

# Integrating Application Express with PayPal Payments Pro

*An Oracle White Paper  
September 2007*

# Integrating Application Express with PayPal Payments Pro

Introduction.....	3
How Application Express Can Integrate with PayPal.....	4
Prerequisites.....	4
PayPal Sandbox.....	5
Wallet for SSL Interaction.....	5
Create Tables.....	5
Create Standalone Procedure for Redirect.....	6
Create Package to Handle Communication with PayPal.....	7
Create Application to Accept Payment.....	13
Create an Application.....	13
Define Substitutions for PayPal Sandbox.....	13
Create an Application Item for PayPal TOKEN.....	14
Create Purchase Tickets Form.....	15
Add Javascript to the Form.....	17
Add Page Processing for Purchase Tickets Form.....	18
Create Confirmation Page.....	19
Create Purchase Confirmed Page.....	23
Create Pay By Credit Card Page.....	25
Create Purchase Confirmed Page for Pay By Credit Card.....	27
Testing The Ticketing Application.....	29
Conclusion.....	30

# Integrating Application Express with PayPal Payments Pro

## INTRODUCTION

Oracle Application Express – a feature of the Oracle Database – is a powerful and easy to use web application development platform. With Oracle Application Express, you can quickly develop and deploy applications in a matter of hours, often without writing a single line of code.

A common scenario for a Web application is to accept payment for goods or services. A popular payment acceptance company is PayPal. PayPal provides an API and process to allow Web sites to accept payments from its users. PayPal provides a standard payment acceptance offering which involves embedding an HTML form that posts and redirects to PayPal. PayPal also provides a Payment Pro offering which allows a Web site to accept both PayPal payments by a redirect method as well as credit card payments directly on the partnering Web site. The integration is achieved through a name value pair API in which URLs are constructed and posted to a service hosted by PayPal.

This whitepaper describes how to build an Application Express Application to integrate with PayPal's Website Payments Pro<sup>1</sup> offering. The paper will describe how to integrate with the PayPal Sandbox<sup>2</sup> which is the test environment provided by PayPal. Integrating with the live environment is simply a matter of changing the API URL, and your API credentials, to use a live PayPal account and service.

---

<sup>1</sup> [https://www.paypal.com/cgi-bin/webscr?cmd=\\_wp-pro-overview-outside](https://www.paypal.com/cgi-bin/webscr?cmd=_wp-pro-overview-outside)

<sup>2</sup> [https://www.paypal.com/IntegrationCenter/ic\\_sandbox.html](https://www.paypal.com/IntegrationCenter/ic_sandbox.html)

## **HOW APPLICATION EXPRESS CAN INTEGRATE WITH PAYPAL**

The PayPal Name Value Pair (NVP) API accepts HTTP posts and responds using NVP parameters. The Oracle Database includes a supplied PL/SQL package, UTL\_HTTP, which allows you to programmatically post information and get responses from URLs. Through the use of PL/SQL processes in an Application Express application, you can call a procedure that uses UTL\_HTTP to interact with the PayPal NVP.

With PL/SQL processes on a page, Application Express can interact with either the PayPal Direct Payment API or the PayPal Express Checkout API. Both are required as options when you use PayPal Website Payments Pro solution.

In the case of the Direct Payment API, you accept a credit card number, expiration date, and card holder information and use a PL/SQL process that uses UTL\_HTTP to post that information over HTTPS to PayPal. You receive a response that indicates if the transaction was successful.

Integrating with the Express Checkout is a little more involved and requires calling three distinct APIs. You first establish with PayPal that you want to start an express checkout transaction by calling SetExpressCheckout. PayPal responds with a token which you will need for subsequent API calls and a redirect. In the call to SetExpressCheckout you pass a URL as a parameter to tell PayPal where to redirect back to. You then redirect to PayPal and pass the token you received.

The consumer then chooses their payment option on the PayPal site and can optionally change their shipping address. Once they finalize the transaction at PayPal, they are redirected to the URL you provided in the call to SetExpressCheckout. Two parameters are passed with the URL you provide, token and payerid.

You create a PL/SQL procedure that has exactly two parameters, token and payerid. This procedure is the URL you pass in SetExpressCheckout to let PayPal know where to redirect to once the transaction is done. The procedure looks up the Application Express session for your application using the token.

Now that the consumer is back at your site, you will make a call to the second API, GetExpressCheckoutDetails. This retrieves information from PayPal about the consumer, such as their name, email, phone number, and shipping address. Your application can use that information to display a summary page and a Pay Now button.

Finally, when the consumer clicks the Pay Now button, you call the third API, DoExpressCheckoutPayment. PayPal responds with an acknowledgement and transaction details which your application can store in a local table.

## **PREREQUISITES**

This paper describes interacting with PayPal's NVP API in the sandbox. You will need PayPal Sandbox credentials. Since the PayPal NVP API requires the use of SSL through HTTPS, you will also need to configure a wallet on your Oracle database server.

## PayPal Sandbox

The PayPal Sandbox is an environment where developers can test and prototype the PayPal APIs. The purpose is to allow the developer to work out any deficiencies in their application prior to actually interacting with the live PayPal API. The Sandbox mirrors live interactions with PayPal as closely as possible. This paper describes interaction with the PayPal Sandbox.

Follow the instructions at the URL below to obtain a PayPal Sandbox account and create your test users:

[https://www.paypal.com/IntegrationCenter/ic\\_sandbox.html#resources](https://www.paypal.com/IntegrationCenter/ic_sandbox.html#resources)

Once you have your test users (a personal account type and a business account type), you will need to login to the PayPal sandbox using the business user you created. You will need to accept and complete the billing agreement before you can interact with the Direct Payment API.

## Wallet for SSL Interaction

In order for UTL\_HTTP to interact with an SSL enabled URL, HTTPS, a wallet needs to be created and accessible to the database server where Application Express is installed. A wallet is a password protected container used to store certificates needed by SSL.

To create a wallet you use the Oracle Wallet Manager. Start the wallet manager and follow the instructions to create a new wallet in the Oracle Database Advanced Security Administrator's Guide:

[http://download.oracle.com/docs/cd/B28359\\_01/network.111/b28530/asowalet.htm#BABGGBIG](http://download.oracle.com/docs/cd/B28359_01/network.111/b28530/asowalet.htm#BABGGBIG)

Be sure to note the file system path where the wallet is stored and the password for the wallet. You will need both pieces of information to configure Application Express to use the wallet you created above. Follow the instructions in the Application Express User's Guide to configure Application Express to use a wallet:

[http://download.oracle.com/docs/cd/B28359\\_01/appdev.111/b32258/adm\\_wrkspc.htm#BABHEHAG](http://download.oracle.com/docs/cd/B28359_01/appdev.111/b32258/adm_wrkspc.htm#BABHEHAG)

## CREATE TABLES

You will need two tables for an application to interact with the PayPal APIs. The first table will contain a token to session mapping so that when PayPal redirects back to your standalone procedure, that procedure will know what Application Express application, page, and session to redirect to. The second table is used for recording the results of transactions with PayPal including the unique transaction id. The SQL for the create table statements is below.

```

create table paypal_session_map(
    session_id      number not null,
    app_id          number not null,
    page_id         number not null,
    payer_id_item   varchar2(4000),
    session_token   varchar2(255)
)
/

create table paypal_transactions(
    session_id      number not null,
    session_token   varchar2(255),
    transaction_id  varchar2(30),
    order_time      varchar2(30),
    amount          number,
    fee_amount      number,
    settle_ammount number,
    payment_status  varchar2(30),
    pending_reason  varchar2(30),
    reason_code     varchar2(30)
)
/

```

Create both of these tables using the Application Express SQL Workshop.

