ACI Commerce Gateway™ Hosted Payment Page Guide

Publication Number: Product # to be registered.

Print Record

Date	Hard Copy Version	Software Version	Reason for New Version
09/26/2002	Version 1.0	Commerce Gateway 3.0	Original documentation.

Contents

Section 1 - Preface

This guide serves as a collection point for all **Hosted Payment Page** details.

Audience

The *ACI Commerce Gateway Hosted Payment Page* is intended to provide a complete understanding of the hosted payment page philosophy and the implementation provided by ACI. Divided into distributable sections, this guide begins with a high level description for the systems architect. The focus then narrows to step-by-step directions for system administrators installing the Commerce Gateway application and configuring the servers. This guide also offers appendices that discuss related issues.

Prerequisites

See the *ACI Commerce Gateway 3.0 Installation Guide* for hardware and software requirements.

Publication Identification

Two entries appearing at the bottom of each page uniquely identify this ACI publication. The publication number (for example, EC-WA100-06 for the ***eCommerce Wallet Database Schema Manual***) appears on every page to assist readers in applying updates to the appropriate manual. The publication date (for example, 08/96 for August, 1996) identifies the last time material on a particular page has been revised. This date can be used to verify that the reader has the most current information and is particularly helpful when dealing with HELP24 to answer operating questions.

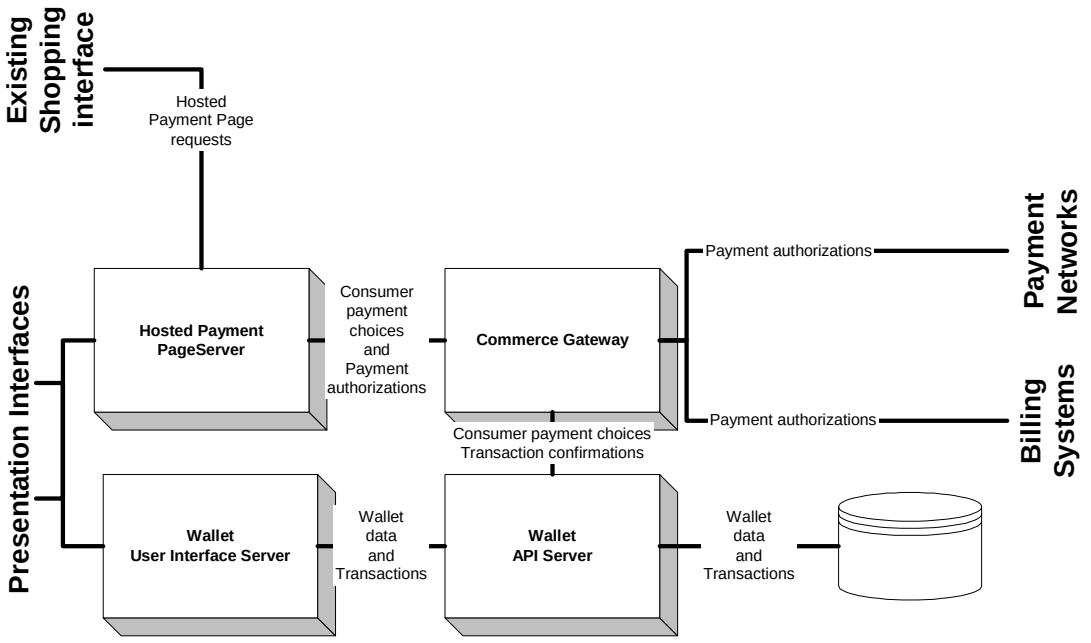
The print record, located on the page immediately following the title page is also helpful when verifying that a manual contains the most current information. An entry is added to the print record each time an ACI publication is modified in any way. Each print record entry includes the date of the change, a unique version number, and the type of change (new, update, or reissue). Version 1 is assigned to a *new* manual. Later versions can be updates or reissues, depending on the changes being made. An *update* typically involves changes on a small number of pages throughout the manual. A *reissue* occurs when all pages of the manual are replaced because changes affect a large number of pages.

Section 2 - Solution Overview

ACI Secure Commerce Suite

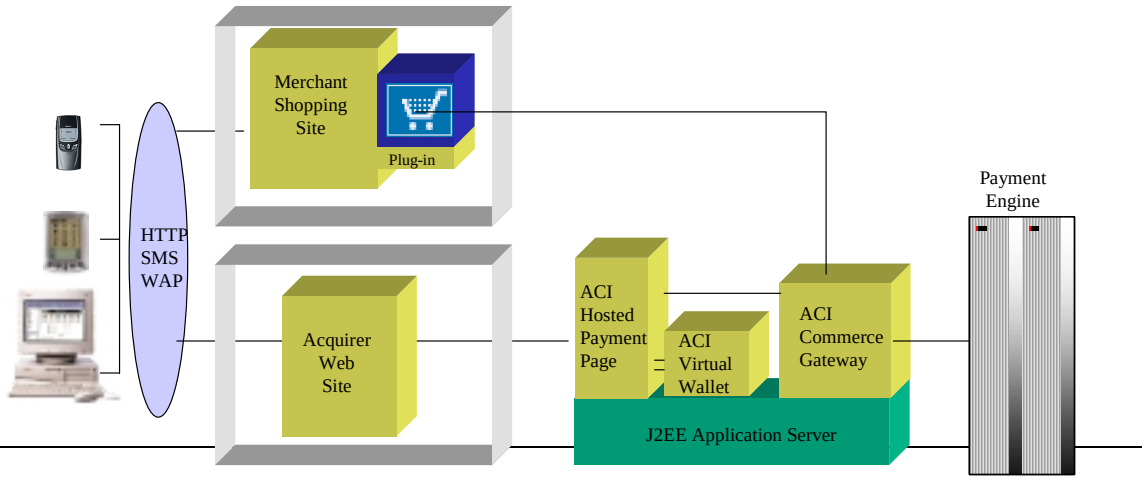
The **Hosted Payment Page** is the most visible piece of the ACI Secure Commerce Suite, providing the critical interface between merchants, consumers, and the payments processing channels of the ACI Commerce Gateway. The hosted payment page is a dynamically generated payments entry interface that replaces dated card data entry screens found at merchant Web sites. The hosted payments page also interacts with the ACI Virtual Wallet for an enhanced shopping experience. When delivered as a package, a consumer can efficiently select payment options from their customized portfolio payment options. This provides the ultimate in efficient, consistent shopping for consumers that shop at merchant Web sites utilizing the ACI Secure Commerce Suite.

The following diagram shows how the hosted payment page fits into a sample configuration of the ACI Secure Commerce Suite. The initial consumer interaction takes place through the pieces on the left side of the diagram. They interact with merchant shopping pages and then use the hosted payment page for selecting the optimal payment mechanism. As the transaction is processed, required data is compiled from the Virtual Wallet Server, checked for fraud in the Commerce Gateway, and then transmitted to the issuing systems for authorization. After the purchase, the consumer can visit the Virtual Wallet data maintenance presentation interface for viewing receipts.



