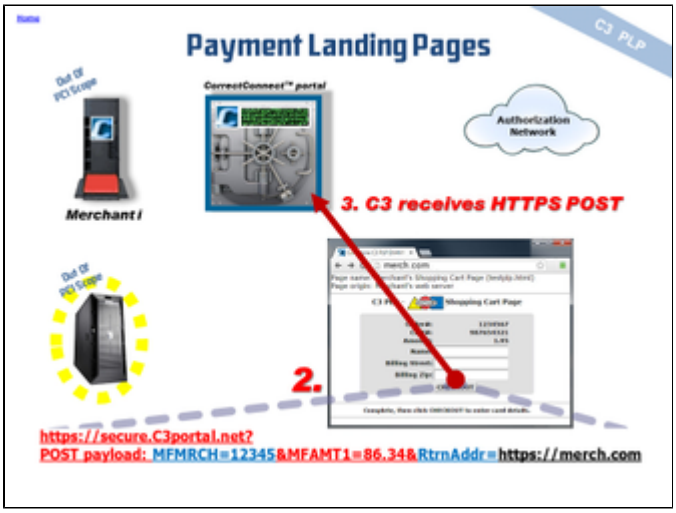
PLP deployed from the Curbstone Portal under CorrectConnect, C3

Note that **these are conceptual examples** and do not properly represent the actual POST process detailed further below in source code. This example does not reflect the use of iFrames, though they are supported in PLP Version 1.0 and demonstrated in the source code examples below in the Steps.

4.1.1. CONCEPTUAL DEMO

Shopping Cart screen is generated by Merchant Web Server for completion of order without card data.

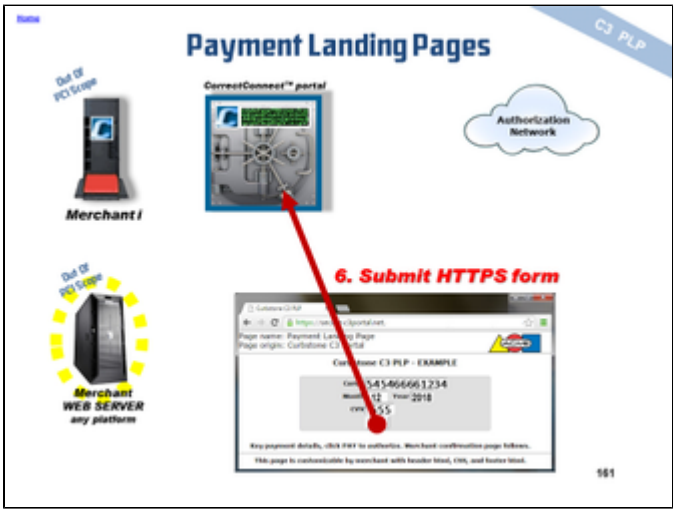
The Submit button of the Cart submits an HTTPS POST form to the Curbstone Portal with all of the order data to prevent re-keying. This includes the Billing Address, Billing Zip, Amount, and other fields required for the card authorization.



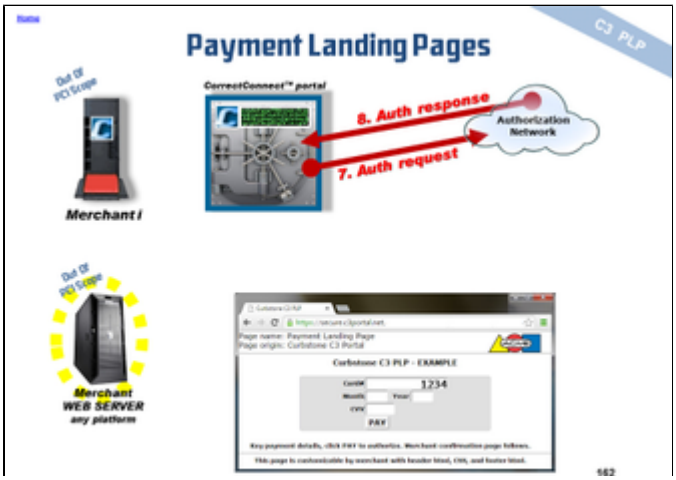
Once the form is processed by the Portal, it generates a payment page for the visitor to complete. Note the ACME Logo, representing that the payment screen is customizable to match the general look and feel of the Shopping Cart page.

The visitor completes the card info and clicks Pay.

The transaction goes back to the Portal for authorization.

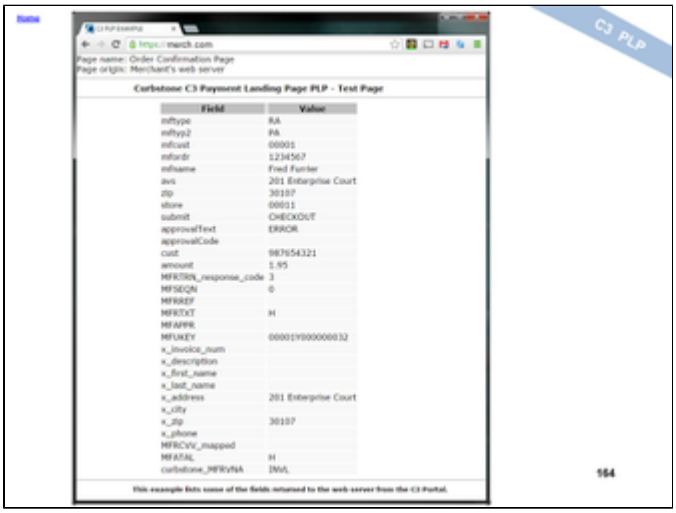


Within one to three seconds, the response is obtained, and the Portal then "sends" the visitor back to the Shopping Cart, along with a payload of fields that show the results of the authorization attempt.

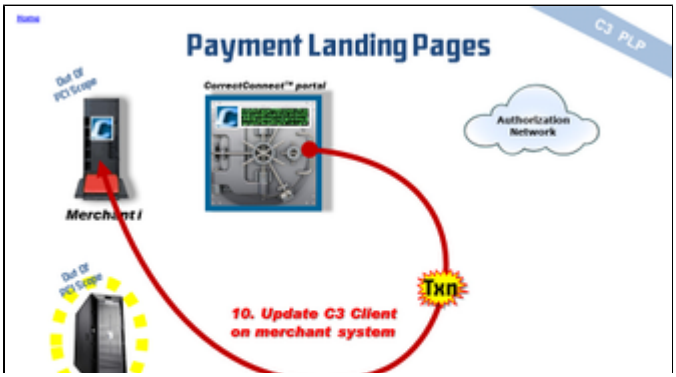


This payload is read by the receiving program - written by the Shopping Cart programmers - and used to complete the order.

This response contains ALL of the fields required for the Shopping Cart to know exactly what happened.

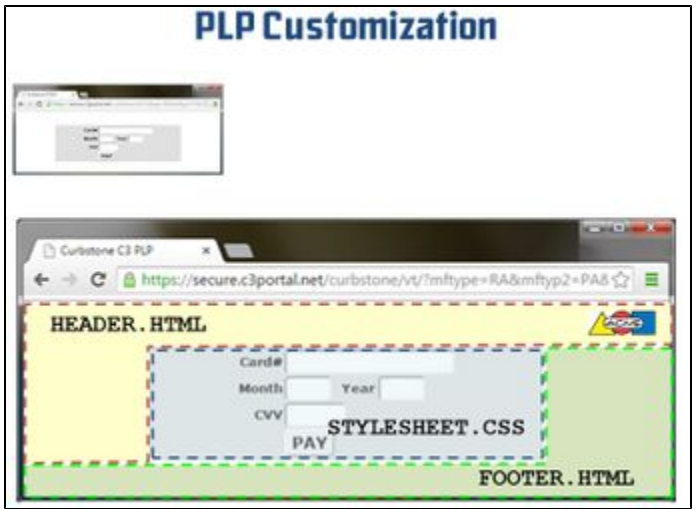


In addition, behind the scenes, the new authorization record is made available to the Native C3 Interface on the Merchant's AS/400, System i, so the payment data can be used to complete the order, or stored for later use as a card-on-file. At no time does actual card data come back to the AS/400 System i, so that system is completely **OUT OF PCI SCOPE**!



The Shopping Cart now generates a conclusion page to the visitor, based on the results it received from the C3 Portal.

The PAY screen from the portal is controlled by an INCLUDE header file, an INCLUDE footer file, and a CSS file. Limited graphics are possible. This is specific to the FULL PAGE REDIRECTION only. Header.html and Footer.html do not apply to iFrames.



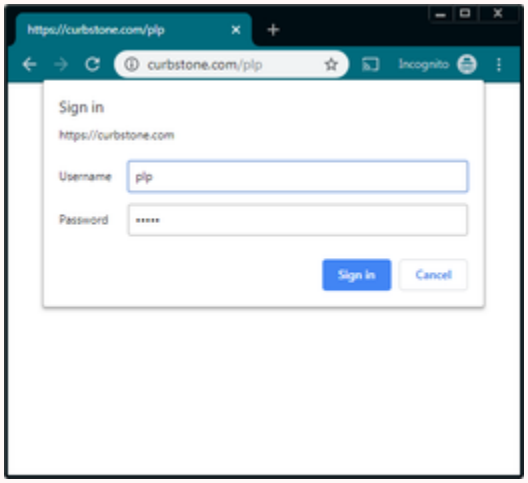
4.1.2. Actual Demo Screens

This is the current live demo we run from

https://curbstone.com/plp

ID and PASSWORD REQUIRED FOR DEMO
This demo requires an ID and password to execute. You will see a prompt in your browser similar to this:

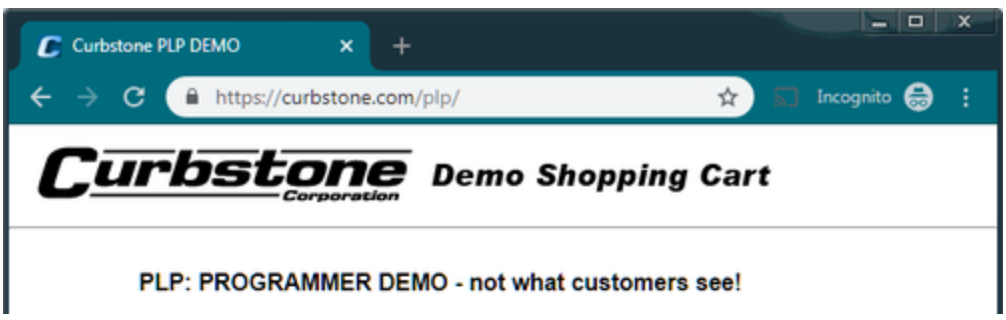
or

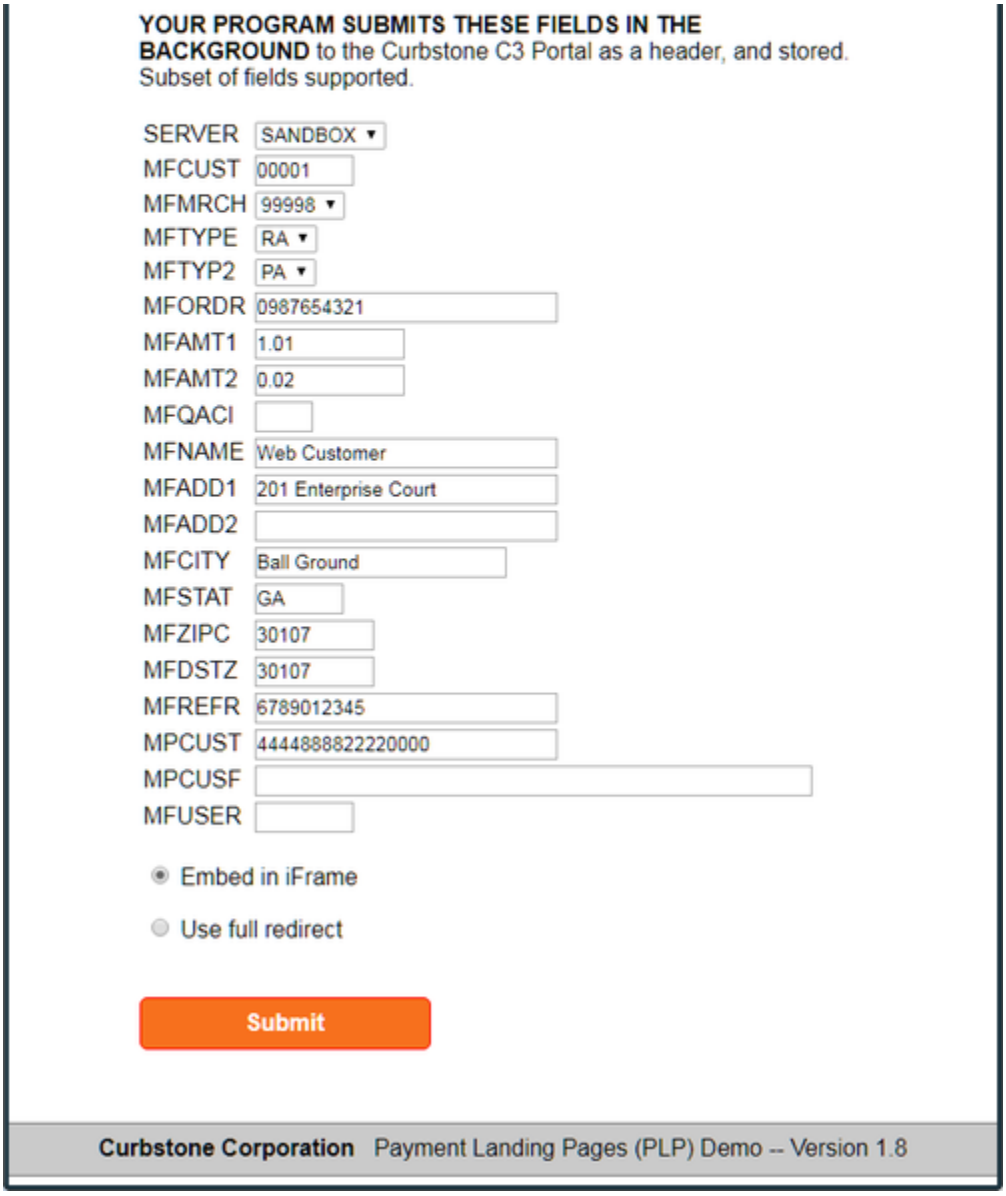



```
==================================

User ID...: plp
Password..: c3plp

==================================
```

The first screen represents what would be provided by the Shopping Cart to the C3 Portal to compose the authorization request. In this demo, we default the values to dummy data for demo purposes. Note the selection here for iFrame or Full Redirect that would normally be done with the Shopping Cart program, but has been brought to the UI to demonstrate both methods.

The demo can be run without any input or changes to the fields. Just click SUBMIT.





The first field will determine if the txn (transaction) is going to our live production servers or to the SANDBOX. We strongly suggest you work against the SANDBOX as much as possible. Be aware, if you are also testing DSI (Data Structure Interface), the txns you process in PLP and use referentially in DSI must be on the same system, **LIVE or SANDBOX**!

COMMON ISSUE TO AVOID
Note that the demo code defaults to the SANDBOX servers at Curbstone Portal. If you are processing PLP transactions and also want to use IPT, DSI, or any other method to access these transactions or the cards used for them, for further processing, that **MUST ALSO BE DONE FROM THE SANDBOX**! **The SANDBOX transactions are not visible to the production Portal!**

The field **MFMRCH** (Curbstone Merchant Code as a child of Curbstone Customer Code MFCUST) has three options, 99998, 97777, and a live MFMRCH.

1. The first option, 99998, which we prefer you use, tells the engine to process the request in our "simulator" which does not go outside the Portal to generate a dummy response. The response is faked depending on the value of the amount, with an amount less than $1, less than $2, etc. generating different combinations of authorizations, AVS, and CVV responses.
2. The second option, 97777, goes to a network test server provided from a major authorization network. The response returned is faked, also based on the dollar amount. The values below 1.00 are used to force various combinations of responses. Inquire of your Project Manager if you want more detail.
3. The third option is to use a LIVE MFMRCH that is assigned to you by your Project Manager when they configure the live accounts. This is typically 00001 or similar, and will NOT start with '9'. If you use a code that does not start with 9, assume it WILL be processed live.

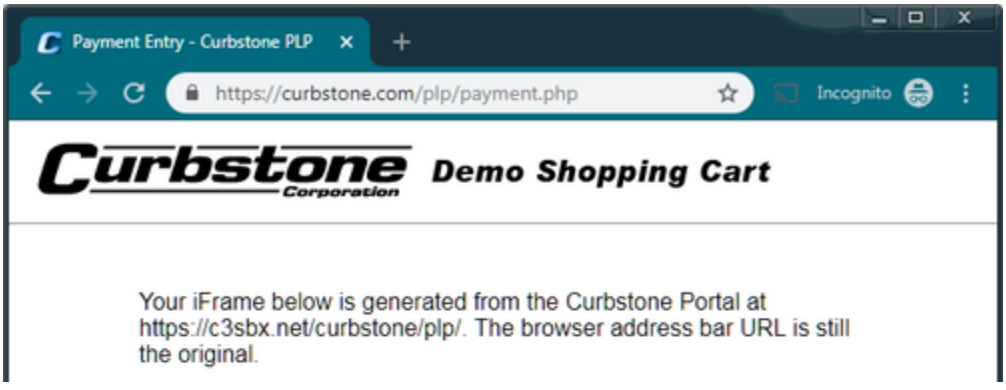
Session vs. Transaction TOKEN
SESSION ID DISPLAYED IN PAYMENT SCREEN

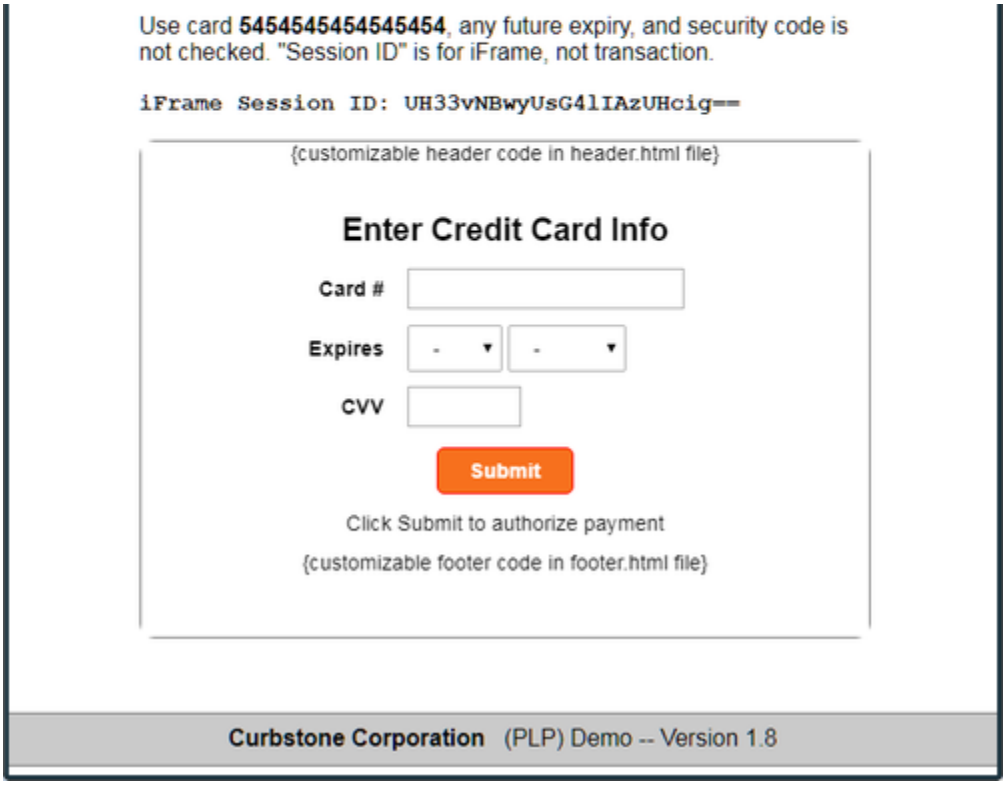
Note that the unique "Session ID" has been displayed (**for demo purposes only**) to identify the instance of the iFrame. This "**Session ID**" is NOT the "**Transaction Token**" (Data Structure field **`MFUKEY`**) that C3 provide for the authorization attempt.