### CREATE STANDALONE PROCEDURE FOR REDIRECT

A standalone procedure is necessary for the redirect from PayPal, because PayPal appends two parameters to the query string of whatever URL you provide to SetExpressCheckout. For that reason you need to create a standalone procedure that matches the signature of the two parameters passed by PayPal, token and payerid. The SQL for the stand alone procedure is below:

```

create or replace procedure paypal_accept
    (token in varchar2,
     PayerId in varchar2)
as
begin
    for c1 in (select session_id, app_id,
                    page_id, payer_id_item
                from paypal_session_map
                where session_token = token ) loop
        owa_util.redirect_url('f?p='||c1.app_id||':'||c1.page_id||'
        :'||c1.session_id||'::::'||c1.payer_id_item||':'||PayerId);
        exit;
    end loop;
end paypal_accept;
/

```

Then grant execute on this procedure to public:

```

grant execute on paypal_accept to public
/

```

Execute the code above using the Command Processor in the Application Express SQL Workshop.

## CREATE PACKAGE TO HANDLE COMMUNICATION WITH PAYPAL

Handling the communication with the PayPal NVP API using UTL\_HTTP is the main thrust of this paper. The package has one procedure that handles all the UTL\_HTTP communication with PayPal. Then, there is a procedure for each API interaction with PayPal and also a function that parses out the response. Below is the SQL for the package specification and body.

```
create or replace package paypal_api
as

function do_post(
    p_api_url           in varchar2,
    p_api_username      in varchar2,
    p_api_password      in varchar2,
    p_signature          in varchar2,
    p_wallet            in varchar2,
    p_wallet_pwd        in varchar2,
    p_method            in varchar2,
    p_parm01            in varchar2,
    p_parm02            in varchar2 default null,
    p_parm03            in varchar2 default null,
    p_parm04            in varchar2 default null,
    p_parm05            in varchar2 default null,
    p_parm06            in varchar2 default null,
    p_parm07            in varchar2 default null,
    p_parm08            in varchar2 default null,
    p_parm09            in varchar2 default null,
    p_parm10            in varchar2 default null )
    return varchar2;

function get_parameter(
    p_response          in varchar2,
    p_parameter         in varchar2 )
    return varchar2;

procedure set_express_checkout(
    p_api_url           in varchar2,
    p_api_username      in varchar2,
    p_api_password      in varchar2,
    p_signature          in varchar2,
    p_wallet            in varchar2,
    p_wallet_pwd        in varchar2,
    p_session_id        in varchar2,
    p_return_page        in varchar2,
    p_payerid_item      in varchar2,
    p_redirect_url      in varchar2,
    p_return_url         in varchar2,
    p_cancel_url        in varchar2,
    p_amount             in varchar2,
    p_description        in varchar2,
    p_token_item        out varchar2 );

procedure get_express_checkout_details(
    p_api_url           in varchar2,
    p_api_username      in varchar2,
    p_api_password      in varchar2,
    p_signature          in varchar2,
    p_wallet            in varchar2,
    p_wallet_pwd        in varchar2,
    p_token             in varchar2,
    p_email_item        out varchar2,
    p_fname_item        out varchar2,
    p_mname_item        out varchar2,
    p_lname_item        out varchar2,
```

```

        p_shiptoname_item      out varchar2,
        p_shiptostreet_item    out varchar2,
        p_shiptostreet2_item   out varchar2,
        p_shiptocity_item      out varchar2,
        p_shiptocc_item        out varchar2,
        p_shiptozip_item       out varchar2,
        p_phonenum_item        out varchar2 );

procedure do_express_checkout_payment(
    p_api_url                in varchar2,
    p_api_username           in varchar2,
    p_api_password           in varchar2,
    p_signature              in varchar2,
    p_wallet                 in varchar2,
    p_wallet_pwd             in varchar2,
    p_session_id            in varchar2,
    p_token                  in varchar2,
    p_payerid               in varchar2,
    p_amount                 in varchar2,
    p_description            in varchar2 );

procedure do_direct_payment(
    p_api_url                in varchar2,
    p_api_username           in varchar2,
    p_api_password           in varchar2,
    p_signature              in varchar2,
    p_wallet                 in varchar2,
    p_wallet_pwd             in varchar2,
    p_session_id            in varchar2,
    p_ip_address            in varchar2,
    p_amount                 in varchar2,
    p_creditcardtype        in varchar2,
    p_account                in varchar2,
    p_expire_date           in varchar2,
    p_first_name            in varchar2,
    p_last_name             in varchar2,
    p_description            in varchar2,
    p_tran_id_item          out varchar2 );

end paypal_api;
/

set define off

create or replace package body paypal_api
as

function do_post(
    p_api_url                in varchar2,
    p_api_username           in varchar2,
    p_api_password           in varchar2,
    p_signature              in varchar2,
    p_wallet                 in varchar2,
    p_wallet_pwd             in varchar2,
    p_method                 in varchar2,
    p_parm01                 in varchar2,
    p_parm02                 in varchar2 default null,
    p_parm03                 in varchar2 default null,
    p_parm04                 in varchar2 default null,
    p_parm05                 in varchar2 default null,
    p_parm06                 in varchar2 default null,
    p_parm07                 in varchar2 default null,
    p_parm08                 in varchar2 default null,
    p_parm09                 in varchar2 default null,
    p_parm10                 in varchar2 default null )
return varchar2

is
    l_http_req               utl_http.req;

```



```

        l_http_resp      utl_http.resp;
        l_response       varchar2(4000);
        l_post           varchar2(4000);
begin
    l_post :=
'USER='||p_api_username||'&PWD='||p_api_password||'&SIGNATURE='||p
_signature||
    '&'||p_parm01;

    if p_parm02 is not null then
        l_post := l_post||'&'||p_parm02;
    end if;
    if p_parm03 is not null then
        l_post := l_post||'&'||p_parm03;
    end if;
    if p_parm04 is not null then
        l_post := l_post||'&'||p_parm04;
    end if;
    if p_parm05 is not null then
        l_post := l_post||'&'||p_parm05;
    end if;
    if p_parm06 is not null then
        l_post := l_post||'&'||p_parm06;
    end if;
    if p_parm07 is not null then
        l_post := l_post||'&'||p_parm07;
    end if;
    if p_parm08 is not null then
        l_post := l_post||'&'||p_parm08;
    end if;
    if p_parm09 is not null then
        l_post := l_post||'&'||p_parm09;
    end if;
    if p_parm10 is not null then
        l_post := l_post||'&'||p_parm10;
    end if;

    l_post := l_post||'&VERSION=2.6&METHOD='||p_method;

    utl_http.set_proxy(apex_application.g_proxy_server, NULL);
    utl_http.set_persistent_conn_support(TRUE);
    utl_http.set_transfer_timeout(300);
    utl_http.set_wallet(p_wallet, p_wallet_pwd);
    l_http_req := utl_http.begin_request(p_api_url, 'POST');
    utl_http.set_header(l_http_req, 'Proxy-Connection', 'Keep-
Alive');
    utl_http.set_header(l_http_req, 'Content-Type',
'application/x-www-form-urlencoded; charset=utf-8');
    utl_http.set_header(l_http_req, 'Content-Length',
length(l_post));
    utl_http.write_text(l_http_req, l_post);
    l_http_resp := utl_http.get_response(l_http_req);
    utl_http.read_text(l_http_resp, l_response);

    return utl_url.unescape(l_response);

end do_post;

function get_parameter(
    p_response          in varchar2,
    p_parameter         in varchar2 )
return varchar2
is
    l_start            number;
    l_end              number;

begin