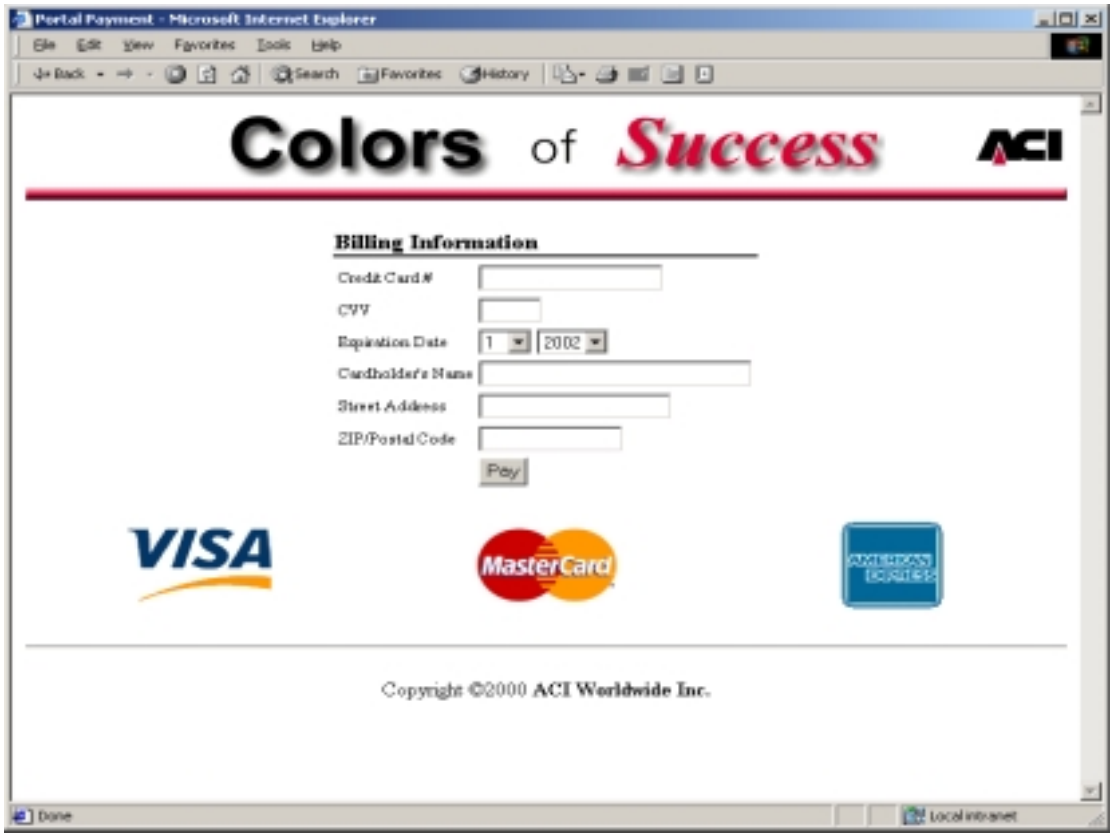
Architecture

The ACI Secure Commerce products support a common Java 2 Enterprise Edition (J2EE) application framework ensuring portability and scalability across multiple platforms and operating systems. This helps the acquirer to gain benefits from a rapid deployment of new business functionality on a common technology platform. The solution also provides a platform that allows multiple co-existing application packages, thereby reducing the cost of providing new applications within the ACI Secure Commerce Suite.



Hosted Payment Page

The ACI **Hosted Payment Page** is a dynamic payments entry interface that integrates with existing merchant shopping sites to expand their current payments processing options. When a consumer selects "Buy" on a merchant's shopping interface, processing is turned over to the hosted payment page where supported payment processing features are offered to the consumer through a seamless URL redirection. Once the payments portion of the shopping experience is completed, the consumer is returned to the merchants shopping site for receipt presentation. The **Hosted Payment Page** provides a consistent interface supporting the acquirer's current requirements for payments processing while offering the ability to implement future payment schemes without costly merchant interface changes.



Section 3 - Processing Flows

Overview

This section details the payment processing involving the hosted payment page.

The Consumer Point of View

The consumer makes a purchase at a merchant Web site:

- Selects merchandise.
- Enters shipping details and click **Buy**.
- Redirected to Commerce Gateway Hosted Payment Page.
- Enters payment details and click **Pay**.
- Redirected to merchant-specified URL.

The Merchant Point of View

The merchant receives a purchase order from a consumer:

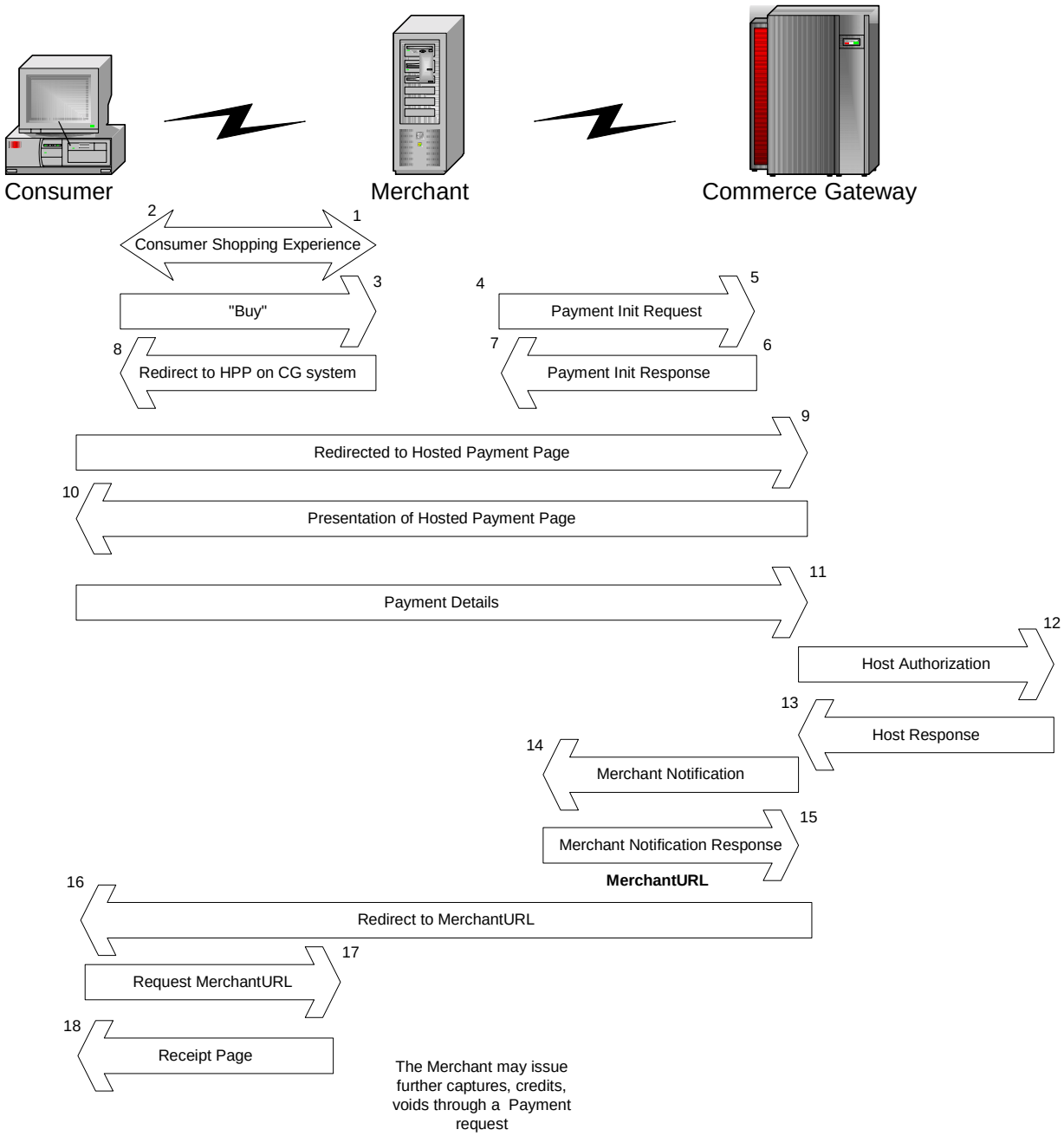
- Sends payment initialization message to the Commerce Gateway.
- Receives a payment ID and payment URL.
- Redirects consumer to the Hosted Payment Page URL.
- Receives notification of transaction results.
- Responds with URL of a receipt page.
- Presents receipt page to consumer.