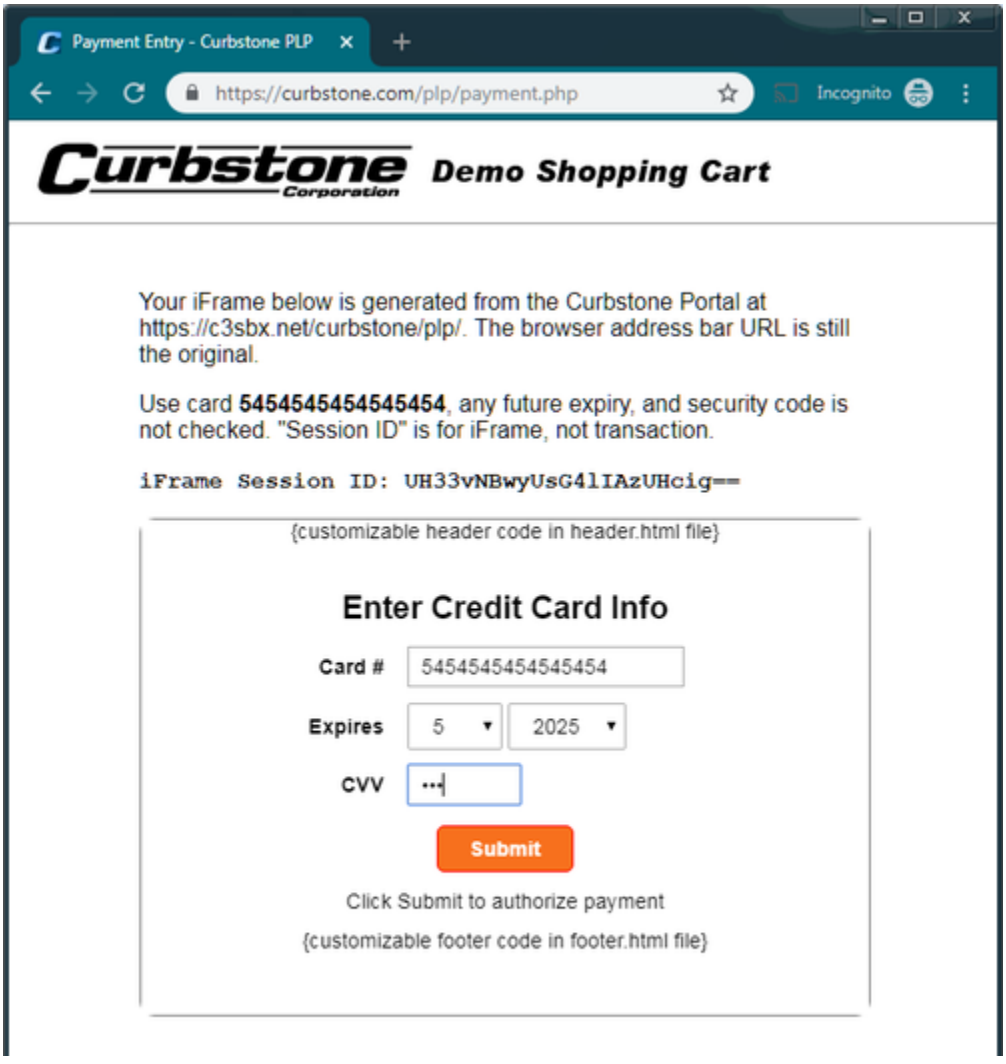
- The "**Session ID**" is truly a unique field that is used only once for the duration of the presentation of the payment Frame or page. This token is transparent to the user and your app and is a critical, but internal field.
- The "**Transaction Token**" is returned in the Data Structure and displayed in the result screen below in the **MFUKEY** field. That is the "remote" token that is retained by the Shopping Cart to identify the authorization when the Order info is sent to the IBM System 'i'. It can be used to REFER to a card on file in a "referential-style" transaction by placing it in the MFKEYP field. The format for this token **MFUKEY** includes your customer number in the first 5 positions, followed by 'Y' to indicate the transaction was generated via PLP, followed by a number that is sequentially generated by the C3 software (e.g.: 00001Y000002345).

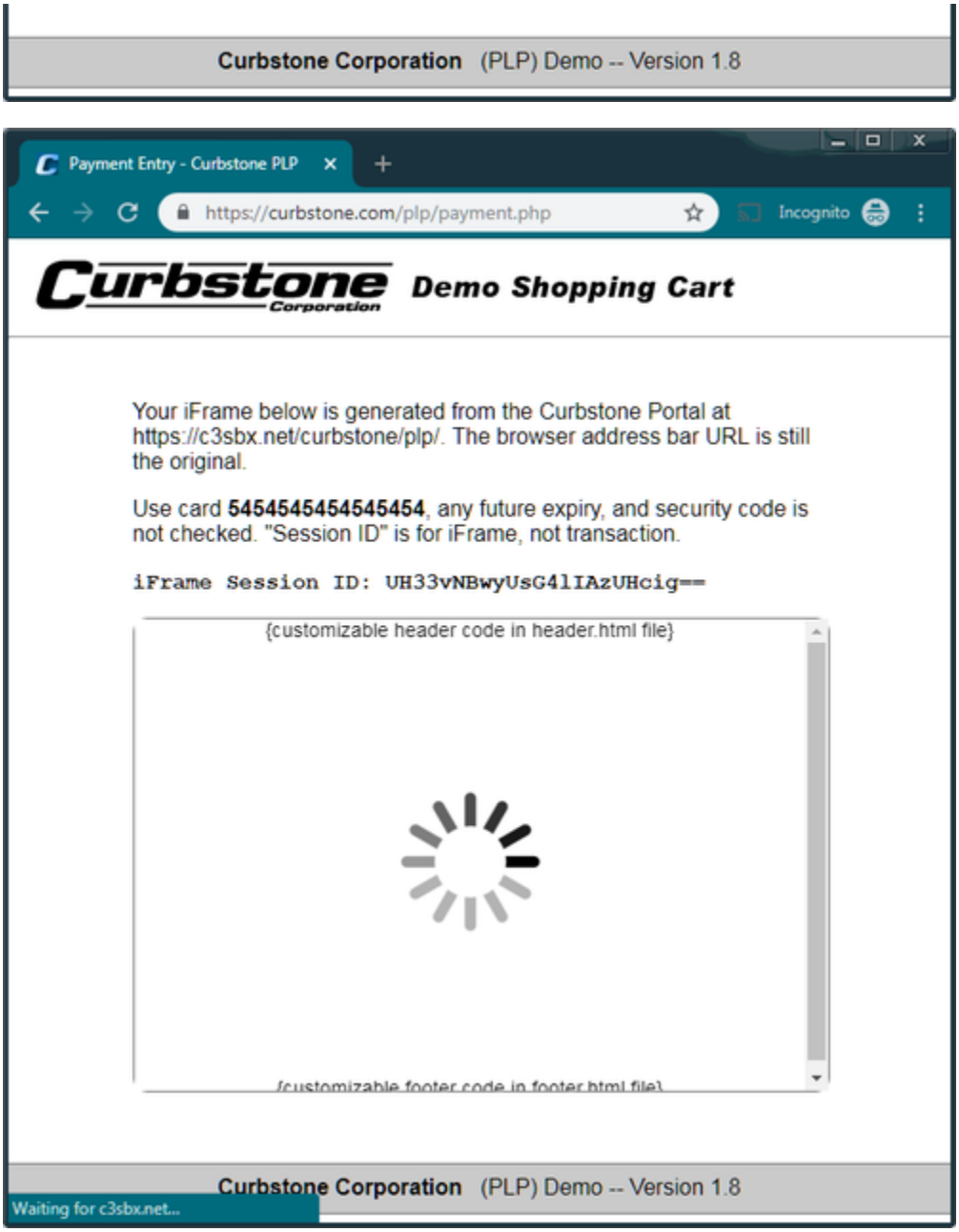
This MFUKEY Remote Token can also be used to store cards on file for later use. Curbstone has a specific procedure we recommend to store cards-on-file and we strongly recommend that you contact your Curbstone Project Manager to discuss to process. Read about retention periods for the data on the C3 Portal in the Master Reference documentations.

This next screen shows the resulting iFrame embedded in a Shopping Cart page. In this demo, the iFrame is the area inside the rectangle. The end-customer keys their card data there and clicks the [PAY] button.

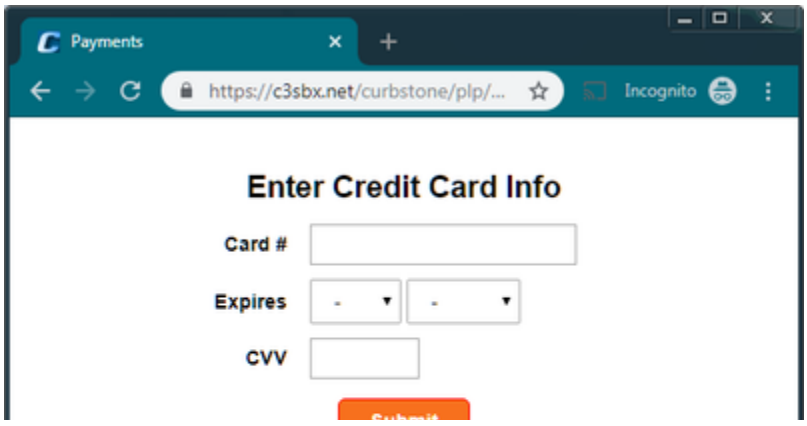


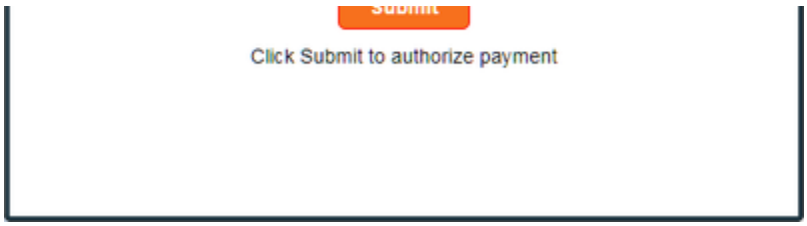

Use card 5454545454545454, any future expiry, and security code is not checked. "Session ID" is for iFrame, not transaction.
iFrame Session ID: UH33vNBwyUsG4lIAzUHcig==
{customizable header code in header.html file}
Enter Credit Card Info
Card #
Expires
CVV
Submit
Click Submit to authorize payment
{customizable footer code in footer.html file}
Curbstone Corporation (PLP) Demo -- Version 1.8


Payment Entry - Curbstone PLP
https://curbstone.com/plp/payment.php
Incognito
Curbstone Corporation
Demo Shopping Cart
Your iFrame below is generated from the Curbstone Portal at https://c3sbx.net/curbstone/plp/. The browser address bar URL is still the original.
Use card 5454545454545454, any future expiry, and security code is not checked. "Session ID" is for iFrame, not transaction.
iFrame Session ID: UH33vNBwyUsG4lIAzUHcig==
{customizable header code in header.html file}
Enter Credit Card Info
Card # 5454545454545454
Expires 5 2025
CVV
Submit
Click Submit to authorize payment
{customizable footer code in footer.html file}



If the option for the "Use full redirect" is chosen programmatically, the payment page will look like this, and not use an iFrame:





Note the full redirect screen in this demo does not display a header.html or footer.html that may be provided by you to customize the look of the screen.

The result is an authorization attempt, and all the fields of the standard C3 Data Structure are returned to the Shopping Cart to evaluate and present a result page to the end-user. In this demo, we have just listed all of the fields of the Data Structure so you can see everything that is returned. In this demo with a bogus card number, the field with the credit card number caused the MFRTRN of 'UL' which is a "local error" code. The cause of the "local error" is shown in the MFATAL field, and refers to the MFCARD field, aka "H".

Also note that the expiration date is returned in the case of a valid authorization so the last 4 digits of the card (MFCARD field), the expiry (MFEDAT), and Token (MFUKEY) can be retained together to prompt for cards on file later. This screen indicates a successful authorization to a test network.

5. Step 1 - Initialize a transaction session

Prior to displaying a PLP iFrame, the Cart must initialize a transaction session by sending a POST request containing all non-card data to the PLP endpoint. (Note that all non-card data must be collected by the Cart prior to this point.) In this example code, the site and page

*https://curbstone.com/plp/**return_url.php***

is used to represent a custom page on the Cart server that is the recipient of the results of the credit card action. Replace with the page of your choice.

API Endpoint
The live API endpoint url is

https://c3plp.net/curbstone/plp/

and is used for live operation. The SANDBOX URL is

https://c3sbx.net/curbstone/plp/

Be aware, your real MFCUST Customer Number will not work on the SANDBOX. If you are using other APIs, like the RPG Client or the DSI Data Structure Interface, the txns must all be performed on the same URL if you are using referentials. Txns on the SANDBOX are not available LIVE.

5.1. Example PLP Source Code Fragment in PHP

Source Code Availability
All source code is available in a ZIP file for download. Visit the support site **https://Curb911.com** and look in the Documentation listing.

Example Code FRAGMENT to Initialize

```
//  ...
//  prior code sets up payload

// Open a connection with the PLP endpoint
$ch = curl_init();

// Configure the POST request
curl_setopt($ch, CURLOPT_URL, $PLP_SVR_URL . '?action=init');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $payload_string);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // Remove line or
change value to 'true' when running in production

// Execute the POST and capture the response
$result = curl_exec($ch);
```

```
if ($result == false) {
    echo '<font face=courier>PLP: Curl error#..: ' . curl_errno($ch) .
"<br>";
    echo 'PLP: Curl error...: ' . curl_error($ch) . "<br>";
    echo 'PLP: Curl target..: ' . $PLP_SVR_URL . "<br></font>";
}

// Close the connection
curl_close($ch);

$result = json_decode($result, true);

// Demonstrates basic success/error checking
switch ($result['MFRTRN']) {
    case 'UG':
        if (!array_key_exists('MFSESS', $result)) {
            die('Transaction session ID was not returned');
        }
        break;

    case 'UL':
    default:
        if (is_array($result)) {
            // Single-character error code
            if (array_key_exists('MFATAL', $result)) {
                echo 'MFATAL ERROR CODE: ' . $result['MFATAL'] . '<br>';
            }
            // Human-readable error message
            if (array_key_exists('MFRTXT', $result)) {
                echo 'MFATAL ERROR MESSAGE: ' . $result['MFRTXT'] .
'<br>';
            }
        }

        die('Transaction initialization failed');
        exit; }

// Extract the transaction session ID from the JSON payload
$SESSION_ID = $result['MFSESS'];

// Build the URL to point to the PLP user interface.
// This will be embedded in the HTML of the web page.
$PLP_TXN_URL = $PLP_SVR_URL . '?MFSESS=' . $SESSION_ID;

//  following code builds page with iFrame
//  ...
```

6. Step 2 - Display the iFrame

Using the returned transaction Session ID, the Cart must render a web page that includes the following required elements:

1) Cart must embed an iFrame somewhere in the page that points to the PLP API URL. The PLP API URL should be composed as follows:

https://c3plp.net/curbstone/plp/?MFSESS=*<session ID retrieved during Step 1>***&mode=embedded**

6.1. The MODE Parameter

6.1.1. Embedded Mode for iFrame

The *mode* parameter should ALWAYS be set to "embedded" if the merchant is embedding PLP within their own site using an iFrame. If this parameter is omitted, then PLP assumes *mode=embedded*.

6.1.2. Redirect Mode for C3 Payment Page

The alternative to ***mode=embedded*** is ***mode=redirect***, which should only be used if the merchant does not wish to embed PLP in an iFrame. This will direct the end-user away from the merchant site to the Curbstone Portal PLP site (`https://c3plp.net`) to complete the payment. Redirect mode should only be used by merchants who explicitly want this behavior.