```

```

    if instr(p_response,p_parameter||'=') = 0 then
        return null;
    end if;

    l_start := instr(p_response,p_parameter||'=') +
length(p_parameter) + 1;
    l_end := instr(p_response,'&',l_start);

    if l_end != 0 then
        return substr(p_response,l_start,l_end - l_start);
    else
        return substr(p_response,l_start);
    end if;

end get_parameter;

procedure set_express_checkout(
    p_api_url          in varchar2,
    p_api_username     in varchar2,
    p_api_password     in varchar2,
    p_signature        in varchar2,
    p_wallet           in varchar2,
    p_wallet_pwd       in varchar2,
    p_session_id       in varchar2,
    p_return_page      in varchar2,
    p_payerid_item     in varchar2,
    p_redirect_url     in varchar2,
    p_return_url       in varchar2,
    p_cancel_url       in varchar2,
    p_amount           in varchar2,
    p_description       in varchar2,
    p_token_item       out varchar2 )
is
    l_response varchar2(4000);
    l_token    varchar2(30);
begin
    l_response := do_post(
        p_api_url          => p_api_url,
        p_api_username     => p_api_username,
        p_api_password     => p_api_password,
        p_signature        => p_signature,
        p_wallet           => p_wallet,
        p_wallet_pwd       => p_wallet_pwd,
        p_method           => 'SetExpressCheckout',
        p_parm01           => 'RETURNURL='||p_return_url,
        p_parm02           => 'CANCELURL='||p_cancel_url,
        p_parm03           => 'AMT='||p_amount,
        p_parm04           => 'DESC='||p_description );

    if get_parameter(l_response,'ACK') != 'Success' then
        raise_application_error(-20001,'Error: '||l_response);
    end if;

    l_token := get_parameter(l_response,'TOKEN');

    p_token_item := l_token;

    delete from paypal_session_map where session_id =
p_session_id;

    insert into paypal_session_map values (p_session_id,
apex_application.g_flow_id, p_return_page, p_payerid_item,
l_token);

    apex_application.g_unrecoverable_error := true;
    owa_util.redirect_url(p_redirect_url||l_token);

end set_express_checkout;

```

```

procedure get_express_checkout_details(
    p_api_url          in varchar2,
    p_api_username     in varchar2,
    p_api_password     in varchar2,
    p_signature        in varchar2,
    p_wallet           in varchar2,
    p_wallet_pwd       in varchar2,
    p_token            in varchar2,
    p_email_item       out varchar2,
    p_fname_item       out varchar2,
    p_mname_item       out varchar2,
    p_lname_item       out varchar2,
    p_shiptoname_item  out varchar2,
    p_shiptostreet_item out varchar2,
    p_shiptostreet2_item out varchar2,
    p_shiptocity_item  out varchar2,
    p_shiptocc_item    out varchar2,
    p_shiptozip_item   out varchar2,
    p_phonenum_item    out varchar2 )
is
    l_response varchar2(4000);
begin

    l_response := do_post(
        p_api_url          => p_api_url,
        p_api_username     => p_api_username,
        p_api_password     => p_api_password,
        p_signature        => p_signature,
        p_wallet           => p_wallet,
        p_wallet_pwd       => p_wallet_pwd,
        p_method           => 'GetExpressCheckoutDetails',
        p_parm01           => 'TOKEN='||p_token );

    if get_parameter(l_response,'ACK') != 'Success' then
        raise_application_error(-20001,'Error: '||l_response);
    end if;

    p_email_item := get_parameter(l_response,'EMAIL');
    p_fname_item := get_parameter(l_response,'FIRSTNAME');
    p_mname_item := get_parameter(l_response,'MIDDLENAME');
    p_lname_item := get_parameter(l_response,'LASTNAME');
    p_shiptoname_item := get_parameter(l_response,'SHIPTONAME');
    p_shiptostreet_item :=
get_parameter(l_response,'SHIPTOSTREET');
    p_shiptostreet2_item :=
get_parameter(l_response,'SHIPTOSTREET2');
    p_shiptocity_item := get_parameter(l_response,'SHIPTOCITY');
    p_shiptocc_item :=
get_parameter(l_response,'SHIPTOCOUNTRYCODE');
    p_shiptozip_item := get_parameter(l_response,'SHIPTOZIP');
    p_phonenum_item := get_parameter(l_response,'PHONENUM');

end get_express_checkout_details;

procedure do_express_checkout_payment(
    p_api_url          in varchar2,
    p_api_username     in varchar2,
    p_api_password     in varchar2,
    p_signature        in varchar2,
    p_wallet           in varchar2,
    p_wallet_pwd       in varchar2,
    p_session_id       in varchar2,
    p_token            in varchar2,
    p_payerid          in varchar2,
    p_amount           in varchar2,
    p_description      in varchar2 )
is
    l_response varchar2(4000);

```

```

begin

    l_response := do_post(
        p_api_url      => p_api_url,
        p_api_username => p_api_username,
        p_api_password => p_api_password,
        p_signature    => p_signature,
        p_wallet       => p_wallet,
        p_wallet_pwd   => p_wallet_pwd,
        p_method       => 'DoExpressCheckoutPayment',
        p_parm01       => 'TOKEN=' || p_token,
        p_parm02       => 'PAYMENTACTION=Sale',
        p_parm03       => 'AMT=' || p_amount,
        p_parm04       => 'PAYERID=' || p_payerid,
        p_parm05       => 'DESC=' || p_description );

    if get_parameter(l_response,'ACK') != 'Success' then
        raise_application_error(-20001,'Error: ' || l_response);
    end if;

    insert into paypal_transactions values (p_session_id, p_token,
    get_parameter(l_response,'TRANSACTIONID'),
        get_parameter(l_response,'ORDERTIME'),
    get_parameter(l_response,'AMT'),get_parameter(l_response,'FEEAMT')
    ,
        get_parameter(l_response,'SETTLEAMT'),
    get_parameter(l_response,'PAYMENTSTATUS'),
    get_parameter(l_response,'PENDINGREASON'),
        get_parameter(l_response,'REASONCODE'));

end do_express_checkout_payment;

procedure do_direct_payment(
    p_api_url          in varchar2,
    p_api_username     in varchar2,
    p_api_password     in varchar2,
    p_signature        in varchar2,
    p_wallet           in varchar2,
    p_wallet_pwd       in varchar2,
    p_session_id       in varchar2,
    p_ip_address       in varchar2,
    p_amount           in varchar2,
    p_creditcardtype   in varchar2,
    p_account          in varchar2,
    p_expire_date      in varchar2,
    p_first_name       in varchar2,
    p_last_name        in varchar2,
    p_description      in varchar2,
    p_tran_id_item     out varchar2 )
is
    l_response         varchar2(4000);
    l_transaction_id   varchar2(30);
begin

    l_response := do_post(
        p_api_url      => p_api_url,
        p_api_username => p_api_username,
        p_api_password => p_api_password,
        p_signature    => p_signature,
        p_wallet       => p_wallet,
        p_wallet_pwd   => p_wallet_pwd,
        p_method       => 'DoDirectPayment',
        p_parm01       => 'PAYMENTACTION=Sale',
        p_parm02       => 'IPADDRESS=' || p_ip_address,
        p_parm03       => 'AMT=' || p_amount,
        p_parm04       => 'CREDITCARDTYPE=' || p_creditcardtype,
        p_parm05       => 'ACCT=' || p_account,
        p_parm06       => 'EXPDATE=' || p_expire_date,
        p_parm07       => 'FIRSTNAME=' || p_first_name,

```

```

        p_parm08          => 'LASTNAME=' || p_last_name,
        p_parm09          => 'DESC=' || p_description );

    if get_parameter(l_response,'ACK') != 'Success' then
        raise_application_error(-20001,'Error: ' || l_response);
    end if;

    l_transaction_id := get_parameter(l_response,'TRANSACTIONID');

    insert into paypal_transactions values (p_session_id, null,
        l_transaction_id,
        null, get_parameter(l_response,'AMT'), null, null, null,
        null, null);

    p_tran_id_item := l_transaction_id;
end do_direct_payment;

end paypal_api;
/

```

Use the Command Processor in the Application Express SQL Workshop to create the paypal\_api package and corresponding package body.