The Commerce Gateway Point of View

The application receives a payment initialization message from a merchant.

- Respond with payment ID and Hosted Payment Page URL.
- Presents Branded Payment Page to the consumer.
- Receives billing details from the consumer.
- Processes the transaction.
- Notifies the merchant of transaction results.
- Receives redirect URL from the merchant.
- Redirects consumer to redirect URL.

Detailed Message Flow



Consumer Browser	Merchant Web Site	ACI Commerce Gateway	Payment Engine
1. Request shopping cart page	2. Return shopping cart page		
3. Click Buy button.	4. Format HTTP payment init request to ACI Commerce Gateway, including transaction details.		
	5. Send POST to ACI Commerce Gateway.	6. Save transaction details. Generate PaymentID.	
		7. Return PaymentID and Hosted Payment Page URL to merchant.	
	8. Save PaymentID. Return page back to consumer containing META redirect to Hosted Payment Page URL and PaymentID.		
9. META redirect to Hosted Payment Page URL – no user action required.		10. Return payment page to consumer. PaymentID is included on page.	

11. Enter payment information and click **Submit** to send. PaymentID is included on page.		12. Retrieve saved transaction details. Submit payment for authorization. 14. Send POST containing PaymentID and transaction status to Mechant Web Site, using a predefined URL.	13. Process transaction. Return status.
	15. Save transaction status. Return receipt page URL.	16. Return page back to consumer containing META redirect to receipt page on Merchant Web Site.	
17. META redirect to receipt page URL – no user action required.	18. Retrieve transaction details and return receipt page to consumer.		

The Consumer Browser is launched and has a session established (#1 & #2) with the Merchant Web Server. The Merchant Web application ("the shopping experience") can be written in ColdFusion, Java Server Pages (JSPs), or even Microsoft's Active Server Pages (ASPs). At some point, the consumer clicks on a "Buy" button (#3). This will cause the Merchant Web application to invoke the ACI e24PaymentPipe plug-in.

The e24PaymentPipe will then take information from the Merchant Web application and format and send a HTTP/HTTPS request to the ACI Commerce Gateway (#4 & #5). The Commerce Gateway performs the Transaction Initialization processing and generates a PaymentID (#6). This PaymentID is used as a key to insert payment initialization data into the PaymentLog table and is also sent back on the response (#7) to the e24 PaymentPipe. The e24 PaymentPipe parses off the returned data (PaymentID & Payment Page URL) and gives it to the Merchant Web application. At this point, the Merchant Web application should store the PaymentID that belongs to this particular session with the Consumer. The Merchant Web application will then issue a META statement on the response to the Consumer Browser (#8). This META statement will contain the PaymentID and the Payment Page URL and will cause the Browser to send a request for the Payment Page to the specified URL (#9). The Consumer does not see this happening and is not prompted for any action. All they know is that they have just clicked the "Buy" button. The Commerce Gateway hosting the Payment Page URL will then return the Payment Page information (#10).

The Consumer then enters payment information into the Payment Page form presented and clicks on the "Pay" button (#11). In addition to the Consumer's payment information, the PaymentID is also sent on this request to the Commerce Gateway. The Commerce Gateway uses the PaymentID to read up the Payment Log and then starts to process the transaction, which will now include Merchant transaction information as well as Consumer payment information. Commerce Gateway will then send the transaction off to a host-processing environment for authorization and a response will be returned (#12 & #13).

At this point, the Consumer Browser has a session established with the Commerce Gateway. We now need to "redirect" the Consumer Browser to the Merchant's Receipt page. First the Commerce Gateway acts as a client and sends a HTTP/HTTPS request to the Merchant Web application (#14). This is a separate session than the one with the Consumer Browser. The request message contains the PaymentID and all of the transaction response data that Commerce Gateway received from the host-processing environment. At this point, the Merchant Web application will need to store the transaction response information and send back a reply that contains the URL to the Merchant's receipt page (#15). Second, the Commerce Gateway will then issue a META statement on the response to the Consumer Browser. This META statement will contain the PaymentID and the Receipt Page URL and will cause the Browser to send a request for the Merchant's Receipt page to the specified URL (#16 & #17).

At this point, the Merchant Web application will use the PaymentID to read up the transaction response information and then display that back to the Consumer Browser (#18).

Hosted Payments Page Key Benefits

Secure

- Consumer talks directly to Acquirer over SSL.
- Only Consumer/Gateway share payment details; the merchant will not see the card information.

Configurable

- Merchant requires no changes to support new payment methods (i.e. Verified by VISA, UCAF)
- Branding of the payments page is configurable at the Institution or Merchant level

Efficient

- When implemented with the Virtual Wallet the consumer only has to select the payment instrument they wish to use. There are no lengthy forms to fill out.

Section 4 - Commerce Gateway Configuration

The Commerce Gateway will need to be configured to support the hosted payment page. Specifically, the following items need to be addressed

- Deploy the Hosted Payment Page servlet
- Configure the Web server presenting the Hosted Payment pages
- Configure the institution branding files
- Configure the merchant branding files
- Configure the terminal payment schemes

Deployment

The hosted payment page servlet and payment pages (Payment.jsp, PostPayment.jsp) are deployed as pieces of the Commerce Gateway application. Refer to the *ACI Commerce Gateway™ Installation Guide* for specifics on installation.

Configure the Web Server

Create the Branding Virtual Directories

The hosted payment page's Java Application server will build the payment pages from a collection of format and image files. The following configuration provides a sample structure of how the directories under a "branding" virtual directory might be configured.