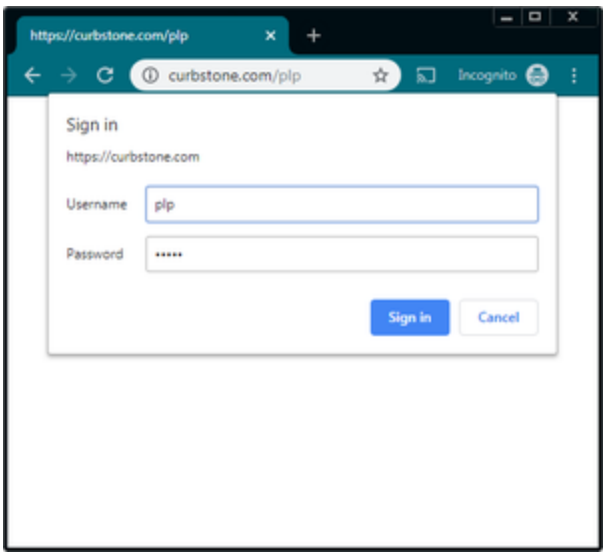
The other scenario in which it is used is for users that have JavaScript disabled. As seen in the "Example PLP iFrame Source Code" snippet below, there is a <noscript> element included that contains a link to PLP, which includes a parameter of *mode=redirect*. **Redirect mode should only be used when the merchant specifically wants this behavior, or when an end-user has JavaScript disabled.**

> **iFrame Hidden by Default**
> The iFrame should be hidden by default (whether via stylesheet or inline styles) in order to ensure that only JavaScript-enabled users can use it for payment. This is necessary because it is not possible to break out of the iFrame upon completion of the transaction if JavaScript is disabled. The PLP JavaScript library (mentioned below) will make the iFrame visible for JavaScript-enabled users and will enable redirection to the target URL ("MPTRGT") upon completion of the transaction.

6.2. PLP DEMO Example Source Code

> **Source Code Availability**
> All source code is ALSO available in a ZIP file for download. Visit the support site `https://Curb911.com` and look in the Documentation listing.

Visit the Curbstone PLP Demo:
`https://curbstone.com/plp`



The ID and password are
`plp/c3plp`

Example PLP iFrame Source Code - index.php - front end

```
<?php
/**
 * Curbstone CorrectConnet PLP DEMO site example source code - index.php
 */

// set the displayed version number for this file
$pagevers = '1.7o';

// Base Path
if (!defined('CLIENT_BASE_PATH')) { define('CLIENT_BASE_PATH', dirname
(__FILE__) . DIRECTORY_SEPARATOR); }

// Determine the include path based on the master config file
$CONFIG = parse_ini_file(CLIENT_BASE_PATH . 'config.ini');

// Specify the url of the credit card payment page
$PAYMENT_URL = $CONFIG['base_url'] . 'payment.php';

// Render the page
echo <<<OUT
<!DOCTYPE html>
<html>
    <head>
        <title>Curbstone PLP DEMO</title>
        <meta charset="UTF-8" />
        <base href="{$CONFIG['base_url']}">
        <link rel="stylesheet" href="styles/qlient.css" />
    </head>
```

```
    <body>

        <a href="{$CONFIG['base_url']}"><img src="Curbstone-plp-demo_01.
png"></a>
        <hr>
        <div id="main">
            <div id="index-container">

<strong>PLP: PROGRAMMER DEMO - not what customers see!<br /><br />
YOUR PROGRAM SUBMITS THESE FIELDS IN THE BACKGROUND</strong>
to the Curbstone C3 Portal as a header, and stored. Subset of fields
supported.
<br /><br />

<form method="post" action="$PAYMENT_URL">
    <table>
        <tr><td>SERVER</td>
            <td><select name="SVRTGT">
                    <option>SANDBOX</option>
                    <option>LIVE</option>
                </select></td></tr>
        <tr><td>MFCUST</td>
            <td><input type="text" name="MFCUST" size="5" maxlength="5"
value="00001" /></td></tr>
        <tr><td>MFMRCH</td>
            <td><select name="MFMRCH">
                    <option>99998</option>
                    <option>97777</option>
                </select></td></tr>
        <tr><td>MFTYPE</td>
            <td><select name="MFTYPE">
                    <option>RA</option>
                    <option>RY</option>
                </select></td></tr>
        <tr><td>MFTYP2</td>
            <td><select name="MFTYP2">
                    <option>PA</option>
                    <option>SA</option>
                </select></td></tr>
        <tr><td>MFORDR</td>
            <td><input type="text" name="MFORDR" size="25" maxlength="
20" value="0987654321" /></td></tr>
        <tr><td>MFAMT1</td>
            <td><input type="text" name="MFAMT1" size="10" maxlength="
8" value="1.01" /></td></tr>
        <tr><td>MFAMT2</td>
            <td><input type="text" name="MFAMT2" size="10" maxlength="
8" value="0.02" /></td></tr>
        <tr><td>MFNAME</td>
            <td><input type="text" name="MFNAME" size="25" maxlength="
```

```
20" value="Web Customer" /></td></tr>
        <tr><td>MFADD1</td>
            <td><input type="text" name="MFADD1" size="25" maxlength="
20" value="201 Enterprise Court" /></td></tr>
        <tr><td>MFADD2</td>
            <td><input type="text" name="MFADD2" size="25" maxlength="
20" value="" /></td></tr>
        <tr><td>MFCITY</td>
            <td><input type="text" name="MFCITY" size="20" maxlength="
15" value="Ball Ground" /></td></tr>
        <tr><td>MFSTAT</td>
            <td><input type="text" name="MFSTAT" size="4" maxlength="2"
value="GA" /></td></tr>
        <tr><td>MFZIPC</td>
            <td><input type="text" name="MFZIPC" size="7" maxlength="5"
value="30107" /></td></tr>
        <tr><td>MFDSTZ</td>
            <td><input type="text" name="MFDSTZ" size="7" maxlength="5"
value="30107" /></td></tr>
        <tr><td>MFREFR</td>
            <td><input type="text" name="MFREFR" size="25" maxlength="
20" value="6789012345" /></td></tr>
        <tr><td>MPCUST</td>
            <td><input type="text" name="MPCUST" size="25" maxlength="
20" value="4444888822220000" /></td></tr>
        <tr><td>MPCUSF</td>
            <td><input type="text" name="MPCUSF" size="50" maxlength="
500" value="" /></td></tr>
                <tr><td>MFUSER</td>
            <td><input type="text" name="MFUSER" size="5" maxlength="
10" value="" /></td></tr>
        <tr><td colspan="2">
            <p><input id="mode-embed" type="radio" name="mode" value="
embed" checked="checked" />
               <label for="mode-embed">Embed in iFrame</label></p>
            <p><input id="mode-redirect" type="radio" name="mode"
value="redirect" />
               <label for="mode-redirect">Use full redirect</label></p><
/td></tr>
    </table>
    <br />
    <div class="submit-button">
        <button type="submit">Submit</button>
    </div>

</form>
</div>
</div><!-- #main -->
<br class="clear" />
```

```
<footer>
   <strong>Curbstone Corporation</strong>   Payment Landing Pages (PLP)
Demo -- Version $pagevers
</footer>

</body>
</html>
OUT;
```

Example DEMO PLP Source Code - payment.php - back end

```
<?php
/**
 * Curbstone CorrectConnet PLP DEMO site example source code - payment.
php
 **/

// set the displayed version number for this file
$pagevers = '1.8';

// Base Path
if (!defined('CLIENT_BASE_PATH')) { define('CLIENT_BASE_PATH', dirname
(__FILE__) . DIRECTORY_SEPARATOR); }

// Determine the include path based on the master config file
$CONFIG = parse_ini_file(CLIENT_BASE_PATH . 'config.ini');

// Specify the url of the ***LIVE*** PLP initialization endpoint
$PLP_API_URL = $CONFIG['plp_api_url'];
// Specify the url of the ***SANDBOX*** PLP initialization endpoint
$PLP_SBX_URL = $CONFIG['plp_sbx_url'];

// Grab the choice of target server from the entry screen
$SVRTGT = $_POST['SVRTGT'];  // LIVE or SANDBOX, your choice!

if ($SVRTGT == 'SANDBOX') {
$PLP_SVR_URL = $PLP_SBX_URL;
} else {
$PLP_SVR_URL = $PLP_API_URL;
}

// Setup the variables for all of the fields in the Curbstone Data
Structure
// This is typically how the data is constructed for the PLP API
// The example program index.php is just a front end to demo the API
    $MFADD1 = $_POST['MFADD1'];  // AVS
    $MFADD2 = addslashes($_POST['MFADD2']);
    $MFAMT1 = $_POST['MFAMT1'];  // purchase amount
```

```
    $MFCITY = $_POST['MFCITY'];
    $MFCUST = $_POST['MFCUST'];
    $MFMETH = '02';  // ecommerce must be 02
    $MFMRCH = $_POST['MFMRCH'];
    $MFQACI = $_POST['MFQACI'];
    $MFNAME = $_POST['MFNAME'];
    $MFREFR = $_POST['MFREFR'];  // invoice number
    $MFSTAT = $_POST['MFSTAT'];
    $MFTYPE = $_POST['MFTYPE'];
    $MFTYP2 = $_POST['MFTYP2'];
    // Level II fields start
    $MFAMT2 = $_POST['MFAMT2'];  // tax amount
    $MFORDR = $_POST['MFORDR'];  // customer PO number
    $MFDSTZ = $_POST['MFDSTZ'];  // destination zip code
    $MFLTXF = '1';               // tax flag
    // Level II fields end

// this is our chance to add some background data programmatically
///////
    // start of user defined fields
    $MFUSD1 = '1';
    $MFUSD2 = '2';
    $MFUSD3 = '3';
    $MFUSD4 = '44';
    $MFUSD5 = '5555555';
    $MFUSD6 = '6666666666';
    $MFUSD7 = '7777777777777777';
    $MFUSD8 = '8888888888888888';
    $MFUSDA = 123456789012345;
    $MFUSDB = 123456789012345;
    $MFUSDC = 123456789012.34;
    // end of user defined fields
    $MFUSER = $_POST['MFUSER'];
    $MFZIPC = $_POST['MFZIPC'];  // AVS
    // this puts the invoice number in the URL to be visible for diags
    $MPCUSF = 'mfrefr=' . $MFREFR . '&mftype=' . $MFTYPE . $MFTYP2 .
'&key3=val3&key4=val4';
    $MPCUSF = $_POST['MPCUSF'];  // Custom query string. Max 300 chars
    $MPCUST = $_POST['MPCUST'];
    $MPTRGT = $CONFIG['target_url'];

// Specify all non-card data that will be transmitted from the client's
server.
// Also, specify a target URL for PLP to return the results of the
transaction to the client?s server.
$payload = array(
    'MFADD1' => $MFADD1,
    'MFADD2' => $MFADD2,
    'MFAMT1' => $MFAMT1,
    'MFAMT2' => $MFAMT2,
```

```
    'MFCITY' => $MFCITY,
    'MFDSTZ' => $MFDSTZ,
    'MFLTXF' => $MFLTXF,
    'MFMETH' => $MFMETH,
    'MFMRCH' => $MFMRCH,
    'MFNAME' => $MFNAME,
    'MFORDR' => $MFORDR,
    'MFQACI' => $MFQACI,
    'MFREFR' => $MFREFR,
    'MFSTAT' => $MFSTAT,
    'MFTYP2' => $MFTYP2,
    'MFTYPE' => $MFTYPE,
    'MFUSD1' => $MFUSD1,
    'MFUSD2' => $MFUSD2,
    'MFUSD3' => $MFUSD3,
    'MFUSD4' => $MFUSD4,
    'MFUSD5' => $MFUSD5,
    'MFUSD6' => $MFUSD6,
    'MFUSD7' => $MFUSD7,
    'MFUSD8' => $MFUSD8,
    'MFUSDA' => $MFUSDA,
    'MFUSDB' => $MFUSDB,
    'MFUSDC' => $MFUSDC,
    'MFUSER' => $MFUSER,
    'MFZIPC' => $MFZIPC,
    'MPCUSF' => $MPCUSF,
    'MPCUST' => $MPCUST,
    'MPTRGT' => $MPTRGT,
        'MFCUST' => $MFCUST
);

// Format the payload for transmission
$payload_string = '';
foreach ($payload as $key => $value) {
    $payload_string .= $key . '=' . urlencode($value) . '&';
}
rtrim($payload_string, '&');

// Open a connection with the PLP endpoint
$ch = curl_init();

// Configure the POST request
curl_setopt($ch, CURLOPT_URL, $PLP_SVR_URL . '?action=init');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $payload_string);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // Remove line or
change value to 'true' when running in production

// Execute the POST and capture the response
```

```
$result = curl_exec($ch);

if ($result == false) {
    echo '<font face=courier>PLP: Curl error#..: ' . curl_errno($ch) .
"<br>";
    echo 'PLP: Curl error...: ' . curl_error($ch) . "<br>";
    echo 'PLP: Curl target..: ' . $PLP_SVR_URL . "<br></font>";
}

// Close the connection
curl_close($ch);

$result = json_decode($result, true);

// Demonstrates basic success/error checking
switch ($result['MFRTRN']) {
    case 'UG':
        if (!array_key_exists('MFSESS', $result)) {
            die('Transaction session ID was not returned');
        }
        break;

    case 'UL':
    default:
        if (is_array($result)) {
            // Single-character error code
            if (array_key_exists('MFATAL', $result)) {
                echo 'MFATAL ERROR CODE: ' . $result['MFATAL'] . '<br>';
            }
            // Human-readable error message
            if (array_key_exists('MFRTXT', $result)) {
                echo 'MFATAL ERROR MESSAGE: ' . $result['MFRTXT'] .
'<br>';
            }
        }

        die('Transaction initialization failed');
        exit; }

// Extract the transaction session ID from the JSON payload
$SESSION_ID = $result['MFSESS'];

// Build the URL to point to the PLP user interface.
// This will be embedded in the HTML of the web page.
$PLP_TXN_URL = $PLP_SVR_URL . '?MFSESS=' . $SESSION_ID;

// Redirect instead of rendering the page if operating in "non-iframe"
mode
if (array_key_exists('mode', $_POST) && $_POST['mode'] == 'redirect') {
    header('Location: ' . $PLP_TXN_URL . '&mode=redirect');
```

```
    exit; }

// Render the payment page.
// Most of the following HTML is completely arbitrary and can be
customized as needed.
// The only required elements are the iFrame and PLP JavaScript library.
echo <<<OUT
<!DOCTYPE html>
<html>
    <head>
        <title>Payment Entry - Curbstone PLP </title>
        <meta charset="UTF-8" />
        <base href="{$CONFIG['base_url']}">
        <link rel="stylesheet" href="styles/qlient.css" />
    </head>
    <body>

        <a href="{$CONFIG['base_url']}"><img src="Curbstone-plp-demo_01.
png"></a>
        <hr>
        <div id="main">

            <div id="payment-container">

<p>Your iFrame below is generated from the Curbstone Portal at
$PLP_SVR_URL.  The browser address bar URL is still the original.</p>

<p>Use card <strong>5454545454545454</strong>, any future expiry,
and security code is not checked.
"Session ID" is for iFrame, not transaction.</p>

                <!-- Only displayed to easily reference the transaction
session ID in use -->
                <!-- Typically would not be displayed to the end-user
cardholder              -->
                <p><strong><font face="courier, monospace">iFrame
Session ID: $SESSION_ID</font></strong></p>

                <!-- REQUIRED -->
                <!-- iFrame MUST include an id of "plp-iframe" -->
                <iframe id="plp-iframe" style="display:none;" src="
{$PLP_TXN_URL}&mode=embedded"></iframe>


                <!-- OPTIONAL -->
                <!-- The noscript tag allows non-JavaScript users to
proceed with the payment -->
                <!-- This will display in place of the iFrame if the
user has JavaScript disabled -->
                <noscript>
```

```
                    <p style="display:block;border:1px solid red;
background-color:pink;font-weight:bold;padding:20px;">
                        JavaScript not enabled. Go to our <a href="
{$PLP_TXN_URL}&mode=redirect">payment portal</a> to complete your order.
                    </p>
                </noscript>

            </div>

            <!-- REQUIRED -->
            <!-- This JavaScript library is responsible for: -->
            <!--    1) Making the iFrame visible -->
            <!--    2) Breaking out of iFrame, redirect to the target
page (MPTRGT) on completion -->
            <script type="text/javascript" src="{$PLP_SVR_URL}scripts
/plp.js"></script>

        </div>

        <br class="clear" />

        <footer>
            <strong>Curbstone Corporation</strong>   (PLP) Demo --
Version $pagevers
        </footer>

    </body>
</html>
OUT;

?>
```

Example DEMO Source Code - return_url.php recipient

```
<?php

// set the displayed version number for this file
$pagevers = '1.7o';

// Base Path
if (!defined('CLIENT_BASE_PATH')) { define('CLIENT_BASE_PATH', dirname
(__FILE__) . DIRECTORY_SEPARATOR); }

// Determine the include path based on the master config file
$CONFIG = parse_ini_file(CLIENT_BASE_PATH . 'config.ini');

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $response = $_POST;
```

```
} else {
    $response = $_GET;
}
ksort($response);

if (array_key_exists('MFRTRN', $response)) {
    // Determine if the transaction was successful
    switch ($response['MFRTRN']) {
        case 'UG':
            $confirmation_msg = 'Authorization successful. Thank you!';
            $txn_status = 'success';
            break;
        case 'UN':
            $confirmation_msg = 'Authorization declined: MFRTXT: ' .
$response['MFRTXT'];
            $txn_status = 'error';
            break;
        case 'UL':
        default:
            $confirmation_msg = 'Field Error - code: MFATAL: ' .
$response['MFATAL'];
            $txn_status = 'error';
            break;
    }
} else {
    $confirmation_msg = '';
    $txn_status = '';
}

$response_fields = '<font face="monospace">';
foreach ($response as $key => $value) {
    $response_fields .= htmlentities($key, ENT_QUOTES) . ' => ' .
htmlentities($value, ENT_QUOTES) . '<br> ';
}
$response_fields .= '</font>';

echo <<<OUT
<!DOCTYPE html>
<html>
<head>
    <title>Payment Complete - Curbstone PLP </title>
    <meta charset="UTF-8" />
    <base href="{$CONFIG['base_url']}">
    <link rel="stylesheet" href="styles/qlient.css" />
</head>
<body>
<a href="{$CONFIG['base_url']}"><img src="Curbstone-plp-demo_01.png"><
/a>
<hr>
<div id="main">
```

```
    <div id="results-container">
        <a class="submit-button" href="{$CONFIG['base_url']}">Submit
Another</a>
        <h3 class="$txn_status">$confirmation_msg</h3>

        $response_fields

<p><b>MFUKEY</b> is the transaction <b>TOKEN</b> to store
with order for later settlement.</p>

        <a class="submit-button" href="{$CONFIG['base_url']}">Submit
Another</a>

    </div>
</div>
<footer>
    <strong>Curbstone Corporation</strong>   Payment Landing Pages
(PLP) Demo -- Version $pagevers
</footer>
</body>
</html>
OUT;
```

Example DEMO Source Code - config.ini

```
; Curbstone CorrectConnect
; Payment Landing Pages "PLP" Client Configuration file
; This is demo code for the reference PLP CLIENT SIDE code
; https://curb911.com   Support Site
; 888-844-8533 24 hour support
ConfigVersion=1.7o

; Client base url
; This must be the working folder for the shopping cart
; from which the included "index.php" is executed.
; Trailing/ slash/ is/ required/
;
; SHOPPING CART URL
base_url = https://curbstone.com/plp/

;;;;;;;;;;;;;;;;  WE WILL USE ONE OF THE TWO BELOW  ;;;;;;;;;;;;;;;;
;
; URL pointing to LIVE SERVER PLP installation
; C3 **LIVE** PORTAL URL
plp_api_url = https://c3plp.net/curbstone/plp/
```

```
; URL pointing to ***SANDBOX*** PLP installation
; C3 SANDBOX PORTAL URL
plp_sbx_url = https://c3sbx.net/curbstone/plp/
;
;;;;;;;;;;;;;;;;  WE WILL USE ONE OF THE TWO ABOVE  ;;;;;;;;;;;;;;;;

; URL to which control is returned - this must be a program
; that can accept the POSTed data from the C3 portal and
; parse it.  Change this to match the program in YOUR Shopping
; Cart that will receive the response from the C3 server as to
; the success or failure of the authorization/storage request.
; This must be a fully-qualified URL that includes the filename
; of the receiving program, unless it is "index.php", in which
; case it must end/ with/ a/ trailing/ slash/!
;
; SHOPPING CART URL
target_url = https://curbstone.com/plp/return_url.php
;
; end of file
```

The dimensions and location of the iFrame can be styled by the Cart. However, custom styling of the internal contents of the iFrame requires storing additional assets on the PLP server. (Please contact Curbstone support if you would like to apply custom styling to the contents of the iFrame.)