## CREATE APPLICATION TO ACCEPT PAYMENT

Now that the communication infrastructure is in place, you can begin building an Application Express application to integrate with PayPal Website Payments Pro. The scenario for this whitepaper is an application to accept payment for tickets to an Application Express training event.

### Create an Application

The first step is to create an application. To create an Application Express application:

1. Login to your Application Express instance.
 

Please note that the application described in this paper cannot be completed on apex.oracle.com because it does not support external network call-outs. You will need to use your own instance to be able to execute the resulting application.
2. Select **Application Builder**
3. Click **Create >**
4. Supply information as required by the Create Application wizard
5. Add one blank page on the Pages step of the wizard
6. Once you have created the application, if your environment requires a proxy server to reach pages on the Internet, supply the proxy server in the Proxy Server field of the Application Definition (Shared Components > Definition - under Application)

### Define Substitutions for PayPal Sandbox

You will use substitutions in your application to define the PayPal URLs and credentials. By using substitutions, you can easily change your application to work with the live PayPal services without changing any other code.

To specify substitutions for the PayPal sandbox in your Application:

1. Click **Shared Components** from the Application Builder home page
2. Click **Definition** under Application on the Shared Components page
3. Enter the following Substitution string and value pairs in the Substitutions section:

```
WALLET_PATH      file:/path/to/wallet
WALLET_PWD       password
API_URL          https://api-3t.sandbox.paypal.com/nvp
API_USERNAME     your_api_username
API_PASSWORD     your_api_password
API_SIGNATURE    your_api_signature
CHECKOUT_URL     https://www.sandbox.paypal.com/cgi-
bin/webscr?cmd=_express-checkout&token=
RETURN_URL       http://host:port/pls/apex/schema.paypal_accept
CANCEL_URL       http://host:port/pls/apex
```

Replace the substitution values with values that match your environment and API credentials. Your API credentials can be obtained from the API Credentials tab of PayPal Developer Central (<https://developer.paypal.com>).

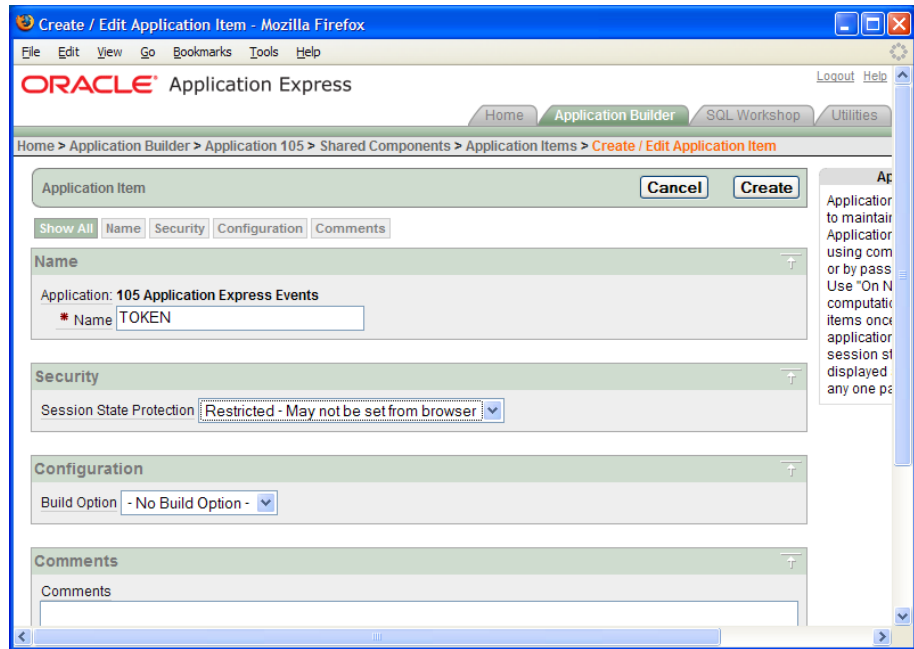
4. Click **Apply Changes**

### **Create an Application Item for PayPal TOKEN**

Interaction with the PayPal Express Checkout feature involves getting a token and passing it with subsequent requests. The token uniquely identifies a transaction. You store this token in an application item.

To create an application item for the PayPal TOKEN:



1. Click **Shared Components** from the Application Builder home page
2. Click **Application Items** under Logic
3. Click **Create >**
4. Enter **TOKEN** in the name field
5. Choose **Restricted** from the Session State Protection list and click **Create**



## Create Purchase Tickets Form

Create a form on page 1 to allow prospective attendees to choose the number of tickets they want to purchase. You will edit the blank page that was created when you created the application.

To create a form to allow attendees to choose the number of tickets they need:

1. Go to the page definition of page 1
2. Click the edit icon (  ) in the Page section under Page Rendering
3. Enter **Purchase Tickets** in the Name and Title fields
4. Click **Apply Changes**
5. Click the create icon (  ) in the Regions section
6. Choose **HTML** from the list of region types and click **Next >**
7. Choose **HTML** from the HTML container type list and click **Next >**
8. Enter **Purchase Tickets** in the Title field
9. Choose **Form Region** from the Region Template list and click **Next >**
10. Click **Create Region**
11. Click the create icon in the Buttons region
12. Choose the **Purchase Tickets** region and click **Next >**
13. Accept the button position default and click **Next >**
14. Enter **CREATE** in the Button Name field
15. Enter **Check Out** in the Label field and click **Next >**

16. Accept default Button Template and click **Next >**
17. Accept default Display Properties and click **Next >**
18. Leave the Branch to Page field blank and click **Create Button**
19. Click the create icon in the Items section
20. Choose **Display Only** as the item type and click **Next >**
21. Choose **Display as Text (saves state)** from the list and click **Next >**
22. Enter **P1\_DESCRIPTION** in the Item Name field
23. Select **Purchase Tickets** from the Region list and click **Next >**
24. Enter **Event:** in the Label field and click **Next >**
25. Enter **Application Express Advanced Workshop** in the Item Source Value text area and click **Create Item**
26. Click the create icon in the Items section
27. Choose **Select List** from the item type list and click **Next >**
28. Choose **Select List** from the Select List Control Type list and click **Next >**
29. Enter **P1\_QUANTITY** in the Item Name field
30. Choose **Purchase Tickets** from the Region List and click **Next >**
31. Choose **No** from the Display Null Option List
32. Enter **STATIC:1,2,3,4,5,6,7,8** in the List of values definition text area and click **Next >**
33. Enter **Quantity** in the Label field and click **Next >**
34. Click **Create Item**
35. Click the create icon in the Items section
36. Choose **Text** from the item type list and click **Next >**
37. Choose **Text Field** from the Text Control Display Type and click **Next >**
38. Enter **P1\_AMOUNT** in the Item Name field
39. Select **Purchase Tickets** from the Region list and click **Next >**
40. Enter **Amount** in the Label field click **Next >**
41. Enter **40** in the Default text area and click **Create Item**
42. Click the create icon in the items section
43. Choose **Radio** from the Item Type list and click **Next >**
44. Choose **Radio group** from the Radio Group Control type list and click **Next >**
45. Enter **P1\_PAYMENT\_OPTION** in the Item Name field
46. Select **Purchase Tickets** from the Region list and click **Next >**