- website\branding\institution1\
- website\branding\institution1\images
- website\branding\institution1\merchant1\

- website\branding\institution1\merchant1\images
- website\branding\institution1\merchant2\
- website\branding\institution1\merchant2\images
- website\branding\institution2\
- website\branding\institution2\images
- website\branding\institution2\merchant1\
- website\branding\institution2\merchant1\ images
- website\branding\institution2\merchant2\
- website\branding\institution2\merchant2\ images

The branding directories do not need execute permission. “Read Only” permission will be the most secure.

Institution Branding Files

Creating Institution Brand Files

Sample brand files are included on the installation CD in the directory.

The Header Footer and style sheet for a hosted payment page can be altered as required to match the formats, color schemes and styles desired. Store the altered pages in the institution virtual directory under the Hosted Payment Page Web server.

Implementing Institution Brand Files

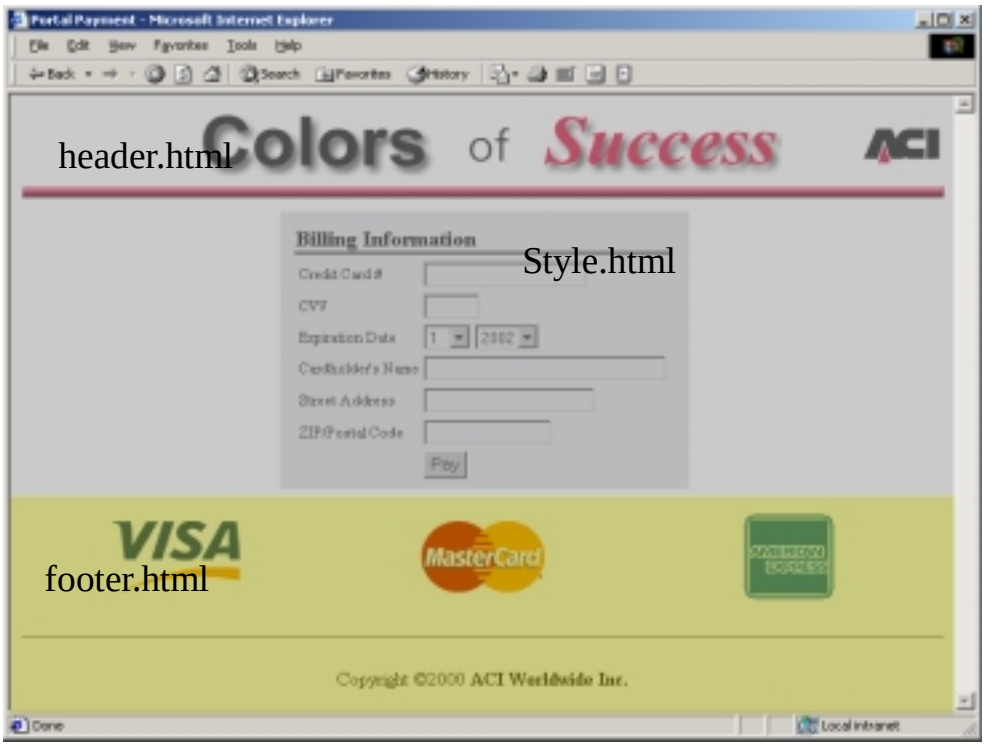
On the Commerce Gateway administrator interface, the Orders menu, the Institution Branding entry allows configuration of the header, footer, and style sheet for an institution’s merchants that do not have any overriding configuration values.

Configure the Merchant Branding Files

Creating Merchant Brand Files

Sample brand files are included on the installation CD directory.

The Header Footer and style sheet for a hosted payment page can be altered as required to match the formats, color schemes and styles desired. Store the altered pages in a virtual directory under the Hosted Payment Page Web server.



ACI suggests creating a brand structure that looks like the following:

website\branding\institution1

- inststyle.css
- instheader.html
- instfooter.html

website\branding\institution1\merchant1

- merchstyle.css
- merchheader.html
- merchfooter.html

website\branding\institution1\merchant2

- merchstyle.css
- merchheader.html
- merchfooter.html

Implementing Merchant Brand Files

On the Commerce Gateway **Administrator interface→Orders→Add Merchant Branding** entry allows configuration of the header, footer, and style sheet for a specific merchant. Values entered on this screen override the values entered for the merchant's institution.

Sample brand files are included on the installation CD in the directory.

Configure the Terminal Payment Schemes

The payment options presented on the hosted payment page are controlled by the Payment Instruments selection on the Terminal configuration screens. For example, removing the VISA 3D-Secure option will prevent the terminal from processing using Verified by VISA logic.

Section 5 - Merchant Integration

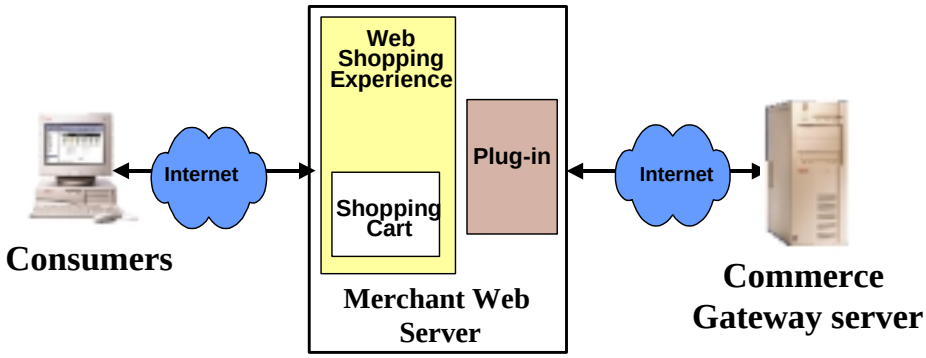
Overview

The ACI Commerce Gateway application provides three merchant plug-ins that integrate easily with existing merchant shopping portals.

The three plug-ins are:

- Hosted Payment Page (e24PaymentPipe)
- Credit Card Transaction Processing (e24TranPipe)
- Batch Maintenance (e24BatchPipe)

The plug-ins are designed to be compatible with shopping sites designed with Java, C/C++, ColdFusion, ActiveX, VB, COM, and ASP. All messages to the merchant are handled via the plug-ins, with the exception of the Merchant Notification messages used to redirect the consumer's browsers.



This section details the Hosted Payment Page plug-in (e24PaymentPipe) and the messages processed on the Hosted Payment Page.

In a standard purchase process, the merchant follows these steps:

- Merchant receives order from consumer.
- Sends Payment Initialization Message to Commerce Gateway via the hosted payment page plug-in (Payment Init). If any errors are encountered with the plug-in they are handled at this time.
- Receives a Payment ID and Payment URL from the plug-in (Payment Init response).
- Redirects consumer to Payment URL hosted on the Commerce Gateway, passing along the Payment ID as a parameter.
- Once the payment has been completed on the Hosted Payment Page, the merchant Receives Notification of Transaction Results (Merchant Notification). The results are stored and analyzed to determine which URL the consumer should see for their receipt.
- Responds to Commerce Gateway with URL of Receipt Page (Merchant Notification Response)
- Presents Receipt Page to consumer
- The transaction is later completed (captures, credits, voids) via a Payment message.