The <noscript> element is not strictly required, though having it present helps give non-JavaScript users a pathway for continuing with the payment. Without it, non-JavaScript users will not be able to proceed with payment.

2) Cart must include the PLP JavaScript library by embedding a <script> element. This library is used to break out of the iFrame and redirect to the target URL ("MPTRGT") upon completion of the transaction.

6.3. Example Required JavaScript for PLP

Example JavaScript for PLP

```
<!-- REQUIRED -->
<!-- This JavaScript library is responsible for: -->
<!--    1) Making the iFrame visible -->
<!--    2) Breaking out of iFrame, redirect to the target page (MPTRGT)
on completion -->
    <script type="text/javascript" src="{$PLP_API_URL}scripts/plp.js"><
/script>
<!-- // end of code example -->
```

6.4. Example JavaScript for responding to PLP events in an iFrame

This section refers to enhanced functions when using an iFrame to contain the PLP page.

The cart can optionally define its own handlers to respond to events emitted from within the PLP iFrame. If included, such custom handlers must be defined **after inclusion** of the PLP JavaScript library.

Example JavaScript for responding to PLP events

```
    <script type="text/javascript">

                // Executes each time the card entry form loads.
        PLP.onFormLoad(function () {
                        // Add custom JavaScript here
        });

        // Executes each time the card entry form is submitted.
        PLP.onFormSubmit(function () {
                         // Add custom JavaScript here
        });

        // Executes whenever a validation error preempts gateway
authorization.
        PLP.onFormValidationError(function (data) {

            // If desired, do something with the validation errors.
            var errors = data.errors.split('|');
            console.log('Validation errors occurred: ', errors);
        });

        // Executes if an error is thrown following final authorization.
        // Note, this only executes if an error is thrown following
final authorization.
        // It does not execute if the card is declined or if C3 returns
a recognized local error.
        PLP.onTransactionError(function (data) {

            // If desired, do something with the transaction errors.
            var errors = data.errors.split('|');
            console.log('Transaction errors occurred: ', errors);
        });

                // Executes whenever PLP returns a 'service
unavailable' error
        PLP.onServiceUnavailable(function () {
            // Add custom JavaScript here
        });

        // Executes following completion of the transaction, whether
auth was successful or not.
        // PLP.onTransactionCompletion(function () {
        //
```

```
        //      WARNING!    WARNING!    WARNING!    WARNING!
WARNING!    WARNING!
        //
        //      Setting this handler disables the default transaction
completion handling mechanism.
        //      This means that the iFrame will not be broken out of
automatically.
        //
        //      Setting this handler disables the default transaction
completion handling mechanism.
        //      This means that the iFrame will not be broken out of
automatically.
        //      So, the cart **MUST** define its own JavaScript to
handle transaction completion.
        //      ONLY DO THIS WITH GOOD REASON, AND IF YOU KNOW WHAT YOU
ARE DOING!
        //      ONLY DO THIS WITH GOOD REASON, AND IF YOU KNOW WHAT YOU
ARE DOING!
        //
        //      WARNING!    WARNING!    WARNING!    WARNING!
WARNING!    WARNING!
        //
        // });

        </script>
<!-- // end of code example -->
```

7. Step 3 - User enters card data and submits transaction

The end user enters all card data into the PLP user interface via the iFrame and submits the transaction. Upon submission, the user interface validates the data entered and will return a message to the user if errors are found. If no errors are found the card data is merged with the data that was transmitted during Step 1. All of this data is then sent to the authorization gateway.

8. Step 4 - PLP redirects to the target URL via POST

Upon response from the authorization gateway, the PLP server triggers the embedded PLP JavaScript library to do the following:

1. Break out of the iFrame
2. POST all transaction result fields to the target URL (MPTRGT) that was specified during Step 1.

Both of these actions are performed by the JavaScript library running on the end-customer browser, so when your Target URL program responds, it goes to the customer browser.

Cart must ensure that the target URL (MPTRGT) is capable of accepting and processing the POST. Example code is provided in our demo, found in the downloadable ZIP file.

9. Modifying the PLP CSS

Certainly you will want to tweak the appearance of the PLP page, whether you embed it as an iFrame or just do a "full redirect". We are glad to accommodate!

The default CSS for the PLP page is embedded in the HTML that we generate from the Portal. This keeps things simple and reduces load. The CSS is as minimal as we can make it for the page. The entire CSS for the page or iFrame (same for both) is embedded in between the <style> and </style> tags. For reference, it follows:

Default PLP CSS

```
<style>
/**
 * v1.2
 * Last Modified: 3/19/2016
 */

body {
    background-color: #fff;
    margin: 0;
    padding: 0;
    font-family:sans-serif;
}

h2 {
    margin: .5rem 0;
}

.plp-error-header {
    display: block;
    width: 100%;
    text-align: center;
    margin-top: 3rem;
    font-weight: normal;
    font-size: 1.2rem;
}

.plp-clear {
    float: none;
    clear: both;
    height: 0;
}

.plp-success,
.plp-error {
    display: block;
    float: left;
    width: 91%;
    border-radius: 5px;
```

```
    padding: .4rem 4% .4rem 4%;
    margin: 0 0 .5rem 0;
}

.plp-success {
    border: 1px solid green;
    background-color: #CCE6CC;
}

.plp-error {
    border: 1px solid red;
    background-color: pink;
}

.plp-payment-form td {
    padding: .3rem .5rem;
}

.plp-payment-form td:first-child {
    text-align: right;
    font-weight: bold;
}

.plp-payment-form input,
.plp-payment-form select {
    padding: .3rem .5rem;
    font-size: 1rem;
}

.plp-payment-form select {
    font-size: 1rem;
    padding: 6px 12px !important;
}

.plp-payment-form button {
    padding: .5rem 1.5rem;
    font-size: 1rem;
    font-weight: bold;
    cursor: pointer;
    border: 1px solid red;
    border-radius: 5px;
    background-color: #F7701D;
    color: #FFF;
}

.plp-payment-form button:hover {
    background-color: #F5D625;
    color: red;
}
```

```
#plp-spinner-wrapper,
#plp-form-wrapper {
    width: 290px;
}

#plp-spinner-wrapper {
    height: 290px;
    margin: 0 auto;
    position: relative;
}

#plp-spinner {
    width: 100px;
    height: 100px;
    position: absolute;
    top: 50%;
    left: 50%;
    margin: -60px 0 0 -50px;
}

#plp-wrapper {
    padding-top: 1rem;
}

#plp-form-wrapper {
    margin: 0 auto;
    padding-top: 10px;
}

#plp-form-wrapper .plp-form-header,
#plp-form-wrapper .plp-form-footer,
#plp-form-wrapper .plp-submit-button {
    text-align: center;
}

#plp-form-wrapper .plp-submit-button {
    padding-top: .5rem;
}

#plp-form-wrapper .plp-form-footer {
    padding: .8rem;
}

</style><style> /* v1.3
 * Last Modified: 5/18/2016
 */

body {
    background-color: #fff;
    margin: 0;
```

```
    padding: 0;
    font-family:sans-serif;
}

h2 {
    margin: 1rem 0;
}

.plp-error-header {
    display: block;
    width: 100%;
    text-align: center;
    margin-top: 3rem;
    font-weight: normal;
    font-size: 1.2rem;
}

.plp-clear {
    float: none;
    clear: both;
    height: 0;
}

.plp-success,
.plp-error {
    display: block;
    float: left;
    width: 91%;
    border-radius: 5px;
    padding: .4rem 4% .4rem 4%;
    margin: 0 0 .5rem 0;
}

.plp-success {
    border: 1px solid green;
    background-color: #CCE6CC;
}

.plp-error {
    border: 1px solid red;
    background-color: pink;
}

.plp-payment-form td {
    padding: .3rem .5rem;
}

.plp-payment-form td:first-child {
    text-align: left;
    font-weight: normal;
```

```
}

.plp-payment-form input,
.plp-payment-form select {
    padding: .3rem .5rem;
    font-size:.8em;
}

.plp-payment-form select {
    font-size: .8em;
    padding: 6px 12px !important;
}

.plp-submit-button {
    text-align: center;
}

.plp-payment-form button {
    padding: .5rem 1rem;
    font-size: .8em;
    font-weight: bold;
    cursor: pointer;
    border: 1px solid red;
    border-radius: 5px;
    background-color: #F7701D;
    color: #FFF;
}

.plp-payment-form button:hover {
    background-color: #F5D625;
    color: red;
}

#plp-spinner-wrapper,
#plp-form-wrapper {
    width: 290px;
}

#plp-spinner-wrapper {
    height: 290px;
    margin: 0 auto;
    position: relative;
}

#plp-spinner {
    width: 100px;
    height: 100px;
    position: absolute;
    top: 50%;
    left: 50%;
```

```
    margin: -60px 0 0 -50px;
}

#plp-wrapper {
    padding-top: 1rem;
}

#plp-form-wrapper {
    margin: 0 auto;
    padding-top: 10px;
}

#plp-form-wrapper .plp-form-header,
#plp-form-wrapper #plp-form-wrapper {
    text-align: left;
}

.plp-form-footer {
        text-align: center;
}

#plp-form-wrapper .plp-submit-button {
    padding-top: .5rem;
}

#plp-form-wrapper .plp-form-footer {
    padding: .8rem;
}
td {
    font-size: small;
}
</style>
```

Conceptually, we are inviting you to submit a small (preferably!) "Custom" <style></style> snippet to us. This will follow OUR default CSS, and any settings that you change will override ours because yours will load last.

We also encourage you to ONLY include in that new "override" list, the specific settings that you require to accomplish your look and feel. Please restrict the entries in your "Custom" styles to only those that you have changed and do not include the <style> tags in the css files.

> **ONLY CHANGES**
> **Only place CHANGED CSS elements in the "Custom" <style> section. Do not copy the entire default CSS.**

9.1. Using the PLP Demo to Work with CSS

1. Visit the Curbstone PLP Demo:
   https://curbstone.com/plp

The ID and password are
`plp/c3plp`
to gain access.

2. Change the radio button at the bottom of the form to "Use full redirect" and submit the form.
3. View the source code of the resulting page. In Chrome, do this by going to **View > Developer > View Source**, or right click and select **View Page Source**. This markup is identical to the markup that displays in the iFrame; it's just not bounded by an iFrame container.
4. Copy the source HTML to a file on your machine for local development and testing of custom CSS additions. The local file can be opened directly in a browser.
5. Locate the **<style></style>** tag that's embedded in the source HTML, but **DON'T change the styles in this tag**. Proceed to the next step.
6. Add your own <style></style> tag immediately **following the end** of the <style></style> that already exists.
7. Put your own custom styles in the new, "Custom" <style></style> tag that you just added.
NOTE: Add new settings to your "Custom" section that override settings that appear earlier in the original <style> section. Keep your new styles segregated and easily identifiable, and keep them in close to the same order as the originals.
8. Refresh your local copy of the page, review, tweak your "Custom" styles, then repeat, until you're happy with how the PLP payment form looks.
9. For additional information on the styles being used and their effect, use the Developers' Tool usually accessed with **<F12>**.
10. Ensure that you test the "Custom" CSS with an error message as well. Just hit Submit on an empty form to get our default error display. Ensure that fits with the changes you have made to the sheet.
11. Copy only the contents of the "Custom" <style></style> section that you created to Curbstone through your Project Manager.
12. Curbstone will load that file on the PLP servers and it will be specific to a particular MFCUST+MFMRCH profile. You must specify those to which it should apply, if you have more than one PLP configuration - multiple MFMRCHs for your MFCUST number.

10. Request Data Fields Definitions

See also documentation file **`C3_Master_Reference_xx.pdf`** on the support site at **`https://curb911.com`**.

All fields included on the request will also be returned on the response.

	Request Field Description	Post Name	Post Type	Len	Req?	Curbstone Type	Stored on Portal
1	Curbstone customer number	`MFCUST`	Text	5	Y	A	Y
2	Curbstone merchant code	`MFMRCH`	Text	5	Y	A	Y
3	Transaction type	`MFTYPE`	Text	2	Y	A	Y
4	Secondary transaction type	`MFTYP2`	Text	2	Y	A	Y

5	Entry method	MFMETH	Text	2	Y	A	Y
6	Order / PO Number	MFORDR	Text	16	Y	A	Y
7	Transaction Amount	MFAMT1	Text	15	Y	N 15,5	Y
8	Customer Name	MFNAME	Text	40	N	A	Y
9	Street Address	MFADD1	Text	24	N	A	Y
10	Street Address 2	MFADD2	Text	24	N	A	Y
11	City	MFCITY	Text	20	N	A	Y
12	State	MFSTAT	Text	20	N	A	Y
13	Zip Code	MFZIPC	Text	9	N	A	Y
14	Ship to Zip Code	MFDSTZ	Text	9	N	A	Y
15	Customer Number (end user)	MPCUST	Text	10	Y	N/A	N
16	Local Tax Flag	MFLTXF	Text	1	N	A	Y
17	Tax amount	MFAMT2	Text	15	N	N 15,5	Y
18	Invoice number	MFREFR	Text	35	Y	A	Y
19	Return Target URL	MPTRGT	Text	500	Y	N/A	N
20	User Defined Field 1	MFUSD1	Text	1	N	A	Y
21	User Defined Field 2	MFUSD2	Text	1	N	A	Y
22	User Defined Field 3	MFUSD3	Text	1	N	A	Y
23	User Defined Field 4	MFUSD4	Text	2	N	A	Y
24	User Defined Field 5	MFUSD5	Text	7	N	A	Y
25	User Defined Field 6	MFUSD6	Text	10	N	A	Y
26	User Defined Field 7	MFUSD7	Text	16	N	A	Y
	~~User Defined Field 8~~	~~MFUSD8~~	~~Text~~	~~16~~	~~N~~	~~A~~	~~Y~~
	~~User Defined Field 9~~	~~MFUSD9~~	~~Text~~	~~32~~	~~N~~	~~A~~	~~Y~~
27	User Defined Field A	MFUSDA	Text	15	N	N 15	Y
28	User Defined Field B	MFUSDB	Text	15	N	N 15	Y
29	User Defined Field C	MFUSDC	Text	15	N	N 15,2	Y
30	User custom fields (name=value string)	MPCUSF	Text	300	N	A	N

***Fields MFUSD8 and MFUSD9 are reserved for Curbstone internal use.**

Note that the "User custom fields" MPCUSF field is designed to contain 300 bytes of any quantity of name=value pairs. These are included, not in the POST payload, but in the URI, strung at the end of the shopping cart URL (* see qualification below). The purpose of these fields, among others, is to allow the shopping cart to include fields that will be logged with the url in the web server log files and elsewhere. Since the payload of the POST is https encrypted, none of those fields are logged anywhere in the many systems that pass the transactions. This allows the programmers of the web shopping cart to have non-sensitive fields exposed for diagnostic or tracking purposes. This is what they will look like

```
https://your-shopping-cart.com/return_url.php?
key1=val1&key2=val2&key3=val3&key4=val4
```

assuming the MPTRGT field is populated like this:

```
'MPTRGT' => 'https://your-shopping-cart.com/return_url.php'
```

and assuming the MPCUSF field is populated like this:

```
'MPCUSF' => 'key1=val1&key2=val2&key3=val3&key4=val4'  // Custom, max 300
chars
```

Note that all URL escaping must be performed prior to populating this field, as well as the inclusion of the field delimiters "&".

(**Note:** MPCUSF fields can be returned in the POST rather than in the URL if the merchant so desires. Simply let your Curbstone Project Manager know, and we will return these key-values pairs via POST.)