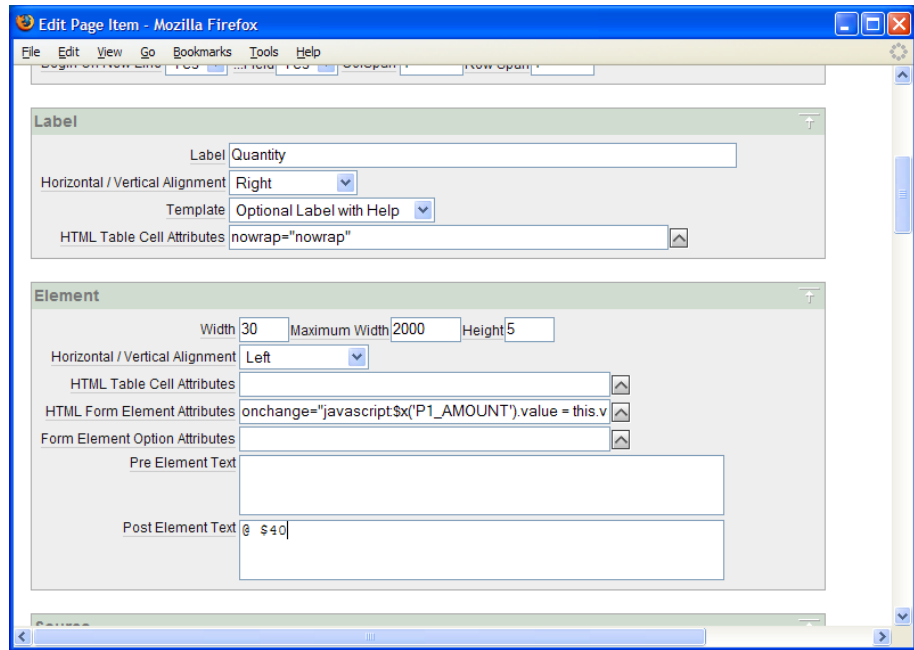
47. Choose **No** from the Display Null Option list
48. Enter **STATIC2:Credit Card;CC,PayPal;PP** in the List of Values Query text area and click **Next >**
49. Enter **Payment Option** in the Label field and click **Next >**
50. Enter **PP** in the Default text area and click **Create Item**

### Add Javascript to the Form

Next you will add some Javascript to the form to populate the amount field based on the number of tickets chosen and to make the amount field read only.

To add Javascript to the P1\_QUANTITY and P1\_AMOUNT fields:

1. Click **P1\_QUANTITY** in the Items area
2. Enter **onchange="javascript:\$x('P1\_AMOUNT').value = this.value\*40;"** in the HTML Form Element Attributes field in the Element area
3. Enter **@ \$40** in the Post Element Text text area
4. Click **Apply Changes**
5. Click **P1\_AMOUNT** in the Items area
6. Enter **onFocus="this.blur()"** in the HTML Form Element Attributes field in the Element area
7. Click **Apply Changes**



### Add Page Processing for Purchase Tickets Form

Depending on the Payment option chosen, when a user clicks Check Out, the application should either call `paypal_api.set_express_checkout`, or redirect them to another page where they will enter their credit card information. The Purchase form page needs a process to call `paypal_api.set_express_checkout` and a branch to another page when the user selects the credit card option.

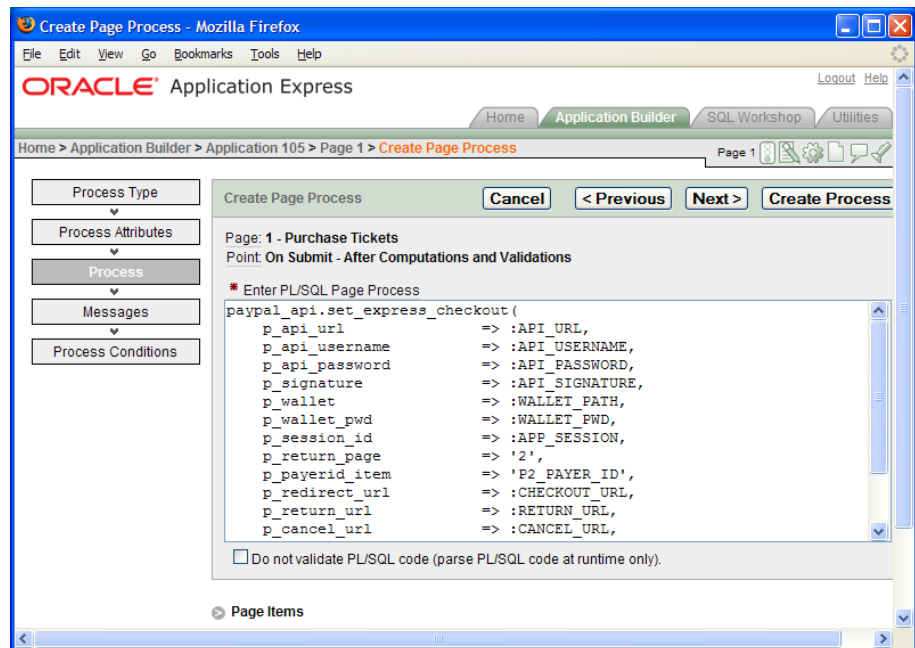
To create a process and a branch for the Purchase Tickets page:

1. Click the create icon in the Processes area under Page Processing
2. Choose **PL/SQL** from the process category list and click **Next >**
3. Enter **SetExpressCheckout** in the Name field and click **Next >**
4. Enter the following code in the PL/SQL Page Process text area:

```
paypal_api.set_express_checkout(
    p_api_url           => :API_URL,
    p_api_username      => :API_USERNAME,
    p_api_password      => :API_PASSWORD,
    p_signature         => :API_SIGNATURE,
    p_wallet            => :WALLET_PATH,
    p_wallet_pwd        => :WALLET_PWD,
    p_session_id        => :APP_SESSION,
    p_return_page       => '2',
    p_payerid_item      => 'P2_PAYER_ID',
    p_redirect_url      => :CHECKOUT_URL,
    p_return_url        => :RETURN_URL,
    p_cancel_url        => :CANCEL_URL,
    p_amount            => :P1_AMOUNT,
    p_description       => :P1_DESCRIPTION,
    p_token_item        => :TOKEN );
```

5. Click **Next >**
6. Leave Success Message and Failure Message text areas blank and click **Next >**

7. Choose **PL/SQL** from the condition type list
8. Enter **:REQUEST = 'CREATE' and :P1\_PAYMENT\_OPTION = 'PP'** in the Expression 1 text area and click **Create Process**
9. Click the create icon in the Branches area under Page Processing
10. Accept the defaults and click **Next >**
11. Enter **4** in the Page field and the Clear Cache field and click **Next >**
12. Choose **PL/SQL** from the condition type list
13. Enter **:REQUEST = 'CREATE' and :P1\_PAYMENT\_OPTION = 'CC'**
14. Click **Create Branch**



## Create Confirmation Page

When a prospective attendee chooses to pay for their tickets with PayPal on your site the `paypal_api.set_express_checkout` procedure will be called. This procedure will call the PayPal SetExpressCheckout API and retrieve a token. The token will be inserted into the `paypal_session_map` table so the token can be associated with the Application Express session. The attendee will then be redirected by the `paypal_api.set_express_checkout` procedure to PayPal to complete their purchase. You supply a return URL when a call is made to the SetExpressCheckout API. The return URL is to the standalone procedure `paypal_accept`, which accepts the token and payerid parameters.