For a detailed description of the Merchant Integration process, please refer to the *ACI Commerce Gateway™ Merchant Integration Guide* when it is released.

Supported Web Site Formats

The plug-ins folder on the installation CD holds the code for the supported plug-in formats

- Active Server Pages (ASP)
- ColdFusion,
- ActiveX/COM
- Visual C/C++

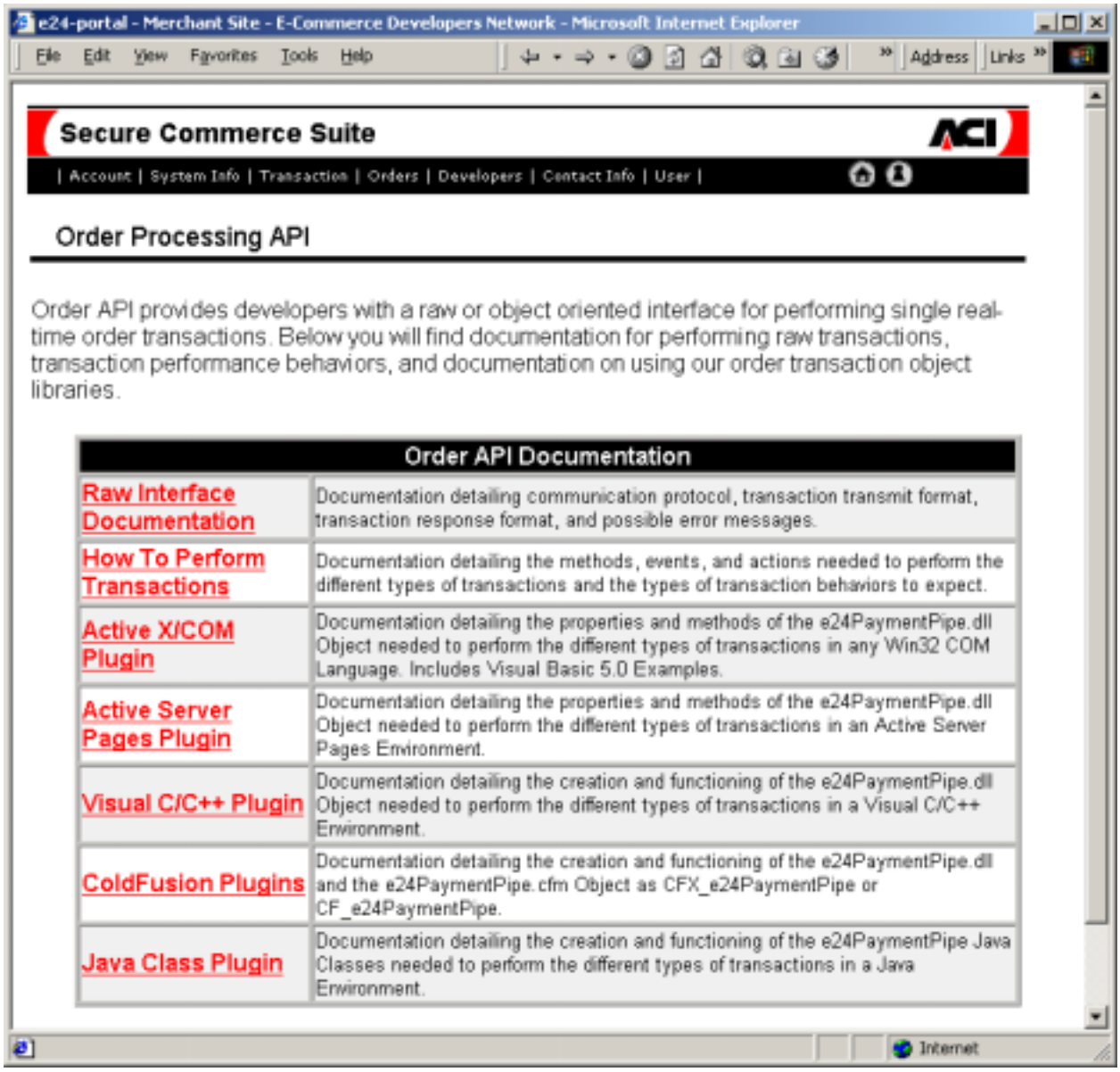
The readme.txt file in each respective folder provides information pertinent to the plug-in format selected.

e24PaymentPipe methods

The e24PaymentPipe uses the following methods to set, initiate and get data within a message.

- GetAction, Amt, Auth, Avr, CurrencyCode, Date, DebugMsg, ErrorMsg, ErrorURL, Id, Language, Password, PaymentId, PaymentPage, Port, RawResponse, Ref, ResponseURL, Result, TrackId, TransId, Udf1, Udf2, Udf3, Udf4, Udf5, WebAddress
- performPaymentInitialization
- SetAction, Amt, Context, CurrencyCode, ErrorURL, Id, Language, Password, PaymentId, PaymentPage, Port, ResponseURL, SSL, TrackId, TransId, Udf1, Udf2, Udf3, Udf4, Udf5, WebAddress

To view the details on a specific message's interface and field structure visit the developers menu on the Commerce Gateway Merchant console and select the orderapi entry. A sample merchant interface can be viewed at http://portal.e24.net/Merchant/loginEntry.jsp, login with a institution of ???, MerchantID of ???, and a password of ???



Hosted Payment Page Message Details

The following messages are the communications supported by the Hosted Payment page plug-in (e24PaymentPipe) for interaction with the ACI Commerce Gateway PaymentInitHTTPServlet. These messages are sent via HTTP using a format similar to http://CommerceGatewayURL/servlet/PaymentInitHTTPServlet. The value for the CommerceGatewayURL is configured in the Institution Profile file.

Payment Init Request

This message is created and sent by the Payment Plug-in to the Commerce Gateway to start a payment transaction. It utilizes the following elements:

- Tran Portal ID
- Tran Portal Password
- Action Code (1=Purchase, 4=Authorization)
- Amount
- Currency Code
- Consumer Language
- Merchant Notification URL
- Error URL
- Track ID
- User Defined Fields (UDF1 - UDF5)

Specifics on the message format will be made available at a future date.

Payment Init Response

This message is a response to the Payment Init Request. The Commerce Gateway generates and returns it to the Payment Plug-in. The merchant should log the payment ID for later use when redirecting the consumer to the Payment URL. It utilizes the following elements:

- Payment ID
- Payment URL

Payment Request

This message is created and sent by the Payment Plug-in to the Commerce Gateway to continue (captures, credits, voids) an initialized payment transaction. It utilizes the following elements:

- Payment ID
- Original Transaction ID
- Tran Portal ID

- Tran Portal Password
- Action Code (1=Purchase, 4=Authorization, etc)
- Amount
- Track ID
- User Defined Fields (UDF1 - UDF5)

Commerce Gateway Messaging to the Merchant

The Commerce Gateway initiates communications with the merchant using a POST request. It does not use the payment plug-in when initiating communications. The Commerce Gateway communicates with the Merchant via the URL configured in the Payment Init files.