11. Response Data Fields Definitions

	Response Fields Names	Description	Stored on Portal
1	MFRTRN	UG=Approved UN=Declined UL=Local Error With Local Error also check MFATAL field for error code	UG - Y UN - Y UL - N
2	MFAPPR	Auth approval code	Y
3	MFSEQN	Network internal tracking	Y
4	MFRREF	More txn result detail	Y
5	MFRTXT	Response reason code text	Y
6	MFRAVS	AVS response code	Y
7	MFRCVV	CVV response code	Y
8	MFUKEY	Unique transaction token	Y
9	MFCARD	Masked card number	Y

10	MFEDAT	Expiry date	Y
11	MFRVNA	Card vendor	Y
12	MFATAL	Local error code (For UL)	N

12. MFATAL Codes for (UL) Local Errors

Code	Description
6	MFCUST - Invalid
7	License Count Overrun
8	Timeout
9	Software not started
?	no response, but possible auth, check
!	Unknown error
A	MFMRCH - Merchant Code
C	MFUSER - Auth user ID
D	MFTYP2 - Secondary txn type
E	MFTYPE - Primary txn type
F	MFADD1 - Address field 1
G	MFAMT1 - Primary amount
H	MFCARD - Card account number mistake
I	MFCVV2 - Card CVV2/CVC2 security code
J	MFEDAT - Expiration date
K	MFMETH - Card entry method
L	MFREFR - Reference/invoice number
P	MFZIPC - Postal or zip code
T	MFLTXF - bad tax value flat (' '|0|1|2)
X	MFCARD - non-numeric

13. CURBSTONE API FIELDS - COMPLETE

See also documentation file **C3_Master_Reference_xx.pdf** on the support site at https://curb911.com.

CURBSTONE API FIELDS - complete

```
* DDS from CCFP9020 and C3FP9020 PHYSICAL FILES
* SUBFIELDS OF C#STRU DATA STRUCTURE
* ============================================================
          K MFUKEY
          R CC9020                    TEXT('TXN STORE')
MF$FCD          1A          TEXT('Force txn denied 0|1')
MF$FCG          1A          TEXT('Force txn +
MF$FCQ          1A          TEXT('Force txn requested +
MF$FTS          1A          TEXT('Flag to settle 0|1')
MF$IED          1A          TEXT('Txn incremented 0|1')
MF$IER          1A          TEXT('Txn increments +
MF$LVD          1A          TEXT('Local void 0|1')
MF$RAD          1A          TEXT('Auth denied 0|1')
MF$RAG          1A          TEXT('Auth granted 0|1')
MF$RAQ          1A          TEXT('Auth requested 0|1')
MF$RED          1A          TEXT('Txn refreshed 0|1')
MF$RER          1A          TEXT('Txn refreshes +
MF$RSD          1A          TEXT('Auth scheduled 0|1')
MF$RVD          1A          TEXT('Reversal denied 0|1')
MF$RVG          1A          TEXT('Reversal granted 0|1')
MF$RVQ          1A          TEXT('Reversal requested 0|1')
MF$SLD          1A          TEXT('Settlement denied 0|1')
MF$SLG          1A          TEXT('Settlement granted 0|1')
MF$SLQ          1A          TEXT('Settlement requested 0|1')
MF$TRP          1A          TEXT('Txn posted 0|1')
MF$TRR          1A          TEXT('Txn reconciled 0|1')
MFACNL          3A          TEXT('Auth channel')
MFACQB          6           TEXT('Acquirer BIN')
MFADD1         35A          TEXT('Address line 1')
MFADD2         35A          TEXT('Address line 2')
MFAMT1         15P 5        TEXT('Txn amount 1')
MFAMT2         15P 5        TEXT('Txn amount 2')
MFAPPR         10A          TEXT('Approval code')
MFATAL          1A          TEXT('Local error')
MFATHC          1           TEXT('Authorization source code')
MFBATC         15P 0        TEXT('Batch number')
MFBDAT         10A          TEXT('Date of birth')
MFCARD         20A          TEXT('Card account number')
MFCHEK         25A          TEXT('Check number')
MFCHKA         25A          TEXT('Check amount')
MFCHKT          1A          TEXT('Check type')
MFCITY         20A          TEXT('City')
MFCURP          1A          TEXT('CurrentPgm')
MFCURR         10A          TEXT('Currency code')
```

```
MFCUST         5A         TEXT('Curbstone customer#')
MFCVV2        10A         TEXT('Security code')
MFDATA        60          TEXT('Internal data')
MFDATE         8A         TEXT('Transaction Date')
MFDSTZ         9A         TEXT('Destination zip code')
MFEDAT         4A         TEXT('Expiry date')
MFGRP3         3          TEXT('Group III ver#')
MFHMID         8          TEXT('Host message ID')
MFIDEN        35A         TEXT('Identification')
MFIERR         7A         TEXT('Internal error')
MFKEYN        15A         TEXT('Next key')
MFKEYP        15A         TEXT('Previous key')
MFLTXF         1A         TEXT('Local tax flag')
MFMETH         2A         TEXT('Entry method')
MFMICR        80A         TEXT('Check MICR data')
MFMKDI         4          TEXT('Market data ID')
MFMRCH         5A         TEXT('Merchant code')
MFNAME        40A         TEXT('Cardholder name')
MFNUMB         8          TEXT('Network batch +
MFNWID         1          TEXT('Network ID code')
MFORDR        16A         TEXT('Order#')
MFPINC        40A         TEXT('PIN Code')
MFPRCD         8A         TEXT('Processed date')
MFPRCT         6A         TEXT('Processed time')
MFPRIO         1A         TEXT('Priority')
MFPROG         1A         TEXT('Pgm')
MFQACI         1A         TEXT('Requested ACI')
MFRACI         1A         TEXT('Returned ACI')
MFRAVS         2A         TEXT('AVS response')
MFRCVV         2A         TEXT('CVV2 response')
MFRDAT         8A         TEXT('Query return date')
MFREFR        35A         TEXT('Txn reference')
MFREQN         7A         TEXT('Settlement request +
MFRETR        35A         TEXT('Retrieval reference')
MFRLIB        10A         TEXT('Return Lib')
MFRLOC        10A         TEXT('Sec location')
MFRQUE        10A         TEXT('Return queue')
MFRREF        35A         TEXT('Response reference')
MFRTIM         6A         TEXT('Query return time')
MFRTRN         2A         TEXT('Return code')
MFRTVR        12          TEXT('Retrieval ref#')
MFRTXT        35A         TEXT('Response text')
MFRVNA         4A         TEXT('Vendor abbr')
MFRVND        16A         TEXT('Returned vendor')
MFSCNL         3A         TEXT('Settlement channel')
MFSEQN        15P 0       TEXT('Txn Sequence')
MFSETA        15P 5       TEXT('Settled amount')
MFSETD         8A         TEXT('Settlement date')
MFSETR        15P 5       TEXT('Settle amt requested')
MFSETT         6A         TEXT('Settlement time')
```

```
MFSHPD          8A          TEXT('Shipment date')
MFSTAN          6           TEXT('System trace audit#')
MFSTAT         20A          TEXT('State')
MFSTID          8A          TEXT('Station ID')
MFSTLD          4           TEXT('Settlement date')
MFSTRN          4           TEXT('Store#')
MFSTUS          7A          TEXT('User app status')
MFSTXT         35A          TEXT('Settle text')
MFSUSR         10A          TEXT('Settle User ID')
MFTERM          8A          TEXT('Terminal number')
MFTIME          6A          TEXT('Transaction time')
MFTRAN         10A          TEXT('Transit routing#')
MFTRID         15           TEXT('Transaction ID')
MFTRK1        256A          TEXT('Track data')
MFTRNS          4           TEXT('Txn seq#')
MFTYP2          2A          TEXT('Secondary type')
MFTYPE          2A          TEXT('Transaction Type')
MFUKEY         15A          TEXT('Unique Key')
MFUSD1          1A          TEXT('MFUSD1 User-defined')
MFUSD2          1A          TEXT('MFUSD2 User-defined')
MFUSD3          1A          TEXT('MFUSD3 User-defined')
MFUSD4          2A          TEXT('MFUSD4 User-defined')
MFUSD5          7A          TEXT('MFUSD5 User-defined')
MFUSD6         10A          TEXT('MFUSD6 User-defined')
MFUSD7         16A          TEXT('MFUSD7 User-defined')
MFUSD8         16A          TEXT('MFUSD8 User-defined') - Reserved for
Curbstone internal use.
MFUSD9         32A          TEXT('MFUSD9 User-defined') - Reserved for
Curbstone internal use.
MFUSDA         15P 0        TEXT('MFUSDA User-defined')
MFUSDB         15P 0        TEXT('MFUSDB User-defined')
MFUSDC         15P 2        TEXT('MFUSDC User-defined')
MFUSER         10A          TEXT('Auth user ID')
MFVALC          4           TEXT('Validation code')
MFVALU         15P 5        TEXT('Returned count')
MFXTRA         10A          TEXT('Internal xtra')
MFZIPC         13A          TEXT('Postal code')
```

14. Customizable Cosmetic Elements

14.1. Client.ini File

This file is used for both embedded (i.e. iFrame) and redirect (i.e. non-iFrame) modes.

Customer-specific versions of this file are only necessary if the customer wishes to deviate from the default configuration values.

Field	Description	Possible	Default Value

		Values	
mask_expiry	Whether or not the card expiration date should be masked prior to being returned by PLP.	true, false	false
custom_field_return_mode	HTTP request mode that should be used to return custom fields. (Custom fields are specified using MPCUSF.)	GET, POST	GET
pay_button_text	The text that displays on the form submit button.	Any string	PAY
pay_button_message	The text that display underneath the form submit button.	Any string	Click 'PAY' to authorize payment

File CLIENT.INI

```
;;;;;  DEFAULT CONFIGURATION
 ************Beginning of data**************
; Client-specific configuration settings
;
mask_expiry = false
custom_field_return_mode = GET
pay_button_text = PAY
pay_button_message = Click 'PAY' to authorize payment
 ************End of Data********************

;;;;;  EXAMPLE CUSTOM CONFIGURATION
 ************Beginning of data**************
; Client-specific configuration settings
;
mask_expiry = false
custom_field_return_mode = GET
pay_button_text = Submit Order
pay_button_message = Click Submit to authorize
 ************End of Data********************
```

14.2. Header.html File

This file is used for only full redirect (i.e. non-iFrame) modes. Customer-specific versions of this file SHOULD ONLY be included if the customer does not plan to embed PLP in an iFrame.

File HEADER.HTML

```
DEFAULT CONFIGURATION
 ************Beginning of data**************
<!DOCTYPE html>
<html>
    <head>
        <title>Curbstone C3 PLP</title>
        <meta charset="UTF-8" />
    </head>
    <body>

        <div id="plp-wrapper">
 ************End of Data********************
```

14.3. Footer.html File

This file is used for both embedded (i.e. iFrame) and redirect (i.e. non-iFrame) modes.

Customer-specific versions of this file SHOULD ONLY be included if the customer does not plan to embed PLP in an iFrame.

File FOOTER.HTML

```
DEFAULT CONFIGURATION
 ************Beginning of data**************
        </div><!-- #plp-wrapper -->
    </body>
</html>
 ************End of Data********************
```

14.4. Form_header.html File

This file is used for both embedded (i.e. iFrame) and redirect (i.e. non-iFrame) modes.

File FORM_HEADER.HTML

```
DEFAULT CONFIGURATION
 ************Beginning of data**************
<div class="plp-form-header">
    <h2>Enter Credit Card Info</h2>
</div>
 ************End of Data********************
```

14.5. Form_footer.html File

This file is used for both embedded (i.e. iFrame) and redirect (i.e. non-iFrame) modes.

Customer-specific versions of this file should only be necessary if extensive customizations are needed. In most cases, it should be enough to just specify a custom message to display under the submit button. To do this, simply set the `pay_button_message` field in client.ini (discussed above).

File FORM_FOOTER.HTML

```
DEFAULT CONFIGURATION
 ************Beginning of data**************
<div class="plp-form-footer">
    {PAY_BUTTON_MESSAGE}
</div>
 ************End of Data********************
```

14.6. Stylesheet.css File

This file is used for both embedded (i.e. iFrame) and redirect (i.e. non-iFrame) modes.

See the CSS stylesheet source code in the next section to see how PLP elements are currently being styled. Any of the styles can be overridden via a customer-specific stylesheet.

14.6.1. PLP Default Styles

PLP Default Styles

```
body {
    background-color: #fff;
    margin: 0;
    padding: 0;
    font-family:sans-serif;
}

h2 {
    margin: .5rem 0;
}

.plp-error-header {
    display: block;
    width: 100%;
    text-align: center;
    margin-top: 3rem;
    font-weight: normal;
    font-size: 1.2rem;
}
```

```
.plp-clear {
    float: none;
    clear: both;
    height: 0;
}

.plp-success,
.plp-error {
    display: block;
    float: left;
    width: 91%;
    border-radius: 5px;
    padding: .4rem 4% .4rem 4%;
    margin: 0 0 .5rem 0;
}

.plp-success {
    border: 1px solid green;
    background-color: #CCE6CC;
}

.plp-error {
    border: 1px solid red;
    background-color: pink;
}

.plp-payment-form td {
    padding: .3rem .5rem;
}

.plp-payment-form td:first-child {
    text-align: right;
    font-weight: bold;
}

.plp-payment-form input,
.plp-payment-form select {
    padding: .3rem .5rem;
    font-size: 1rem;
}

.plp-payment-form select {
    font-size: 1rem;
    padding: 6px 12px !important;
}

.plp-payment-form button {
    padding: .5rem 1.5rem;
    font-size: 1rem;
```

```
    font-weight: bold;
    cursor: pointer;
    border: 1px solid red;
    border-radius: 5px;
    background-color: #F7701D;
    color: #FFF;
}

.plp-payment-form button:hover {
    background-color: #F5D625;
    color: red;
}

#plp-spinner-wrapper,
#plp-form-wrapper {
    width: 290px;
}

#plp-spinner-wrapper {
    height: 290px;
    margin: 0 auto;
    position: relative;
}

#plp-spinner {
    width: 100px;
    height: 100px;
    position: absolute;
    top: 50%;
    left: 50%;
    margin: -60px 0 0 -50px;
}

#plp-wrapper {
    padding-top: 1rem;
}

#plp-form-wrapper {
    margin: 0 auto;
    padding-top: 10px;
}

#plp-form-wrapper .plp-form-header,
#plp-form-wrapper .plp-form-footer,
#plp-form-wrapper .plp-submit-button {
    text-align: center;
}

#plp-form-wrapper .plp-submit-button {
    padding-top: .5rem;
```

```
}

#plp-form-wrapper .plp-form-footer {
    padding: .8rem;
}
```

14.7. CSS/HTML Modification Stipulations

CSS and HTML MODIFICATIONS
ALL changes must be approved by Curbstone prior to being placed in service.

ALL files must reside on the Curbstone portal server. No reference to any external file is allowed in customized CSS code.

15. Demo Source Code for Reference

This is the actual source code from the "live" demo documented above. We suggest it only as a reference that is know to work. The above source in Step 1 and Step 2 is optimized to focus on the specifics of only what is absolutely needed.

The following code is from the online demo, and has extraneous material that is not critical to integration. We provide it because more examples are always better than fewer. This is available in a ZIP file by download from our support site.

15.1. Demo: CSS Style File

Reference Style Sheet

```
html {
    font-size: 90.%;
}

body {
    float: left;
    width: 100%;
    margin: 0;
    padding: 0;
    font-family:sans-serif;
}


header {
    float: left;
    width: 100%;
    height: 6rem;
    background-color: #e7e7e7;
```

```
    border-bottom: 1px solid #777;
}

header > a {
    float: left;
    width: 50%;
    display: block;
    font-size: 2.5rem;
    padding: 2rem .5rem 0 1rem;
}


header > ul {
    display: block;
    float: right;
    width: 45%;
    list-style-type: none;
    margin: 3rem 1rem 1rem 0;
    padding: 0;
    font-size: 1.2rem;
}

header > ul li {
    float: right;
    width: 27%;
    text-align: right;
    padding: .5rem 0 0 0;
}

nav {
    float: left;
    width: 100%;
    height: 3rem;
    background-color: #cacaca;
    border-bottom: 1px solid #777;
}

nav ul {
    list-style-type: none;
    margin: .2rem 0 0 2rem;
    padding: 0;
    font-size: 1.5rem;
}

nav ul li {
    float: left;
    width: 15%;
    padding: .5rem 1rem 0 1rem;
}
```

```
footer {
    float: left;
    text-align: center;
    width: 100%;
    height: 1.7rem;
    background-color: #cacaca;
    border-top: 1px solid #777;
    border-bottom: 1px solid #777;
    padding: .5rem 0 0 0;
    margin-top: 30px;
}

footer ul {
    list-style-type: none;
    margin: .5rem 0 0 2rem;
    padding: 0;
}

footer ul li {
    float: left;
    width: 10%;
    padding: .5rem 0 0 .5rem;
}

.clear {
    float: none;
    clear: both;
    height: 0;
}

.center {
    margin: 0 auto;
}


#main {
    float: left;
    width: 100%;
    padding: 20px 0;
}

#index-container,
#payment-container {
    width: 500px;
    padding: 0;
    font-family:sans-serif;
}

#payment-container {
    height: 500px;
```

```
}

#payment-container iframe {
    border-radius: 10px;
    border: 1px solid #777;
    width: 500px;
    height: 340px;
    padding: 0;
    font-family:sans-serif;
}

#index-container,
#payment-container,
#results-container {
    margin: 0 auto;
}

#results-container {
    width: 800px;
}

a.submit-button,
.submit-button button {
    display: block;
    width: 200px;
    text-align: center;
    padding-top: .5rem;
    padding: .5rem 1.5rem;
    font-size: 1rem;
    font-weight: bold;
    cursor: pointer;
    border: 1px solid red;
    border-radius: 5px;
    background-color: #F7701D;
    color: #FFF;
    text-decoration: none;
}

a.submit-button:hover,
.submit-button button:hover {
    background-color: #F5D625;
    color: red;
    text-decoration: none;
}

.success { color: green; }
.error { color: red; }
```

15.2. Demo: config.ini Configuration File

Reference Demo Configuration File

```
; Curbstone CorrectConnect
; Payment Landing Pages "PLP" Client Configuration file
; This is part of the demo code for the reference PLP client side
; Comment lines in this file are preceded by the semicolon ";"
; https://curb911.com   Support Site
; 888-844-8533

; Client base url
; This must be the working folder for the shopping cart
; from which the included "index.php" is executed.
; Trailing/ slash/ is/ required/
;
base_url = https://curbstone.com/plp/

; URL pointing to PLP installation
; This will be provided by Curbstone for your implementation
; In the demo, the URL
;     https://c3plp.net/curbstone/plp/
; must be used - do not change this value unless instructed
;
plp_api_url = https://c3plp.net/curbstone/plp/

; URL to which control is returned - this must be a program
; that can accept the POSTed data from the C3 portal and
; parse it.  Change this to match the program in YOUR Shopping
; Cart that will receive the response from the C3 server as to
; the success or failure of the authorization/storage request.
; This must be a fully-qualified URL that includes the filename
; of the receiving program, unless it is "index.php", in which
; case it must end/ with/ a/ trailing/ slash/!
;
target_url = https://curbstone.com/plp/return_url.php
```