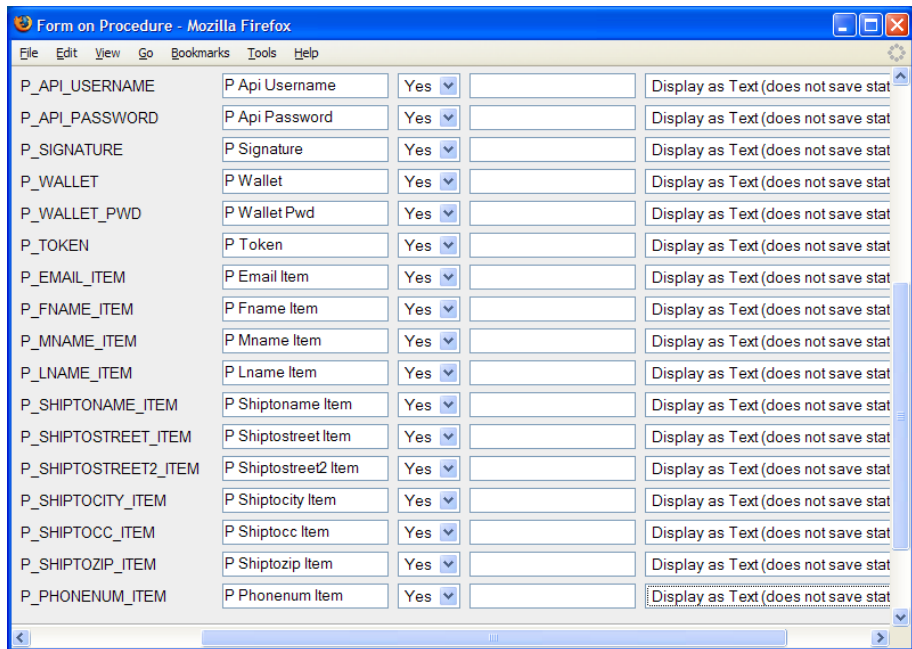
The `paypal_accept` procedure takes the token ID and looks up the session information in the `paypal_session_map` table. It uses that information to redirect back to the Application Express Events ticketing application in session context.

The page to redirect to is a confirmation page which displays the prospective attendee their billing information. This information is obtained by calling the PayPal GetExpressCheckoutDetails API. You need to create a page with display-only items that calls paypal\_api.get\_express\_checkout\_details to populate those items, and then finally calls paypal\_api.do\_express\_checkout\_payment which invokes the corresponding PayPal API to complete the transaction.

You can use the Form on Procedure wizard as a shortcut to create page 2 of the ticketing application. You will run the wizard and then customize the result.


To create the confirmation page:

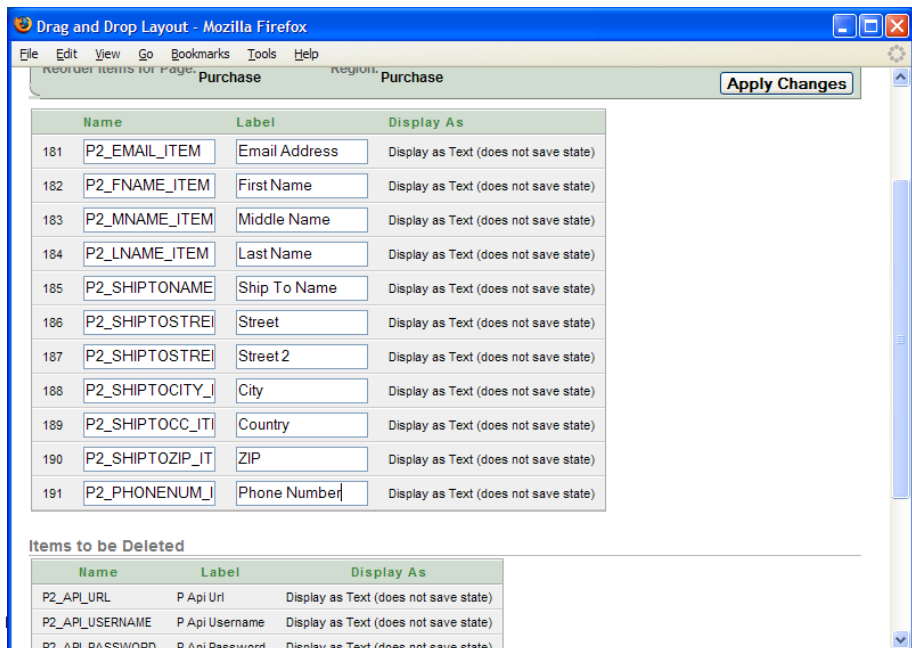
1. Click **Create Page >** on the Application Builder home page
2. Choose **Form** from the page type list and click **Next >**
3. Choose **Form on Procedure** and click **Next >**
4. Choose the owner of the paypal\_api package and click **Next >**
5. Choose **paypal\_api.get\_express\_checkout\_details** from the pop-up list and click **Next >**
6. Enter **2** in the Page Number field
7. Enter **Confirm Purchase** in the Page Name and Region Name fields
8. Enter **Purchase** in the Submit Button Label field
9. Choose **Breadcrumb** from the Breadcrumb list
10. Click **Purchase Tickets** to select the Parent Entry for the breadcrumb and click **Next >**
11. Choose **Use an existing tab set and reuse an existing tab** and click **Next >**
12. Choose **T\_PURCHASE\_TICKETS** (this may be T\_PAGE\_1) from the Use Tab list and click **Next >**
13. Leave Invoking Page and Button Label blank and click **Next >**
14. Enter **3** in the Branch here on Submit field
15. Enter **1** in the Branch here on Cancel field and click **Next >**
16. Change the Display Type of all items to **Display as Text** and click **Next >**
17. Click **Finish**



You can use the new drag and drop form layout feature to quickly edit the multiple items created by the form on procedure wizard. You will reference the application substitutions created in the beginning of this paper for the first 6 parameters, and the TOKEN application item for the seventh. You will delete all those items and then change the labels of the rest of the items.

To edit/delete items using drag and drop form layout:

1. Click the Drag and drop icon (  ) in the Items area of the page definition of page 2
2. Drag the first seven items, **P2\_API\_URL** to **P2\_TOKEN** to the trash can and click **Next >**
3. Change the Label for each item to be a readable label, e.g., Email Address
4. Click **Apply Changes**



The second parameter needed when PayPal redirects back to the Application Express ticketing application is PayerID. You need to create an item for PayerID on this page because it will be used in a subsequent call to the PayPal API.

To create an item for PayerID:

1. Click the create icon in the Items area on the page definition of page 2
2. Select **Hidden** from the Item Type list and click **Next >**
3. Enter **P2\_PAYER\_ID** in the Item Name field
4. Select **Confirm Purchase** (with the higher sequence number) from the Region list and click **Next >**
5. Click **Create Item**

Now you need to edit the process that was created by the Form on Procedure wizard. You will change the process point and the process itself to reference the application substitutions and item created earlier.

To edit the process created by the Form on Procedure Wizard:

1. Click **Run Stored Procedure** in the Processes area under Page Processing on the page definition of page 2
2. Enter **GetExpressCheckoutDetails** in the name field
3. Choose **On Load – After Header** from the Process Point List
4. Edit the process to use your application substitutions and item as follows:

```
#OWNER#.PAYPAL_API.GET_EXPRESS_CHECKOUT_DETAILS(  
P_API_URL => :API_URL,  
P_API_USERNAME => :API_USERNAME,  
P_API_PASSWORD => :API_PASSWORD,  
P_SIGNATURE => :API_SIGNATURE,  
P_WALLET => :WALLET_PATH,  
P_WALLET_PWD => :WALLET_PWD,  
P_TOKEN => :TOKEN,  
P_EMAIL_ITEM => :P2_EMAIL_ITEM,  
P_FNAME_ITEM => :P2_FNAME_ITEM,  
P_MNAME_ITEM => :P2_MNAME_ITEM,  
P_LNAME_ITEM => :P2_LNAME_ITEM,  
P_SHIPTONAME_ITEM => :P2_SHIPTONAME_ITEM,  
P_SHIPTOSTREET_ITEM => :P2_SHIPTOSTREET_ITEM,  
P_SHIPTOSTREET2_ITEM => :P2_SHIPTOSTREET2_ITEM,  
P_SHIPTOCITY_ITEM => :P2_SHIPTOCITY_ITEM,  
P_SHIPTOCC_ITEM => :P2_SHIPTOCC_ITEM,  
P_SHIPTOZIP_ITEM => :P2_SHIPTOZIP_ITEM,  
P_PHONENUM_ITEM => :P2_PHONENUM_ITEM);
```

5. Click **Apply Changes**

The process point of the above process was changed to call the PayPal API during page rendering to get the details of the transaction so it could be displayed on the confirmation page. The next step is to call the PayPal API

DoExpressCheckoutPayment which finalizes payment for the transaction. You create a process on the page that calls paypal\_api.do\_express\_checkout\_payment when the Purchase button is pressed.