Merchant Notification Request

The Commerce Gateway sends this message to the merchant server once the consumer has submitted their payment details on the Payment URL. The merchant should store the transaction information and analyze the Result code to determine the status of the transaction. Based upon the status the merchant should then send the response to the Commerce Gateway with an appropriate receipt URL. It utilizes the following elements:

- Payment ID
- Transaction ID
- Result Code
- Auth Code
- Post Date
- Merchant Track ID
- User Defined Fields (UDF1 - UDF5)

Merchant Notification Response

The Payment Plug-in sends this message to the Commerce Gateway after determining which receipt URL the consumer should be sent to. It utilizes the following elements:

- Receipt URL

Section 6 - Testing Commerce Gateway Installation

This section of the document details the installation and processing of a sample merchant site on a Windows server that uses the hosted payment page presented on an exiting Commerce Gateway.

Merchant Server Prerequisites

Java Developer Kit

1. Pull the file jdk-1_3_1_x-win.exe from http://java.sun.com/j2se/1.3/
2. Perform a default installation on your root drive.

Get Latest Tomcat Version

1. Visit http://jakarta.apache.org
2. Select the **Download → binaries** link on the left side of the page.
3. Under the release builds section, select the most recent Tomcat version.
4. In the **bin** folder select the jakarta-tomcat-4.1.10.exe to download and then execute.
5. Accept default installation values until the **Installation Options** screen. On this screen select the run at NT service option.
6. Continue accepting the defaults. When prompted, **8080** for the default port, **admin** for Username, and leave the password blank.

Install the Demo

1. Copy the **\CommerceGateway\Samples\Merchant\MerchantDemo.war** file to the webapps directory under the Tomcat base installation directory. (**C:\Program Files\Apache Group\Tomcat 4.1\webapps**).

2. Restart the Apache Tomcat application/service.

3. Start the Commerce Gateway.

Configure the Merchant

1. Navigate to http://localhost:8080/MerchantDemo/index.jsp

2. Select the **Configure Demo** link at the bottom of the window. This will let you configure the sample merchant's parameters described below. Sample values based on a default Commerce Gateway installation are shown in parenthesis.

   Gateway Server Context – The URL Context of the Commerce Gateway application (/Gateway)

   Gateway Server Name – The DNS name or IP address of the Commerce Gateway server (CGservername)

   Terminal Password – The password for the terminal (password)

   Currency Code – The transaction's currency code (840)

   Merchant Notification URL – The URL to which Merchant Notification messages will be sent (http://merchantservername:8080/MerchantDemo/servlet/NotifyServlet)

   Gateway Server Port – The port the Commerce Gateway server listens on (8000)

   Consumer Language – The language of the consumer... used to determine Locale of hosted payment page (USA)

   Merchant Error URL – The URL to which Consumers will be sent in the case of an error. (http://merchantservername:8080/MerchantDemo/error.jsp)

Tran Portal ID – The TranPortalID of the terminal (1000)

Use SSL – Send transaction over SSL (0)

Action Code – The action code for this transaction (1)

3. Select **Update** and then make a purchase.

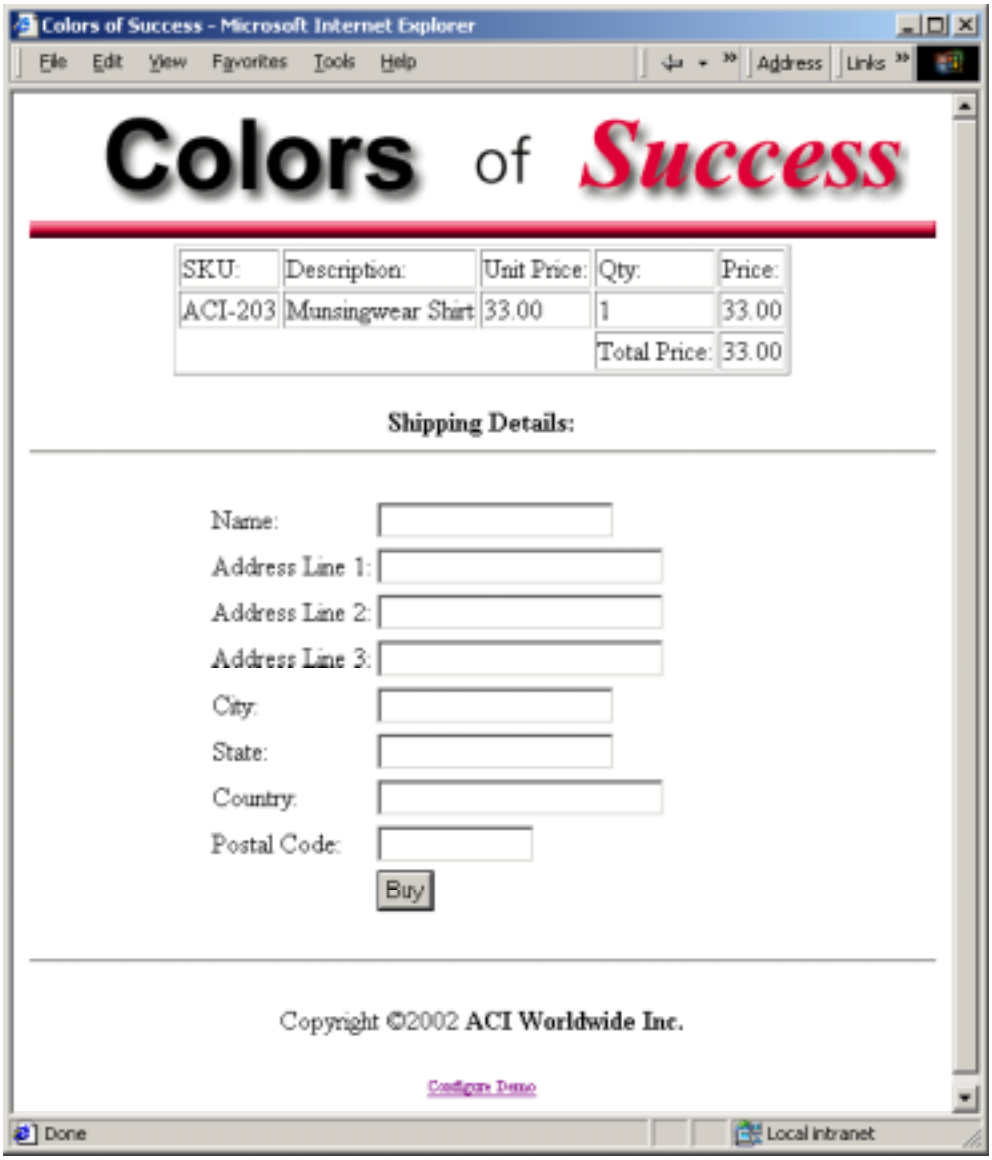
Troubleshooting

Errors will appear in the C:\Program Files\Apache Group\Tomcat 4.1\logs\stdout.log if you are running as a service, otherwise they will display on the console.

Understanding the Code

The deployed .war file is nothing but a fancy zip file. You can view the contents using an unzip tool to uncover the magic.

- This screen is an example of a merchant's shopping cart.