15.3. Demo index.php: Example Checkout Page Source Code

Reference Demo Cart Completion Page

```
<?php
/**
 * Curbstone CorrectConnet PLP DEMO site example source code - index.php
 */
```

```
// set the displayed version number for this file
$pagevers = '1.8';

// Base Path
if (!defined('CLIENT_BASE_PATH')) { define('CLIENT_BASE_PATH', dirname
(__FILE__) . DIRECTORY_SEPARATOR); }

// Determine the include path based on the master config file
$CONFIG = parse_ini_file(CLIENT_BASE_PATH . 'config.ini');

// Specify the url of the credit card payment page
$PAYMENT_URL = $CONFIG['base_url'] . 'payment.php';

// Render the page
echo <<<OUT
<!DOCTYPE html>
<html>
    <head>
        <title>Curbstone PLP DEMO</title>
        <meta charset="UTF-8" />
        <base href="{$CONFIG['base_url']}">
        <link rel="stylesheet" href="styles/qlient.css" />
    </head>
    <body>

        <a href="{$CONFIG['base_url']}"><img src="Curbstone-plp-demo_01.
png"></a>
        <hr>
        <div id="main">
            <div id="index-container">

<strong>PLP: PROGRAMMER DEMO - not what customers see!<br /><br />
YOUR PROGRAM SUBMITS THESE FIELDS IN THE BACKGROUND</strong>
to the Curbstone C3 Portal as a header, and stored. Subset of fields
supported.
<br /><br />

<form method="post" action="$PAYMENT_URL">
    <table>
        <tr><td>SERVER</td>
            <td><select name="SVRTGT">
                    <option>SANDBOX</option>
                    <option>LIVE</option>
                </select></td></tr>
        <tr><td>MFCUST</td>
            <td><input type="text" name="MFCUST" size="5" maxlength="5"
value="00001" /></td></tr>
        <tr><td>MFMRCH</td>
            <td><select name="MFMRCH">
                    <option>99998</option>
```

```
                    <option>97777</option>
                    <option>98772</option>
                    <option>97666</option>
                </select></td></tr>
        <tr><td>MFTYPE</td>
            <td><select name="MFTYPE">
                    <option>RA</option>
                    <option>RY</option>
                </select></td></tr>
        <tr><td>MFTYP2</td>
            <td><select name="MFTYP2">
                    <option>PA</option>
                    <option>SA</option>
                    <option>  </option>
                </select></td></tr>
        <tr><td>MFORDR</td>
            <td><input type="text" name="MFORDR" size="25" maxlength="
20" value="0987654321" /></td></tr>
        <tr><td>MFAMT1</td>
            <td><input type="text" name="MFAMT1" size="10" maxlength="
8" value="1.01" /></td></tr>
        <tr><td>MFAMT2</td>
            <td><input type="text" name="MFAMT2" size="10" maxlength="
8" value="0.02" /></td></tr>
        <tr><td>MFQACI</td>
            <td><input type="text" name="MFQACI" size="1" maxlength="1"
value=" " /></td></tr>
        <tr><td>MFNAME</td>
            <td><input type="text" name="MFNAME" size="25" maxlength="
20" value="Web Customer" /></td></tr>
        <tr><td>MFADD1</td>
            <td><input type="text" name="MFADD1" size="25" maxlength="
20" value="201 Enterprise Court" /></td></tr>
        <tr><td>MFADD2</td>
            <td><input type="text" name="MFADD2" size="25" maxlength="
20" value="" /></td></tr>
        <tr><td>MFCITY</td>
            <td><input type="text" name="MFCITY" size="20" maxlength="
15" value="Ball Ground" /></td></tr>
        <tr><td>MFSTAT</td>
            <td><input type="text" name="MFSTAT" size="4" maxlength="2"
value="GA" /></td></tr>
        <tr><td>MFZIPC</td>
            <td><input type="text" name="MFZIPC" size="7" maxlength="5"
value="30107" /></td></tr>
        <tr><td>MFDSTZ</td>
            <td><input type="text" name="MFDSTZ" size="7" maxlength="5"
value="30107" /></td></tr>
        <tr><td>MFREFR</td>
            <td><input type="text" name="MFREFR" size="25" maxlength="
```

```
20" value="6789012345" /></td></tr>
        <tr><td>MPCUST</td>
            <td><input type="text" name="MPCUST" size="25" maxlength="
20" value="4444888822220000" /></td></tr>
        <tr><td>MPCUSF</td>
            <td><input type="text" name="MPCUSF" size="50" maxlength="
500" value="" /></td></tr>
                <tr><td>MFUSER</td>
            <td><input type="text" name="MFUSER" size="5" maxlength="
10" value="" /></td></tr>
        <tr><td colspan="2">
            <p><input id="mode-embed" type="radio" name="mode" value="
embed" checked="checked" />
               <label for="mode-embed">Embed in iFrame</label></p>
            <p><input id="mode-redirect" type="radio" name="mode"
value="redirect" />
               <label for="mode-redirect">Use full redirect</label></p><
/td></tr>
    </table>
    <br />
    <div class="submit-button">
        <button type="submit">Submit</button>
    </div>

</form>
</div>
</div><!-- #main -->
<br class="clear" />

<footer>
   <strong>Curbstone Corporation</strong>   Payment Landing Pages (PLP)
Demo -- Version $pagevers
</footer>

</body>
</html>
OUT;
```

15.4. Demo payment.php: Example Payment Page Source Code

Reference Demo iFrame Payment Page

```
<?php
/**
 * Curbstone CorrectConnet PLP DEMO site example source code - payment.
php
 **/
```

```
// set the displayed version number for this file
$pagevers = '1.8';

// Base Path
if (!defined('CLIENT_BASE_PATH')) { define('CLIENT_BASE_PATH', dirname
(__FILE__) . DIRECTORY_SEPARATOR); }

// Determine the include path based on the master config file
$CONFIG = parse_ini_file(CLIENT_BASE_PATH . 'config.ini');

// Specify the url of the ***LIVE*** PLP initialization endpoint
$PLP_API_URL = $CONFIG['plp_api_url'];
// Specify the url of the ***SANDBOX*** PLP initialization endpoint
$PLP_SBX_URL = $CONFIG['plp_sbx_url'];

// Grab the choice of target server from the entry screen
$SVRTGT = $_POST['SVRTGT'];  // LIVE or SANDBOX, your choice!

if ($SVRTGT == 'SANDBOX') {
$PLP_SVR_URL = $PLP_SBX_URL;
} else {
$PLP_SVR_URL = $PLP_API_URL;
}

// Setup the variables for all of the fields in the Curbstone Data
Structure
// This is typically how the data is constructed for the PLP API
// The example program index.php is just a front end to demo the API
    $MFADD1 = $_POST['MFADD1'];  // AVS
    $MFADD2 = addslashes($_POST['MFADD2']);
    $MFAMT1 = $_POST['MFAMT1'];  // purchase amount
    $MFCITY = $_POST['MFCITY'];
    $MFCUST = $_POST['MFCUST'];
    $MFMETH = '02';  // ecommerce must be 02
    $MFMRCH = $_POST['MFMRCH'];
    $MFNAME = $_POST['MFNAME'];
    $MFREFR = $_POST['MFREFR'];  // invoice number
    $MFSTAT = $_POST['MFSTAT'];
    $MFTYPE = $_POST['MFTYPE'];
    $MFTYP2 = $_POST['MFTYP2'];
    // Level II fields start
    $MFAMT2 = $_POST['MFAMT2'];  // tax amount
    $MFORDR = $_POST['MFORDR'];  // customer PO number
    $MFDSTZ = $_POST['MFDSTZ'];  // destination zip code
    $MFLTXF = '1';               // tax flag
    // Level II fields end

// this is our chance to add some background data programmatically
///////
```

```
    // start of user defined fields
    $MFUSD1 = '1';
    $MFUSD2 = '2';
    $MFUSD3 = '3';
    $MFUSD4 = '44';
    $MFUSD5 = '5555555';
    $MFUSD6 = '6666666666';
    $MFUSD7 = '7777777777777777';
    $MFUSD8 = '8888888888888888';
    $MFUSDA = 123456789012345;
    $MFUSDB = 123456789012345;
    $MFUSDC = 123456789012.34;
    // end of user defined fields
    $MFUSER = $_POST['MFUSER'];
    $MFZIPC = $_POST['MFZIPC'];  // AVS
    // this puts the invoice number in the URL to be visible for diags
    $MPCUSF = 'mfrefr=' . $MFREFR . '&mftype=' . $MFTYPE . $MFTYP2 .
'&key3=val3&key4=val4';
    $MPCUSF = $_POST['MPCUSF'];  // Custom query string. Max 300 chars
    $MPCUST = $_POST['MPCUST'];
    $MPTRGT = $CONFIG['target_url'];

// Specify all non-card data that will be transmitted from the client's
server.
// Also, specify a target URL for PLP to return the results of the
transaction to the client?s server.
$payload = array(
    'MFADD1' => $MFADD1,
    'MFADD2' => $MFADD2,
    'MFAMT1' => $MFAMT1,
    'MFAMT2' => $MFAMT2,
    'MFCITY' => $MFCITY,
    'MFDSTZ' => $MFDSTZ,
    'MFLTXF' => $MFLTXF,
    'MFMETH' => $MFMETH,
    'MFMRCH' => $MFMRCH,
    'MFNAME' => $MFNAME,
    'MFORDR' => $MFORDR,
    'MFQACI' => $MFQACI,
    'MFREFR' => $MFREFR,
    'MFSTAT' => $MFSTAT,
    'MFTYP2' => $MFTYP2,
    'MFTYPE' => $MFTYPE,
    'MFUSD1' => $MFUSD1,
    'MFUSD2' => $MFUSD2,
    'MFUSD3' => $MFUSD3,
    'MFUSD4' => $MFUSD4,
    'MFUSD5' => $MFUSD5,
    'MFUSD6' => $MFUSD6,
    'MFUSD7' => $MFUSD7,
```

```
    'MFUSD8' => $MFUSD8,
    'MFUSDA' => $MFUSDA,
    'MFUSDB' => $MFUSDB,
    'MFUSDC' => $MFUSDC,
    'MFUSER' => $MFUSER,
    'MFZIPC' => $MFZIPC,
    'MPCUSF' => $MPCUSF,
    'MPCUST' => $MPCUST,
    'MPTRGT' => $MPTRGT,
        'MFCUST' => $MFCUST
);

// Format the payload for transmission
$payload_string = '';
foreach ($payload as $key => $value) {
    $payload_string .= $key . '=' . urlencode($value) . '&';
}
rtrim($payload_string, '&');

// Open a connection with the PLP endpoint
$ch = curl_init();

// Configure the POST request
curl_setopt($ch, CURLOPT_URL, $PLP_SVR_URL . '?action=init');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $payload_string);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false); // Remove line or
change value to 'true' when running in production

// Execute the POST and capture the response
$result = curl_exec($ch);

if ($result == false) {
    echo '<font face=courier>PLP: Curl error#..: ' . curl_errno($ch) .
"<br>";
    echo 'PLP: Curl error...: ' . curl_error($ch) . "<br>";
    echo 'PLP: Curl target..: ' . $PLP_SVR_URL . "<br></font>";
}

// Close the connection
curl_close($ch);

$result = json_decode($result, true);

// Demonstrates basic success/error checking
switch ($result['MFRTRN']) {
    case 'UG':
        if (!array_key_exists('MFSESS', $result)) {
            die('Transaction session ID was not returned');
```

```
        }
        break;

    case 'UL':
    default:
        if (is_array($result)) {
            // Single-character error code
            if (array_key_exists('MFATAL', $result)) {
                echo 'MFATAL ERROR CODE: ' . $result['MFATAL'] . '<br>';
            }
            // Human-readable error message
            if (array_key_exists('MFRTXT', $result)) {
                echo 'MFATAL ERROR MESSAGE: ' . $result['MFRTXT'] .
'<br>';
            }
        }

        die('Transaction initialization failed');
        exit; }

// Extract the transaction session ID from the JSON payload
$SESSION_ID = $result['MFSESS'];

// Build the URL to point to the PLP user interface.
// This will be embedded in the HTML of the web page.
$PLP_TXN_URL = $PLP_SVR_URL . '?MFSESS=' . $SESSION_ID;

// Redirect instead of rendering the page if operating in "non-iframe"
mode
if (array_key_exists('mode', $_POST) && $_POST['mode'] == 'redirect') {
    header('Location: ' . $PLP_TXN_URL . '&mode=redirect');
    exit; }

// Render the payment page.
// Most of the following HTML is completely arbitrary and can be
customized as needed.
// The only required elements are the iFrame and PLP JavaScript library.
echo <<<OUT
<!DOCTYPE html>
<html>
    <head>
        <title>Payment Entry - Curbstone PLP </title>
        <meta charset="UTF-8" />
        <base href="{$CONFIG['base_url']}">
        <link rel="stylesheet" href="styles/qlient.css" />
    </head>
    <body>

        <a href="{$CONFIG['base_url']}"><img src="Curbstone-plp-demo_01.
png"></a>
```

```
        <hr>
        <div id="main">

            <div id="payment-container">

<p>Your iFrame below is generated from the Curbstone Portal at
$PLP_SVR_URL.  The browser address bar URL is still the original.</p>

<p>Use card <strong>5454545454545454</strong>, any future expiry,
and security code is not checked.
"Session ID" is for iFrame, not transaction.</p>

                <!-- Only displayed to easily reference the transaction
session ID in use -->
                <!-- Typically would not be displayed to the end-user
cardholder             -->
                <p><strong><font face="courier, monospace">iFrame
Session ID: $SESSION_ID</font></strong></p>

                <!-- REQUIRED -->
                <!-- iFrame MUST include an id of "plp-iframe" -->
                <iframe id="plp-iframe" style="display:none;" src="
{$PLP_TXN_URL}&mode=embedded"></iframe>


                <!-- OPTIONAL -->
                <!-- The noscript tag allows non-JavaScript users to
proceed with the payment -->
                <!-- This will display in place of the iFrame if the
user has JavaScript disabled -->
                <noscript>
                    <p style="display:block;border:1px solid red;
background-color:pink;font-weight:bold;padding:20px;">
                        JavaScript not enabled. Go to our <a href="
{$PLP_TXN_URL}&mode=redirect">payment portal</a> to complete your order.
                    </p>
                </noscript>

            </div>

            <!-- REQUIRED -->
            <!-- This JavaScript library is responsible for: -->
            <!--    1) Making the iFrame visible -->
            <!--    2) Breaking out of iFrame, redirect to the target
page (MPTRGT) on completion -->
            <script type="text/javascript" src="{$PLP_SVR_URL}scripts
/plp.js"></script>

        </div>
```

```
        <br class="clear" />

        <footer>
            <strong>Curbstone Corporation</strong>   (PLP) Demo --
Version $pagevers
        </footer>

    </body>
</html>
OUT;

?>
```

15.5. Demo return_url.php: Example Demo Target Page Source Code

Reference Demo Recipient Page

```
<?php

// set the displayed version number for this file
$pagevers = '1.8';

// Base Path
if (!defined('CLIENT_BASE_PATH')) { define('CLIENT_BASE_PATH', dirname
(__FILE__) . DIRECTORY_SEPARATOR); }

// Determine the include path based on the master config file
$CONFIG = parse_ini_file(CLIENT_BASE_PATH . 'config.ini');

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $response = $_POST;
} else {
    $response = $_GET;
}
ksort($response);

if (array_key_exists('MFRTRN', $response)) {
    // Determine if the transaction was successful
    switch ($response['MFRTRN']) {
        case 'UG':
            $confirmation_msg = 'Authorization successful.';
            $txn_status = 'success';
            break;
        case 'UN':
            $confirmation_msg = 'Authorization declined: MFRTXT: ' .
$response['MFRTXT'];
            $txn_status = 'error';
```

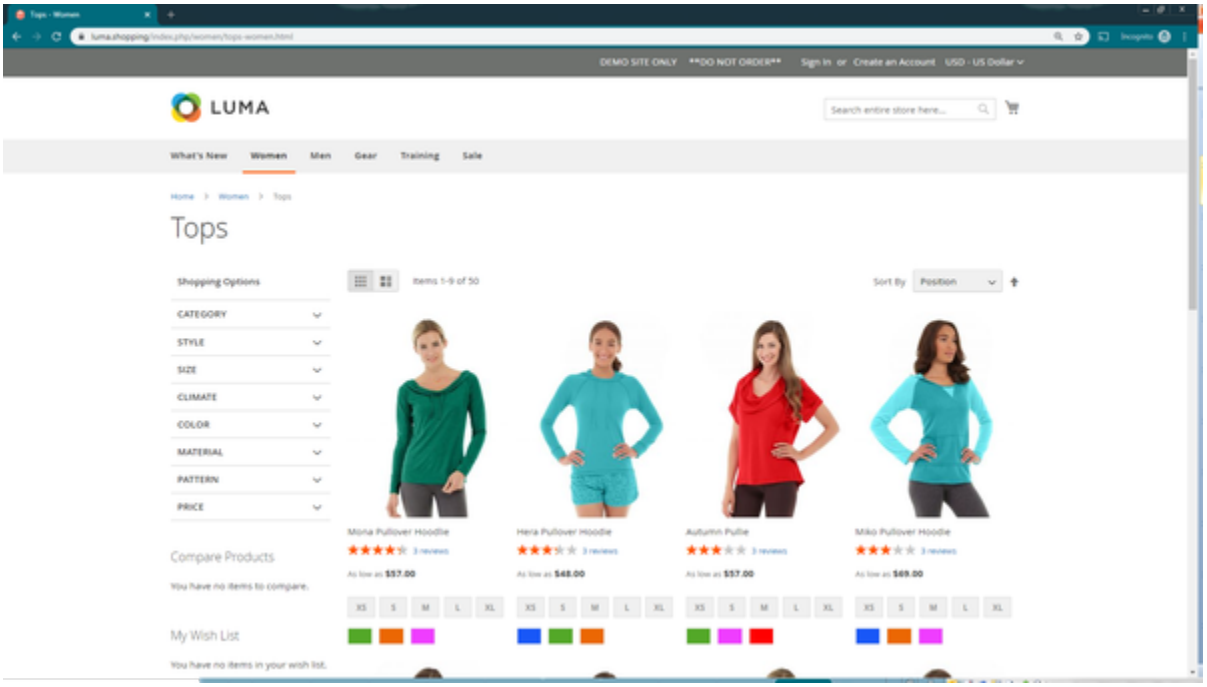
```
            break;
        case 'UL':
        default:
            $confirmation_msg = 'Field Error - code: MFATAL: ' .
$response['MFATAL'];
            $txn_status = 'error';
            break;
    }
} else {
    $confirmation_msg = '';
    $txn_status = '';
}

$response_fields = '<font face="monospace">';
foreach ($response as $key => $value) {
    $response_fields .= htmlentities($key, ENT_QUOTES) . ' => ' .
htmlentities($value, ENT_QUOTES) . '<br> ';
}
$response_fields .= '</font>';

echo <<<OUT
<!DOCTYPE html>
<html>
<head>
    <title>Payment Complete - Curbstone PLP </title>
    <meta charset="UTF-8" />
    <base href="{$CONFIG['base_url']}">
    <link rel="stylesheet" href="styles/qlient.css" />
</head>
<body>
<a href="{$CONFIG['base_url']}"><img src="Curbstone-plp-demo_01.png"><
/a>
<hr>
<div id="main">
    <div id="results-container">
        <a class="submit-button" href="{$CONFIG['base_url']}">Submit
Another</a>
        <h3 class="$txn_status">$confirmation_msg</h3>

        $response_fields

<p><b>MFUKEY</b> is the transaction <b>TOKEN</b> to store
with order for later settlement.</p>

        <a class="submit-button" href="{$CONFIG['base_url']}">Submit
Another</a>

    </div>
</div>
<footer>
```

```
    <strong>Curbstone Corporation</strong>   Payment Landing Pages
(PLP) Demo -- Version $pagevers
</footer>
</body>
</html>
OUT;
```