To create a process that calls paypal\_api.do\_express\_checkout\_payment:

1. Click the create icon in the Processes area under Page Processing on the page definition of page 2
2. Choose **PL/SQL** from the process category list and click **Next >**
3. Enter **DoExpressCheckoutPayment** in the Name field and click **Next >**
4. Enter the following in the PL/SQL Page Process text area:

```
paypal_api.do_express_checkout_payment (
  p_api_url           => :API_URL,
  p_api_username      => :API_USERNAME,
  p_api_password      => :API_PASSWORD,
  p_signature         => :API_SIGNATURE,
  p_wallet            => :WALLET_PATH,
  p_wallet_pwd        => :WALLET_PWD,
  p_session_id        => :APP_SESSION,
  p_token             => :TOKEN,
  p_payerid           => :P2_PAYER_ID,
  p_amount            => :P1_AMOUNT,
  p_description        => :P1_DESCRIPTION );
```

5. Click **Next >**
6. Leave the message text areas blank and click **Next >**
7. Choose **SUBMIT** from the When Button Pressed list
8. Click **Create Process**

The `paypal_api.do_express_checkout_payment` procedure calls the PayPal API to complete the transaction and then stores the confirmation information in the `paypal_transactions` table.

### Create Purchase Confirmed Page

The final step of the PayPal Express Checkout process is to create a confirmation page with a report on the `paypal_transactions` table.

To create the Purchase Confirmed Page:

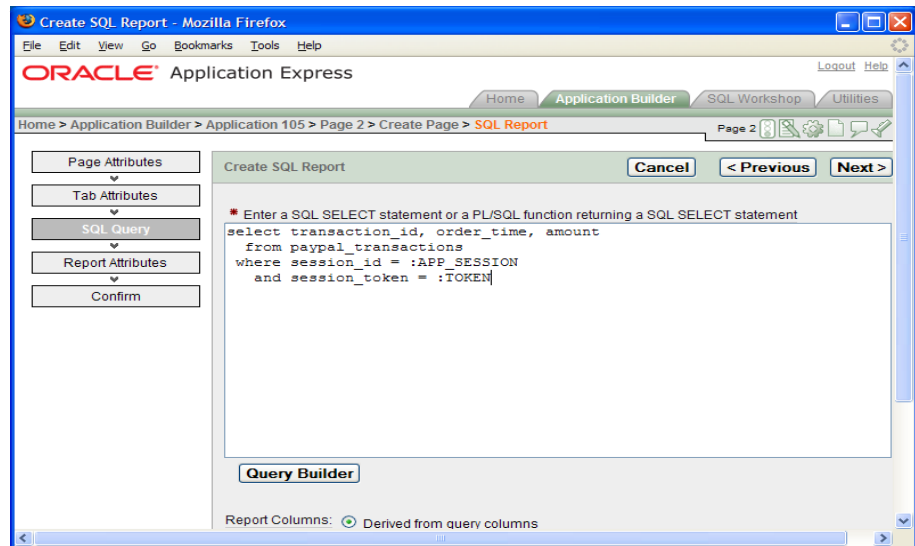
1. Click **Create Page >** from the Application Builder home page
2. Choose **Report** from the page type list and click **Next >**
3. Choose **SQL Report** and click **Next >**
4. Enter **Purchase Confirmed** in the Page Name field
5. Choose **Breadcrumb** from the Breadcrumb list
6. Enter **3** in the Page Number field
7. Click **Purchase Tickets** to select the Parent Entry for the breadcrumb and click **Next >**
8. Choose **Use an existing tab set and reuse an existing tab within that tab set** and click **Next >**
9. Choose **T\_PURCHASE\_TICKETS** (may also be `T_PAGE_1`) from the Use Tab list and click **Next >**
10. Enter the following in the SQL SELECT statement text area:

```

select transaction_id, order_time, amount
  from paypal_transactions
 where session_id = :APP_SESSION
    and session_token = :TOKEN

```

11. Click **Next >**
12. Choose **Value Attribute Pairs** from the Report Template list
13. Enter **Purchase Confirmed** in the Region Name field and click **Next >**
14. Click **Finish**



You should create a button on this page to allow for navigation back to the Purchase Tickets page.

To create a button for navigation back to the Purchase Tickets page:

1. Click the create icon in the Buttons area on the page definition of page 3
2. Choose **Purchase Confirmed** (with the higher sequence number) from the region list and **click Next >**
3. Leave the default button position and click **Next >**
4. Enter **PURCHASE** in the Button Name field
5. Enter **Purchase More Tickets** in the Label field
6. Select **Redirect to URL without submitting page** from the Action list and click **Next >**
7. Leave the button template defaulted and click **Next >**
8. Leave the button position defaulted and click **Next >**
9. Enter **1** in the Page field
10. Enter **1** in the Clear Cache field
11. Click **Create Button**



## Create Pay By Credit Card Page


PayPal's Website Payments Pro requires that you offer customers the choice of using the PayPal Express Checkout or to pay with a credit card directly on your site. You have just completed the Express Checkout portion and now must provide the ability to purchase with a credit card directly on your site. The Purchase Tickets page already contains an item to allow the purchaser to make that choice and also has a branch to page 4. You must create page 4 to accept a credit card directly.

You can again use the Create Form on Procedure wizard as a shortcut with some customization to create page 4 that accepts credit cards directly.

To use the Create Form on Procedure wizard to create the pay by credit card page:

1. Click **Create Page >** from the Application Builder home page
2. Choose **Form** from the page type and click **Next >**
3. Choose **Form on a Procedure** and click **Next >**
4. Choose the owner of the paypal\_api procedure and click **Next >**
5. Choose **paypal\_api.do\_direct\_payment** and click **Next >**
6. Enter **4** in the Page Number field
7. Enter **Pay By Credit Card** in the Page Name and Region Name fields
8. Enter **Pay Now** in the Submit Button Label field
9. Choose **Breadcrumb** in the Breadcrumb list
10. Click **Purchase Tickets** to select the Parent Entry for the breadcrumb and click **Next >**
11. Choose **Use an existing tab set and reuse an existing tab within that tab set** in the Tab Options list and click **Next >**
12. Choose **T\_PURCHASE\_TICKETS** (may also be T\_PAGE\_1) from the Use Tab list and click **Next >**
13. Leave the Invoking Page and Button Label blank and click **Next >**
14. Enter **5** in the Branch here on Submit field
15. Enter **1** in the Branch here on Cancel field and click **Next >**
16. Change the Display Type for the IP Address and Tran ID item to **Hidden** and click **Next >**
17. Click **Finish**

You can use the Drag and drop form layout feature again to quickly edit/delete multiple items. You will not need the items for the PayPal API parameters because you created application substitutions for them.

1. Click the Drag and drop icon (  ) in the Items area of the page definition of page 4
2. Drag the first seven items, **P4\_API\_URL** to **P4\_SESSION\_ID** to the trash can

3. Drag the **P4\_DESCRIPTION** item to the trash can
4. Click **Next >**
5. Change the labels of the items to be more readable, e.g., Amount
6. Click **Apply Changes**

You will make a couple of other edits to these items. P4\_AMOUNT should be display-only and have a value that matches the value on page 1. P4\_CREDITCARDTYPE should be a radio group. Finally, you will set the source of P4\_IP\_ADDRESS to be a PL/SQL procedure that will determine the IP address of the current user.