- Selecting the **Buy** button causes the buy.jsp page to execute.

Buy.jsp – This file instantiates the e24PaymentPipe, passes the data to the plug-in, and parses the response. It appends the payment ID to the returned URL and redirects the browser to the Hosted payment page URL.

A sample skeleton of the source follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<META name="GENERATOR" content="IBM WebSphere Studio">
</HEAD>
</HTML>
<%@ page language="java" session="true" %>
<%@ page import="com.aciworldwide.demo.*" %>
<%@ page import="e24PaymentPipe" %>
<%@ page import="java.util.Date" %>
<%@ page import="java.util.Random" %>

<%
Random rnd = new Random(System.currentTimeMillis());
String sku = request.getParameter("SKU");
String desc = request.getParameter("NAME");
String qty = request.getParameter("QTY");
String amount = request.getParameter("AMOUNT");
String trackid = String.valueOf(Math.abs(rnd.nextLong()));
String details = sku + "#" + desc + "#" + qty;

String name = request.getParameter("name");
String addr1 = request.getParameter("addr1");
String addr2 = request.getParameter("addr2");
String addr3 = request.getParameter("addr3");
String city = request.getParameter("city");
String state = request.getParameter("state");
String country = request.getParameter("country");
String postalCd = request.getParameter("postalcd");

// build Payment Init message
e24PaymentPipe pipe = new e24PaymentPipe();
pipe.setSSL( Integer.parseInt(Config.getProperty("gateway.ssl", "0")) );
pipe.setWebAddress( Config.getProperty("gateway.server.name") );
pipe.setPort( Config.getProperty("gateway.port") );
pipe.setContext( Config.getProperty("gateway.context", "") );
pipe.setId( Config.getProperty("gateway.id") );
pipe.setPassword( Config.getProperty("gateway.password") );
pipe.setAction( Config.getProperty("tran.action") );
pipe.setCurrency( Config.getProperty("tran.currency") );
pipe.setLanguage( Config.getProperty("consumer.language") );
pipe.setResponseURL( Config.getProperty("merchant.notifyURL") );
pipe.setErrorURL( Config.getProperty("merchant.errorURL") );
pipe.setAmt(amount);
pipe.setTrackId(trackid);
pipe.setUdf1(details);

// send the Payment Initialization message
if(pipe.performPaymentInitialization() != pipe.SUCCESS) {
     System.out.println("Error sending Payment Initialization Request: ");
     System.out.println(pipe.getDebugMsg());

    response.sendRedirect( response.encodeRedirectURL("error.jsp") );
    return;
}

System.out.println(" PaymentInit Sent and Response Received.");

// get results
```

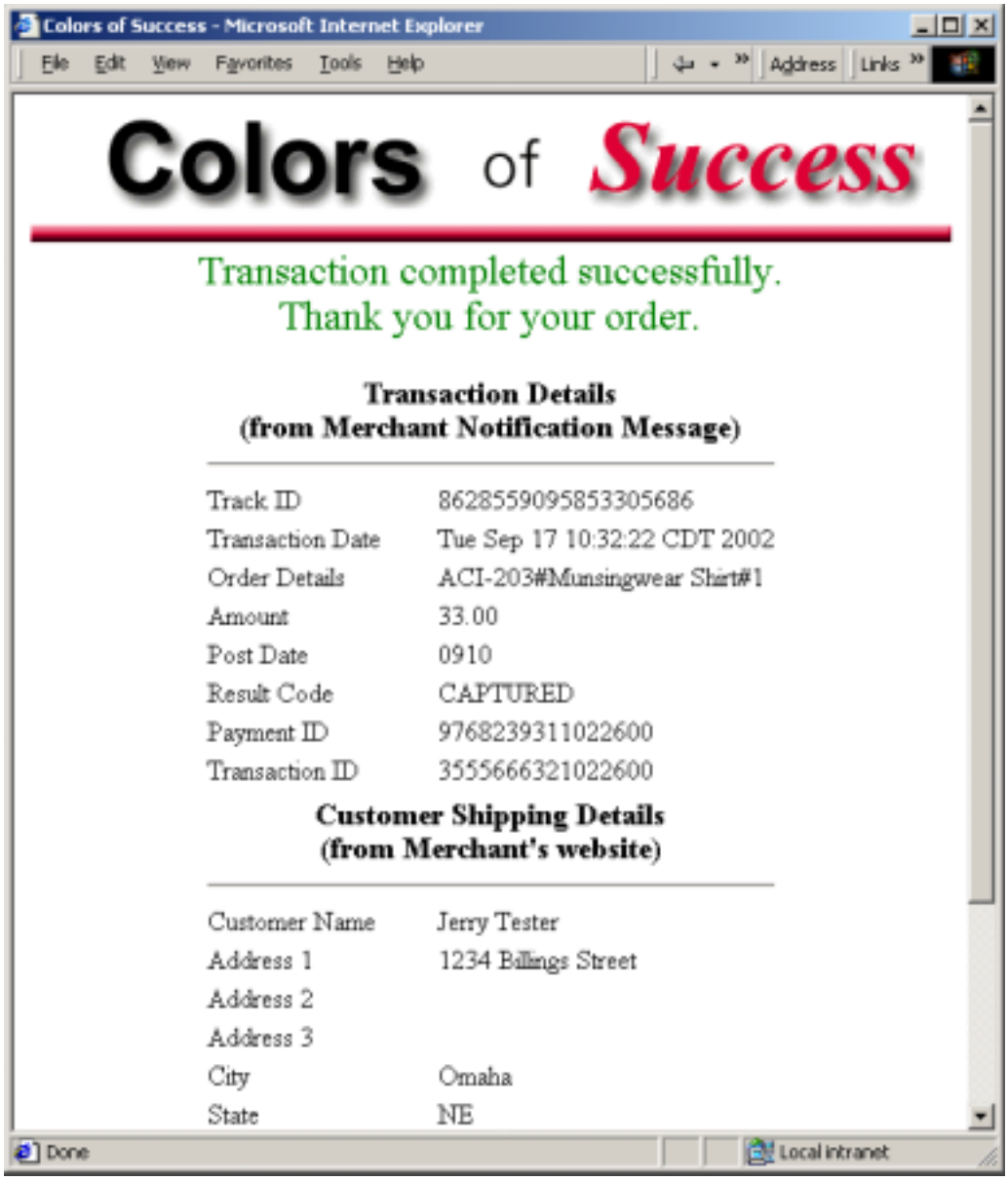
```
String payID = pipe.getPaymentId();
String payURL = pipe.getPaymentPage();

// build Order
Order order = new Order();
order.setPaymentID(Long.parseLong(payID));
order.setAmount( Double.parseDouble(amount) );
order.setOrderDetails( details );
order.setTrackID( trackid );
order.setTranDate(new Date().toString());
order.setName(name);
order.setAddr1(addr1);
order.setAddr2(addr2);
order.setAddr3(addr3);
order.setCity(city);
order.setState(state);
order.setCountry(country);
order.setPostalCode(postalCd);