16. MAGENTO 2 Extension

Curbstone has an extension for the popular Magento shopping cart system. Inquire about our MLP if you run Magento 2.x. We built a fully-functional Curbstone Magento DEMO site at

> **http://luma.shopping**



17. C3 Isolated Card Entry - ICE

To see what ICE does, visit the demo:

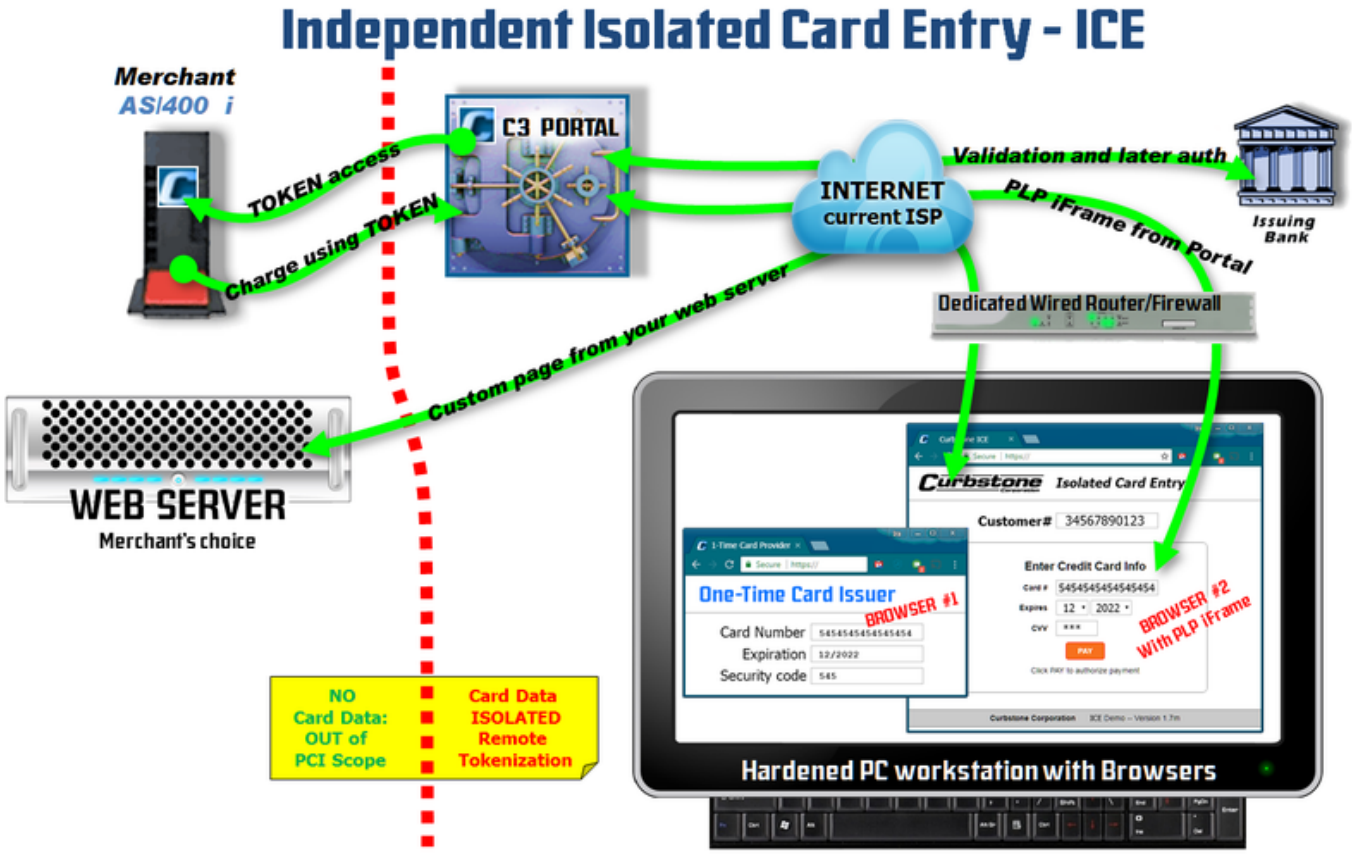
the ICE demo at

> **https://curbstone.com/ice**
>
> user = `ice`
> password = `c3ice`

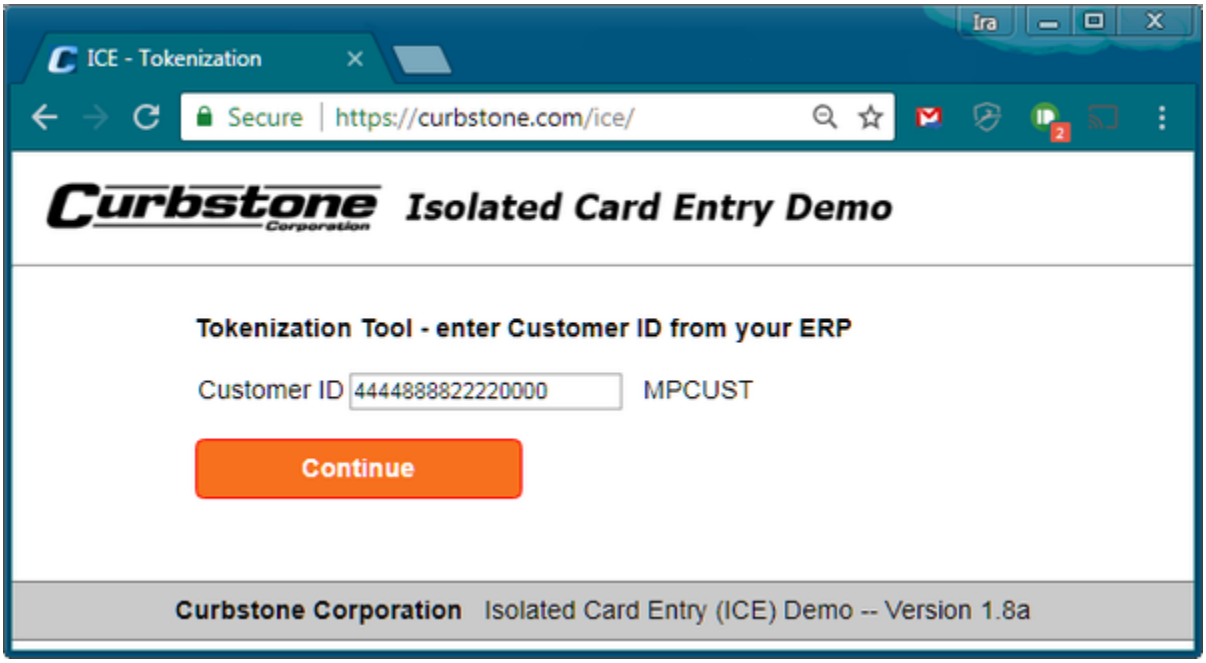
One use of PLP can be to enter card data to associate it with a customer ID in your existing ERP software. An example of this use would be to store one-time-use credit cards that your A/R people have to copy and paste from a third-party web site. If they did this on their everyday workstations, your entire

infrastructure would be INSTANTLY back in scope of PCI. Assuming you engage the Curbstone technology (IPT, PLP, EMV, DSI) when you are handling credit card data, you wanted your systems to be OUT of PCI scope.

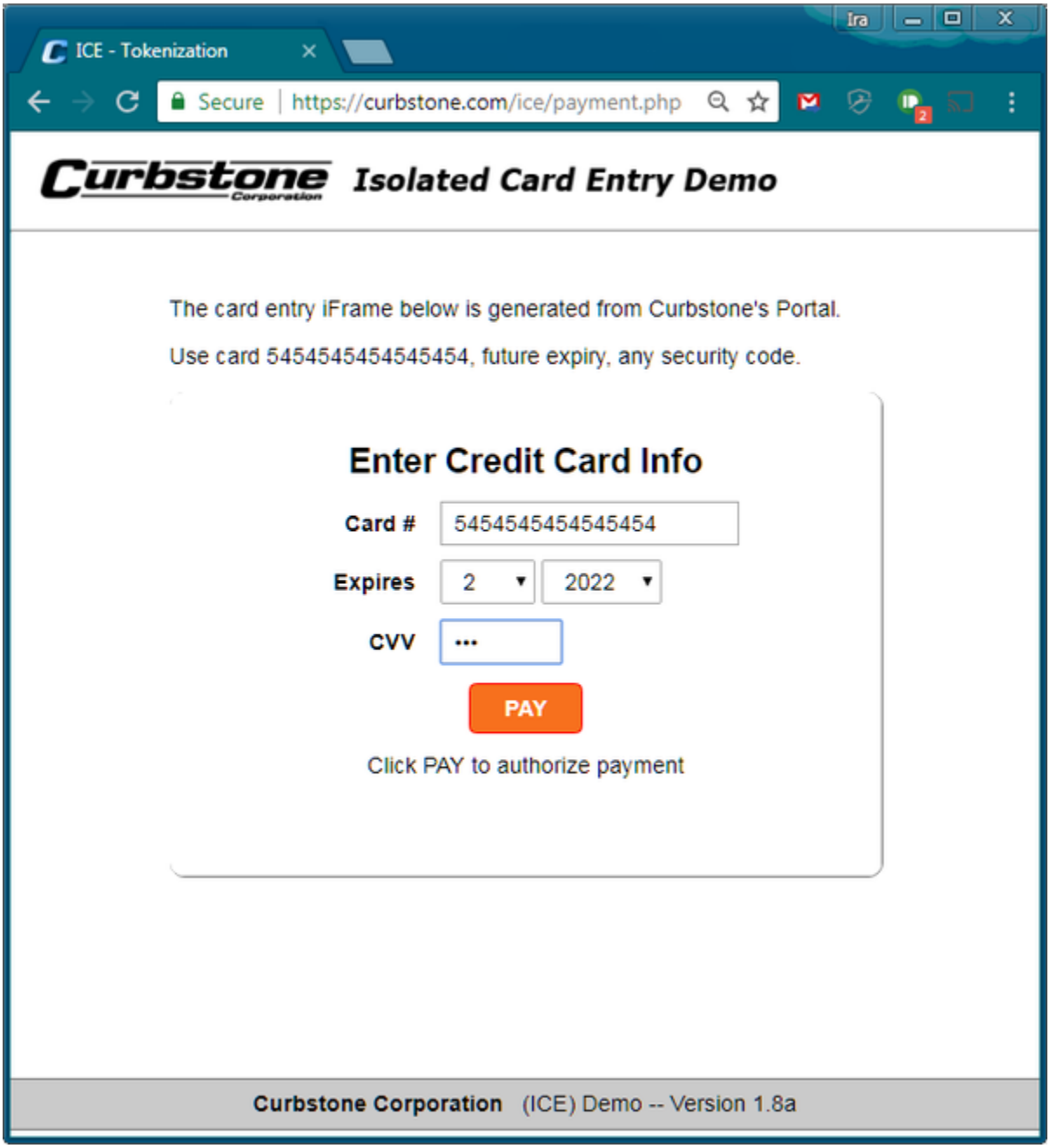
This utility allows you to designate ONE isolated workstation to handle this operation.



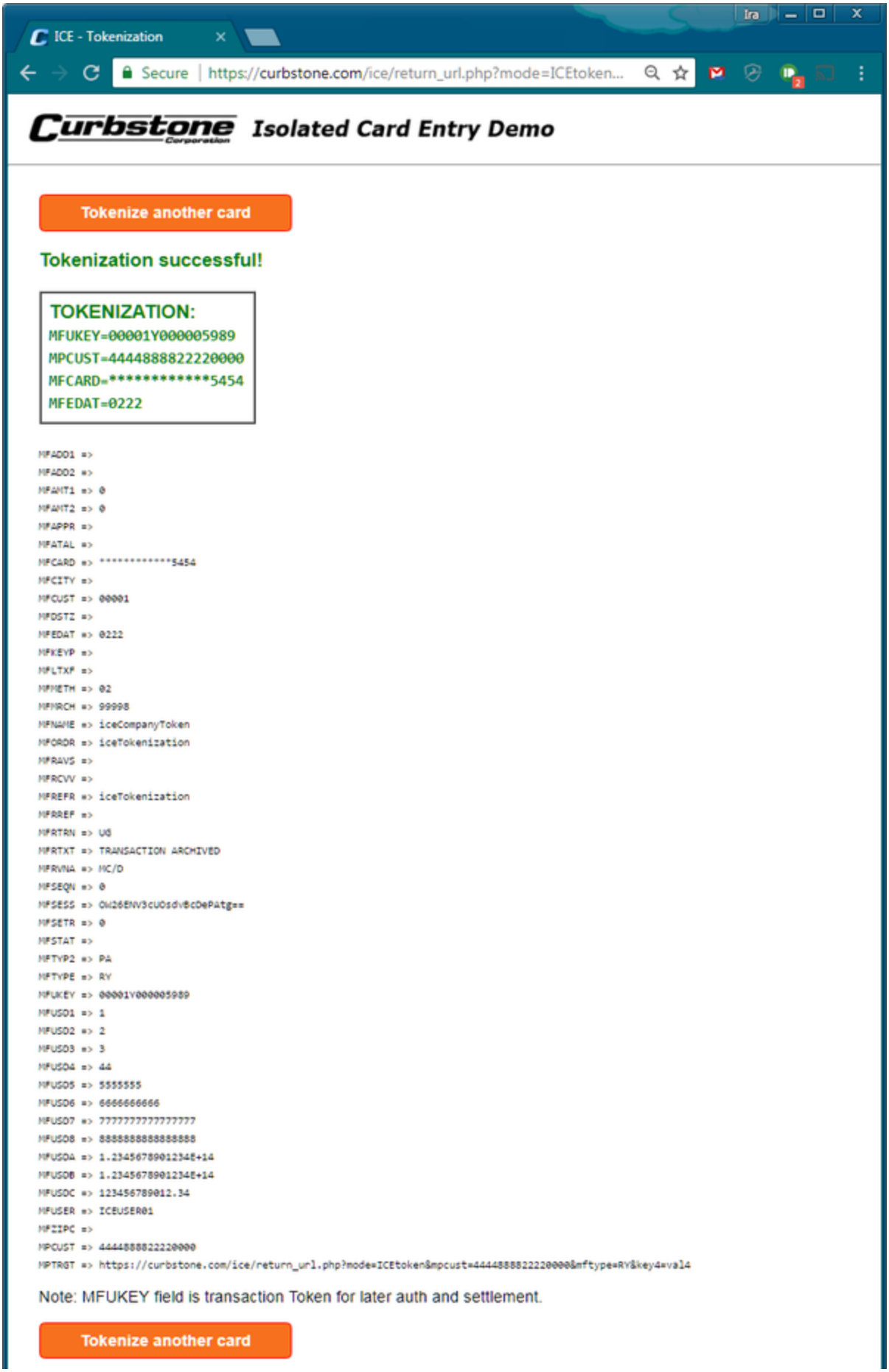
The startup screen:

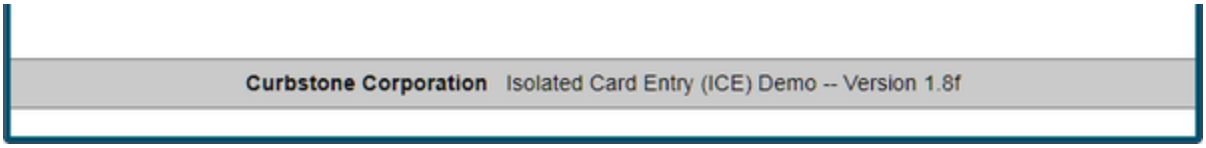


Data entry screen:



The result screen:


ICE - Tokenization
Secure | https://curbstone.com/ice/return_url.php?mode=ICEtoken...
Curbstone Corporation
Isolated Card Entry Demo
Tokenize another card
Tokenization successful!
TOKENIZATION:
MFUKEY=00001Y000005989
MPCUST=4444888822220000
MFCARD=**********5454
MFEDAT=0222
MFADD1 =>
MFADD2 =>
MFANT1 => 0
MFANT2 => 0
MFAPPR =>
MFATAL =>
MFCARD => **********5454
MFCITY =>
MFCUST => 00001
MFDSTZ =>
MFEDAT => 0222
MFKEYP =>
MFLTXF =>
MFMETH => 02
MFMRCH => 99998
MFNAME => iceCompanyToken
MFORDR => iceTokenization
MFRAVS =>
MFRCVV =>
MFREFR => iceTokenization
MFRREF =>
MFRTRN => U8
MFRTXT => TRANSACTION ARCHIVED
MFRVNA => MC/D
MFSEQN => 0
MFSESS => OW26ENV3cUOsdvBcDePAtg==
MFSETR => 0
MFSTAT =>
MFTYP2 => PA
MFTYPE => RY
MFUKEY => 00001Y000005989
MFUSD1 => 1
MFUSD2 => 2
MFUSD3 => 3
MFUSD4 => 44
MFUSD5 => 5555555
MFUSD6 => 6666666666
MFUSD7 => 7777777777777777
MFUSD8 => 8888888888888888
MFUSD9 => 1.2345678901234E+14
MFUSDB => 1.2345678901234E+14
MFUSDC => 123456789012.34
MFUSER => ICEUSER01
MFZIPC =>
MPCUST => 4444888822220000
MPTRGT => https://curbstone.com/ice/return_url.php?mode=ICEtoken&mpcust=4444888822220000&mftype=RY&key4=val4
Note: MFUKEY field is transaction Token for later auth and settlement.
Tokenize another card



The above results in the card being deposited in the Curbstone Portal as an "archived" transaction with a type (MFTYPE) of 'RY' which is designated for cards-on-file. This will be purged on the longer retention timing of (default) one year, as opposed to settled transactions that are purged by default every 60 days.

The ERP Customer number that you provided is embedded in a field in the Curbstone Data Structure on your local transaction file C3/C3FP9020 and is almost instantly available for you to bill or use.

Following is the source code for the ICE demo at
https://curbstone.com/ice

ICE Demo Source - config.ini

ICE Demo Source - config.ini

```
; Curbstone CorrectConnect
; Isolated Card Entry "ICE" Client Configuration file
; This is part of the demo code for the reference ICE client side code
; Comment lines in this file are preceded by the semicolon ";"
; https://curb911.com   Support Site
; 888-844-8533

; do not change
version=1.7

; Client base url
; This must be the working folder for the shopping cart
; from which the included "index.php" is executed.
; Trailing/ slash/ is/ required/
;
; YOUR SERVER URL
base_url = https://curbstone.com/ice/

; CURBSTONE URL pointing to Portal installation
; This will be provided by Curbstone for your implementation
; In the demo, the URL has to be
;     https://c3plp.net/curbstone/plp/
;
; C3 PORTAL URL
ice_api_url = https://c3plp.net/curbstone/plp/

; URL to which control is returned - this must be a program
; that can accept the POSTed data from the C3 portal and
; parse it.  Change this to match the program in YOUR Shopping
; Cart that will receive the response from the C3 server as to
; the success or failure of the authorization/storage request.
; This must be a fully-qualified URL that includes the filename
```

```
; of the receiving program, unless it is "index.php", in which
; case it must end/ with/ a/ trailing/ slash/!
;
; YOUR SERVER URL FOR RESPONSE
target_url = https://curbstone.com/ice/return_url.php
;
; end of file
```

17.1. ICE Demo Source - index.php

ICE Demo Source - index.php

```
<?php
/**
 * index.php   ice demo
 */

// set the displayed version number for this file
$pagevers = '1.9';

// Base Path
if (!defined('CLIENT_BASE_PATH')) { define('CLIENT_BASE_PATH', dirname
(__FILE__) . DIRECTORY_SEPARATOR); }

// Determine the include path based on the master config file
$CONFIG = parse_ini_file(CLIENT_BASE_PATH . 'config.ini');

// Specify the url of the credit card payment page
$PAYMENT_URL = $CONFIG['base_url'] . 'payment.php';

// Render the page
echo <<<OUT
<!DOCTYPE html>
<html>
  <head>
  <title>ICE - Tokenization</title>
    <meta charset="UTF-8" />
    <base href="{$CONFIG['base_url']}">
    <link rel="stylesheet" href="styles/qlient.css" />
  </head>
  <body>

  <a href="{$CONFIG['base_url']}"><img src="Curbstone-ice-demo_01.png"><
/a>
  <hr>
  <div id="main"><div id="index-container">
```

```
<strong>Tokenization Tool - enter Customer ID from your ERP</strong>
<br /><br />

<form method="post" action="$PAYMENT_URL">

Customer ID
<input type="text" name="MPCUST" size="20" maxlength="20" value="
4444888822220000" required />
  MPCUST<br />
  <div class="submit-button">
    <button type="submit">Continue</button>
  </div>
</form>

  </div></div><!-- #main -->
  <br class="clear" />
  <footer>
<strong>Curbstone Corporation</strong>   Isolated Card Entry (ICE) Demo
-- Version $pagevers
  </footer>
</body>
</html>
OUT;
?>
```

17.2. ICE Demo Source - payment.php

ICE Demo Source - payment.php

```
<?php
/**
 payment.php
 **/

// set the displayed version number for this file
$pagevers = '1.9';

// Base Path
if (!defined('CLIENT_BASE_PATH')) { define('CLIENT_BASE_PATH', dirname
(__FILE__) . DIRECTORY_SEPARATOR); }

// Determine the include path based on the master config file
$CONFIG = parse_ini_file(CLIENT_BASE_PATH . 'config.ini');

// Specify the url of the Portal initialization endpoint
$ICE_API_URL = $CONFIG['ice_api_url'];
```

```
    // this is the Customer ID from your entry
    // that is the Customer ID in your ERP software
    $MPCUST = $_POST['MPCUST'];

    // this puts the Customer# in the URL to be visible for diags
/logging
    $MPCUSF = 'mode=ICEtoken&mpcust=' . $MPCUST .
'&mftype=RY&key4=val4';

$payload = array(
    'MFADD1' => ' ',
    'MFADD2' => ' ',
    'MFAMT1' => '0.00',
    'MFAMT2' => '0.00',
    'MFCITY' => ' ',
    'MFDSTZ' => ' ',
    'MFLTXF' => ' ',
    'MFMETH' => '02',
    'MFMRCH' => '99998',
    'MFNAME' => 'iceCompanyToken',
    'MFORDR' => 'iceTokenization',
    'MFREFR' => 'iceTokenization',
    'MFSTAT' => ' ',
    'MFTYP2' => 'PA',
    'MFTYPE' => 'RY',
    'MFUSD1' => '1',
    'MFUSD2' => '2',
    'MFUSD3' => '3',
    'MFUSD4' => '44',
    'MFUSD5' => '5555555',
    'MFUSD6' => '6666666666',
    'MFUSD7' => '7777777777777777',
    'MFUSD8' => '8888888888888888',
    'MFUSDA' => 123456789012345,
    'MFUSDB' => 123456789012345,
    'MFUSDC' => 123456789012.34,
    'MFUSER' => 'ICEUSER01',
    'MFZIPC' => ' ',
    'MPCUSF' => $MPCUSF,
    'MPCUST' => $MPCUST,
    'MPTRGT' => $CONFIG['target_url'],
        'MFCUST' => '00001'
);

// Format the payload for transmission
$payload_string = '';
foreach ($payload as $key => $value) {
    $payload_string .= $key . '=' . urlencode($value) . '&';
}
rtrim($payload_string, '&');
```

```
// Open a connection with the Portal endpoint
$ch = curl_init();

// Configure the POST request
curl_setopt($ch, CURLOPT_URL, $ICE_API_URL . '?action=init');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $payload_string);

// Remove line or change value to 'true' when running in production
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
// Remove line or change value to 'true' when running in production

// Execute the POST and capture the response
$result = curl_exec($ch);

if ($result == false) {
    echo 'Curl error #: ' . curl_errno($ch) . "<br>";
    echo 'Curl error: ' . curl_error($ch) . "<br>"; }

// Close the connection
curl_close($ch);

$result = json_decode($result, true);

// Demonstrates basic success/error checking
switch ($result['MFRTRN']) {
    case 'UG':
        if (!array_key_exists('MFSESS', $result)) {
            die('Transaction session ID was not returned');
        }
        break;

    case 'UL':
    default:
        if (is_array($result)) {
            // Single-character error code
            if (array_key_exists('MFATAL', $result)) {
                echo 'MFATAL ERROR CODE: ' . $result['MFATAL'] . '<br>';
            }
            // Human-readable error message
            if (array_key_exists('MFRTXT', $result)) {
                echo 'MFATAL ERROR MESSAGE: ' . $result['MFRTXT'] .
'<br>';
            }
        }

        die('Transaction initialization failed');
        exit; }
```

```
// Extract the transaction session ID from the JSON payload
$session_ID = $result['MFSESS'];

// Build the URL to point to the Portal user interface.
// This will be embedded in the HTML of the web page.
$ICE_TXN_URL = $ICE_API_URL . '?MFSESS=' . $session_ID;

// Render the payment page.
// Most of the following HTML can be customized as needed
// The only required elements are the iFrame and JavaScript library.
echo <<<OUT
<!DOCTYPE html>
<html>
    <head>
    <title>ICE - Tokenization</title>
        <meta charset="UTF-8" />
        <base href="{$CONFIG['base_url']}">
        <link rel="stylesheet" href="styles/qlient.css" />
    </head>
    <body>

        <a href="{$CONFIG['base_url']}"><img src="Curbstone-ice-demo_01.
png"></a>
        <hr>
        <div id="main">

            <div id="payment-container">

<p>The card entry iFrame below is generated from Curbstone's Portal.</p>

<p>Use card 5454545454545454, future expiry, any security code.</p>

<p>Customer ID: <strong>$MPCUST</strong></p>

                <!-- REQUIRED -->
                <!-- iFrame MUST include an id of "plp-iframe" -->
                <iframe id="plp-iframe" style="display:none;" src="
{$ICE_TXN_URL}&mode=embedded"></iframe>

                <!-- OPTIONAL -->
                <!-- The noscript tag allows non-JavaScript users to
proceed with the payment -->
                <!-- This will display in place of the iFrame if the
user has JavaScript disabled -->
                <noscript>
                    <p style="display:block;border:1px solid red;
background-color:pink;font-weight:bold;padding:20px;">
                        JavaScript not enabled. Go to our <a href="
{$ICE_TXN_URL}&mode=redirect">payment portal</a> to complete your order.
```

```
                    </p>
                </noscript>

            </div>

            <!-- REQUIRED -->
            <!-- This JavaScript library is responsible for: -->
            <!--    1) Making the iFrame visible -->
            <!--    2) Breaking out of iFrame, redirect to the target
page (MPTRGT) on completion -->
            <script type="text/javascript" src="{$ICE_API_URL}scripts
/plp.js"></script>

        </div>
        <br class="clear" />
        <footer>
            <strong>Curbstone Corporation</strong>   (ICE) Demo --
Version $pagevers
        </footer>

    </body>
</html>
OUT;
?>
```

17.3. ICE Demo Source - return_url.php

ICE Demo Source - return_url.php

```
<?php

// set the displayed version number for this file
$pagevers = '1.9';

// Base Path
if (!defined('CLIENT_BASE_PATH')) { define('CLIENT_BASE_PATH', dirname
(__FILE__) . DIRECTORY_SEPARATOR); }

// Determine the include path based on the master config file
$CONFIG = parse_ini_file(CLIENT_BASE_PATH . 'config.ini');

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $response = $_POST;
} else {
    $response = $_GET;
}
ksort($response);
```

```
if (array_key_exists('MFRTRN', $response)) {
    // Determine if the transaction was successful
    switch ($response['MFRTRN']) {
        case 'UG':
            $confirmation_msg = 'Tokenization successful!';
            $txn_status = 'success';
            break;
        case 'UN':
            $confirmation_msg = 'Submission declined: ' . $response
['MFRTXT'];
            $txn_status = 'error';
            break;
        case 'UL':
        default:
            $confirmation_msg = 'Field Error - code: ' . $response
['MFATAL'];
            if ($response['MFATAL'] == 'H') {
            $confirmation_msg = $confirmation_msg . ' - Bad Card
Number';
            }
            $txn_status = 'error';
            break;
    }
} else {
    $confirmation_msg = '';
    $txn_status = '';
}

$token_msg .= '<table border=1 cellpadding=8><tr><td>TOKENIZATION:
<br><font face="monospace">MFUKEY=' . $response['MFUKEY'];
$token_msg = $token_msg . '<br>MPCUST=' . $response['MPCUST'];
$token_msg = $token_msg . '<br>MFCARD=' . $response['MFCARD'];
$token_msg = $token_msg . '<br>MFEDAT=' . $response['MFEDAT'] . '<
/font></td></tr></table>';

$response_fields = '<small><font face="monospace">';
foreach ($response as $key => $value) {
    $response_fields .= htmlentities($key, ENT_QUOTES) . ' => ' .
htmlentities($value, ENT_QUOTES) . '<br> ';
}
$response_fields .= '</font></small>';

echo <<<OUT
<!DOCTYPE html>
<html>
<head>
    <title>ICE - Tokenization</title>
    <meta charset="UTF-8" />
    <base href="{$CONFIG['base_url']}">
```

```
    <link rel="stylesheet" href="styles/qlient.css" />
</head>
<body>
<a href="{$CONFIG['base_url']}"><img src="Curbstone-ice-demo_01.png"><
/a>
<hr>
<div id="main">
    <div id="results-container">
        <a class="submit-button" href="{$CONFIG['base_url']}">Tokenize
another card</a>
        <h3 class="$txn_status">$confirmation_msg</h3>

        <h3 class="$txn_status">$token_msg</h3>

        $response_fields

<p>Note: MFUKEY field is transaction Token for later auth and
settlement.</p>

        <a class="submit-button" href="{$CONFIG['base_url']}">Tokenize
another card</a>

    </div>
</div>
<footer>
    <strong>Curbstone Corporation</strong>   Isolated Card Entry (ICE)
Demo -- Version $pagevers
</footer>
</body>
</html>
OUT;
?>
```

18. C3 "Check Operation" Utility

Curbstone provides a customer utility that you can use to confirm the portal can be reached. It is a quick test to see that the C3 server is online and functional. Click on this link:

https://curb911.com/subscriber-test/

19. **FAQ**

Q1. Is there JS API available to manipulate iFrame? The document mentions some event handlers, but we need full details. Is it possible to manipulate the submission? If not then there would be a UI constraint that submit button must be within iFrame

A1 - Due to the same origin policy*, it is not possible to manipulate the contents of the iFrame, since the host page and iFrame source live on different domains. However, Curbstone does provide a limited set of "hooks" that allow the shopping cart to be notified via JavaScript when certain events occur within the iFrame.

This includes a hook that allows the shopping cart to receive a message (within JS) whenever the form is submitted, however, there is no way for the shopping cart itself to trigger the submission.

Review these docs to learn more about the JavaScript hooks:
*Same Origin Policy: https://en.wikipedia.org/wiki/Same-origin_policy

Q2. If submit button must be within iFrame then we would need to have another button to submit other payment types or page changes, which might require page UI redesign
A2 - The submit button has to be within the iFrame.

Q3. Can we style iFrame content so that it matches hosting page styles completely, and the user might not even be aware of it?
A3 - Yes, but this has to be done ahead of time in conjunction with Curbstone since these custom styles have to reside on Curbstone's servers (i.e. the host page cannot reach into the iFrame in order to style it on page render).

Q4. Is it acceptable to complete the transaction but not complete the order? For example, an error after redirecting back, or UI flow constraints if the iFrame is located not on the last page in checkout?
A4 - It is certainly possible, though I can't say whether or not this is "acceptable". For example, if the customer goes through the steps of submitting a transaction via the PLP iFrame, and the response / redirect occurs but triggers some sort of error in the shopping cart (e.g. if there is a bug in the shopping cart logic, etc.), then the actual credit processing transaction would have already completed, but perhaps from the standpoint of the shopping cart the order has not completed. Much about these sorts of scenarios are dependent on the particulars of the shopping cart's checkout flow and the specifics of how the PLP integrator has integrated their shopping cart with PLP.

Q5. What if Curbstone is unavailable or transaction keeps erroring? What about local failover to some alternate processor?
A5 – The PLP integrator is the only party involved at that point, who can do anything to gracefully handle the situation is the shopping cart itself. The onus of handling PLP error scenarios falls on the client.

Q6. Is the iFrame responsive/mobile friendly? Is it possible to have multiple different custom styles?
A6 - The relevant first question here is whether or not the shopping cart is responsive / mobile-friendly. The shopping cart can directly control the dimensions of the iFrame container as well as its placement on the page, so it is likely that PLP can be integrated in a mobile-friendly manner.

Each unique MFCUST+MFMRCH combination can have its own set of styles. It is not possible for the same MFCUST+MFMRCH combo to have multiple different sets of styles.

Q7. Is it OK to generate a new session ID every time the customer takes other action on the checkout page (shipping, insurance, etc) resulting in changes to the total amount?
A7 – This is not the preferred approach, as it burdens the PLP server with (likely) avoidable traffic and processing. This isn't a problem technically. However, the shopping cart has to keep track of the most current session ID, and each time the Session ID changes the cart has to regenerate the iFrame, either dynamically with JavaScript or by triggering a full page refresh. This can be done dynamically with JavaScript by doing something like:

```
document.getElementById("plp-iframe").src = "https://c3plp.net/curbstone/plp/?MFSESS=" + new_session_ID;
```

Note: JavaScript would first have to be used to retrieve a new session ID, which the example above stores in a variable named "new_session_ID".

Note: multiple approaches could use JavaScript for retrieving a new session ID. Due to browsers' cross-origin policy, it would likely be easiest to trigger an AJAX call to an endpoint on the shopping cart's same domain, which would in turn call out to PLP, retrieve the new ID, and ferry it back to the browser as the AJAX response.

Q8. What is a partial authorization and how is it handled?
A8 – A partial authorization occurs when the available credit balance on an account is less than the amount of the auth request and it is approved for the lesser amount. This will depend on the type of card used. The amount will be returned and replace what was sent in field MFAMT1. There is no "indicator" returned which identifies the transaction as being partially authorized. It's up to you to check the MFAMT1 sent against the MFAMT1 returned and take the appropriate action, i.e.: reject the auth and void it, or accept it and make arrangements with your customer to pay the additional funds due. If you do not wish to support partial authorizations at all, you will need to speak with your acquirer to see if they can deactivate it on your MID account. We cannot deactivate it in the C3 software.

All source code is available in a ZIP file posted on the support site.

Address questions to your Support Center account at **https://curb911.com**.

###