// insert row into database
Orders orders = new Orders();

if ( !orders.add(order) ) {
     response.sendRedirect( response.encodeRedirectURL("error.jsp") );
     return;
}
response.sendRedirect( payURL + "?PaymentID=" + payID );
%>
```

WEB-INF\classes\com\aciworldwide\demo**NotifyServlet.class** – This servlet receives the Merchant Notification Request, archives the message details in the merchant's existing order processing system, and generates the receipt URL. A sample skeleton of the source follows:

```
/**
 * ---- Important Notice -----
 * This code is for demonstration purposes ONLY.  It is meant to
 * show the basic logic required to handle a Hosted Payment Page
 * transaction.
 *
 * This code is not intended for production use.
 */
package com.aciworldwide.demo;

import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class NotifyServlet extends HttpServlet {

   public void doPost(HttpServletRequest request, HttpServletResponse resp)
                    throws ServletException, IOException {

        /*
         * This is an example of how a Merchant might build a servlet to
         * receive and process Merchant Notification Messages
         */
         ServletOutputStream out;
         out = resp.getOutputStream();

         try {
              // get message details sent from Commerce Gateway
              long paymentID  = Long.parseLong(request.getParameter("paymentid") );
              String result   = request.getParameter("result");
              String postdate = request.getParameter("postdate");
              String details  = request.getParameter("udf1");
              long tranid     = Long.parseLong( request.getParameter("tranid") );
              String auth     = request.getParameter("auth");
              String trackid  = request.getParameter("trackid");

              // Note:
              // The Order and Orders objects are conceptual abstractions of the
              // Merchant's database.

              // read up the order information from the Merchant's database
              // in this example we are able to use the Payment ID. A Merchant
              // may not be able to use this particular field for that. However,
              // the Merchant does have control of the Track ID field, which is
              // sent in the Merchant Notification Message as well.
              Orders orders = new Orders();
              Order order = orders.fetch(paymentID);
              if ( order == null ) {
                   System.out.println("Could not find Order #" + paymentID);
                   String urlStr = "REDIRECT=http://" + request.getServerName() +
```

```
                        ":" + request.getServerPort() +
                    request.getContextPath() + "/error.jsp";
                out.print(urlStr);
                return;
            }

            // update the merchant database with the result of transaction
            order.setResultCode(result);
            order.setPostDate(postdate);
            order.setTranID(tranid);
            order.setAuthCode(auth);
            orders.update(order);
            System.out.println("Notified of Order #" + paymentID +
                                "completion.");

            // respond with the URL the consumer should be redirected to
            String url = "REDIRECT=http://" +
                         request.getServerName() + ":" +
                         request.getServerPort() + "/" +
                         request.getContextPath() +
                         "/result.jsp?paymentid=" + paymentID;

            out.print(url);
           out.flush();
          out.close();
          return;
        } catch (Exception e) {

       out.flush();
           out.close();
           return;
        }
    }
}
```

Result.jsp – The sample receipt page.



A sample skeleton of the source follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ page language="java" session="true"%>
<%@ page import="com.aciworldwide.demo.Order" %>
<%@ page import="com.aciworldwide.demo.Orders" %>
<%@ page import="java.text.DecimalFormat" %>
```

```
<HTML>
<HEAD>
<TITLE>Colors of Success</TITLE>
<META HTTP-EQUIV="PRAGMA" CONTENT="NO-CACHE">
<META name="GENERATOR" content="IBM WebSphere Studio">
</HEAD>

<BODY>
<%@include file="header.jsp" %>

<CENTER>

<%
// get Merchant Notification parameters
String payID = request.getParameter("paymentid");
if ( payID == null ) {
     response.sendRedirect(response.encodeRedirectURL("error.jsp"));
     return;
}


long paymentID  = Long.parseLong( payID );

// read up the order info
Orders orders = new Orders();
Order order = orders.fetch(paymentID);

if ( order == null ) {
     response.sendRedirect(response.encodeRedirectURL("error.jsp"));
     return;
}

if ( order.getResultCode().equals("CAPTURED") ||
     order.getResultCode().equals("APPROVED") ) { %>
     <CENTER>
          <FONT size="5" color="GREEN">
               Transaction completed successfully.<BR>
               Thank you for your order.
          </FONT>
     </CENTER>
<% } else { %>
          <FONT size="5" color="RED">
              An error occurred while processing your order.<BR>
               Please try again.
          </FONT>
          <P>
          <FONT color="BLUE"><A href="index.jsp">Click for Main Page</A></FONT>
<% } %>
     <P><P>
     <TABLE>
          <TR>
               <TD colspan="2" align="center">
                    <FONT size="4"><B>Transaction Details
                    <br>(from Merchant Notification Message)</B></FONT>
               </TD>
          </TR>
          <TR>
               <TD colspan="2" align="center">
                    <HR>
               </TD>
          </TR>
          <TR>
```

```
     <TD width="40%">Track ID</TD>
     <TD><%=order.getTrackID()%></TD>
</TR>
<TR>
     <TD>Transaction Date</TD>
     <TD><%=order.getTranDate()%></TD>
</TR>
<TR>
     <TD>Order Details</TD>
     <TD><%=order.getOrderDetails()%></TD>
</TR>
<TR>
     <TD>Amount</TD>
<%
String total = new
     DecimalFormat("#####.00").format(order.getAmount());
%>
     <TD><%=total%></TD>
</TR>
<TR>
     <TD>Post Date</TD>
     <TD><%=order.getPostDate()%></TD>
</TR>
<TR>
     <TD>Result Code</TD>
     <TD><%=order.getResultCode()%></TD>
</TR>

<TR>
     <TD>Payment ID</TD>
     <TD><%=order.getPaymentID()%></TD>
</TR>
<TR>
     <TD>Transaction ID</TD>
     <TD><%=order.getTranID()%></TD>
</TR>

<TR>
</TR>
<TR>
     <TD colspan="2" align="center">
          <FONT size="4"><B>Customer Shipping Details
                              <br>(from Merchant's website)</B></FONT>
     </TD>
</TR>
<TR>
     <TD colspan="2" align="center">
          <HR>
     </TD>
</TR>
<TR>
     <TD>Customer Name</TD>
     <TD><%=order.getName()%></TD>
</TR>
<TR>
     <TD>Address 1</TD>
     <TD><%=order.getAddr1()%></TD>
</TR>
<TR>
     <TD>Address 2</TD>
     <TD><%=order.getAddr2()%></TD>
</TR>
```

```
                <TR>
                     <TD>Address 3</TD>
                     <TD><%=order.getAddr3()%></TD>
                </TR>
                <TR>
                     <TD>City</TD>
                     <TD><%=order.getCity()%></TD>
                </TR>
                <TR>
                     <TD>State</TD>
                     <TD><%=order.getState()%></TD>
                </TR>
                <TR>
                     <TD>Country</TD>
                     <TD><%=order.getCountry()%></TD>
                </TR>
                <TR>
                     <TD>Postal Code</TD>
                     <TD><%=order.getPostalCode()%></TD>
                </TR>
           </TABLE>

</CENTER>

<%@ include file="footer.jsp"%>
</BODY>
</HTML>
```