To edit the P4\_AMOUNT, P4\_CREDITCARDTYPE, and P4\_IP\_ADDRESS items:

1. Click **P4\_AMOUNT** in the Items area on the page definition of page 4
2. Choose **Display as Text (does not save state)** from the Display As list
3. Enter **class="fielddatabold"** in the HTML Table Cell Attributes field in the Element area
4. Enter **\$** in the Pre Element Text text area
5. Enter **USD** in the Post Element Text text area
6. Choose **PL/SQL Expression or Function** in the Source Type list in the Source area
7. Enter **:P1\_AMOUNT** in the Source value or expression text area
8. Click **Apply Changes**
9. Click **P4\_CREDITCARDTYPE** in the Items area
10. Select **Radiogroup** from the Display As list
11. Enter **Visa** in the Default value text area in the Default section
12. Choose **No** from the Display Extra Values list in the List of Values section
13. Choose **No** from the Display Null list
14. Enter **4** in the Number of Columns field
15. Enter **STATIC2:Visa,MasterCard,Discover,Amex** in the List of values definition text area
16. Click **Apply Changes**
17. Click **P4\_IP\_ADDRESS** in the Items area
18. Choose **PL/SQL Expression or Function** from the Source Type list in the Source area
19. Enter the following in the Source value or expression text area:  

```
owa_util.get_cgi_env('REMOTE_ADDR')
```
20. Click **Apply Changes**

Now that you have customized all the form items, you need to alter the process that was created by the wizard. You will use application-level substitutions and an item for reusability and to allow for easily changing these substitutions to point to the production PayPal payment service.

In the process, you will also obfuscate the credit card account number entered by the user so it is not stored locally.

To alter the process created by the create form on procedure wizard:

1. Click **Run Stored Procedure** in the Processes area under Page Processing on the page definition of page 4
2. Enter **DoDirectPayment** in the Name field
3. Enter the following in the Process text area:

```
declare
  l_acct varchar2(30) := :P4_ACCOUNT;
begin
  :P4_ACCOUNT := 'XXXXXXXXXXXX' || substr(l_acct, -4);

  paypal_api.do_direct_payment(
    p_api_url           => :API_URL,
    p_api_username      => :API_USERNAME,
    p_api_password      => :API_PASSWORD,
    p_signature         => :API_SIGNATURE,
    p_wallet            => :WALLET_PATH,
    p_wallet_pwd        => :WALLET_PWD,
    p_session_id        => :APP_SESSION,
    p_ip_address        => :P4_IP_ADDRESS,
    p_amount            => :P1_AMOUNT,
    p_creditcardtype    => :P4_CREDITCARDTYPE,
    p_account           => l_acct,
    p_expire_date       => :P4_EXPIRE_DATE,
    p_first_name        => :P4_FIRST_NAME,
    p_last_name         => :P4_LAST_NAME,
    p_description       => :P1_DESCRIPTION,
    p_tran_id_item      => :P4_TRAN_ID_ITEM);
end;
```

4. Click **Apply Changes**

### Create Purchase Confirmed Page for Pay By Credit Card

Lastly, you will create a purchase confirmation page similar to the express checkout confirmation page for the pay by credit card payment option. The difference on this page is that the transaction cannot be identified by the token, so it will be identified by the transaction ID.

To create the Purchase Confirmed Page:

1. Click **Create Page >** from the Application Builder home page
2. Choose **Report** from the page type list and click **Next >**
3. Choose **SQL Report** and click **Next >**
4. Enter **Purchase Confirmed** in the Page Name field
5. Choose **Breadcrumb** from the Breadcrumb list
6. Enter **5** in the Page Number field

7. Click **Purchase Tickets** to select the Parent Entry for the breadcrumb and click **Next >**
8. Choose **Use an existing tab set and reuse an existing tab within that tab set** and click **Next >**
9. Choose **T\_PURCHASE\_TICKETS** (may also be called T\_PAGE\_1) from the Use Tab list and click **Next >**
10. Enter the following in the SQL SELECT statement text area:
 

```
select transaction_id, amount
  from paypal_transactions
 where transaction_id = :P4_TRAN_ID_ITEM
```
11. Click **Next >**
12. Choose **Value Attribute Pairs** from the Report Template list
13. Enter **Purchase Confirmed** in the Region Name field and click **Next >**
14. Click **Finish**

You need to create a button on this page to allow for navigation back to the Purchase Tickets page.

To create a button for navigation back to the Purchase Tickets page:

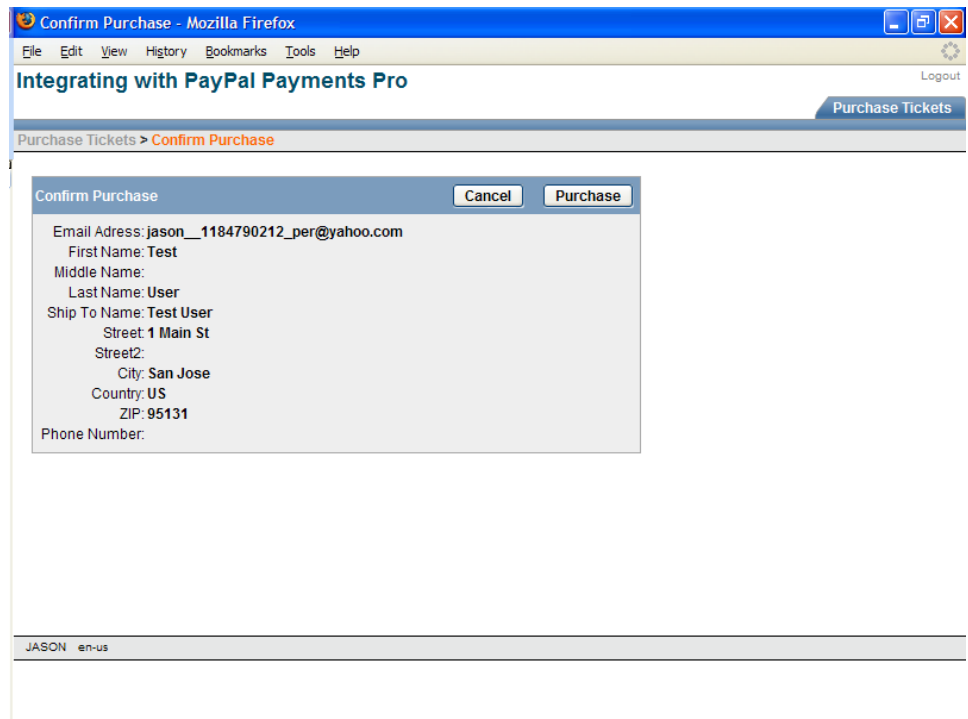
1. Click the create icon in the Buttons area on the page definition of page 5
2. Choose **Purchase Confirmed** (with the higher sequence number) from the region list and click **Next >**
3. Leave the default button position and click **Next >**
4. Enter **PURCHASE** in the Button Name field
5. Enter **Purchase More Tickets** in the Label field
6. Select **Redirect to URL without submitting page** from the Action list and click **Next >**
7. Leave the button template defaulted and click **Next >**
8. Leave the button position defaulted and click **Next >**
9. Enter **1** in the Page field
10. Enter **1** in the Clear Cache field
11. Click **Create Button**

## TESTING THE TICKETING APPLICATION

Your application is now complete and you are ready to test it. Prior to running the application, you must go to <https://developer.paypal.com/> and sign in with your developer account. After logging in, use the same browser to run your ticketing application. Note that this is not the experience your users will encounter when you change the application to work with PayPal's live site and APIs. This is only a requirement for the Sandbox.

Test both PayPal and Credit Card payment methods. You must use the personal test account you created in the Prerequisites section. You can get the credit card account number and expiration date to use by going to the Sandbox, Test Accounts tab, and click View Details under the personal test account that you created. Note that the expiration date you enter should be in the form MMYYYY.

Finally, use the Sandbox to login in to the business test account and verify that you can see the recently completed transactions.



Oracle Application Express Free Trial Service

<http://htmldb.oracle.com>

Oracle Application Express Home

<http://htmldb.oracle.com/otn>

Oracle Application Express How To Documents

<http://htmldb.oracle.com/howtos>

Oracle Application Express Studio

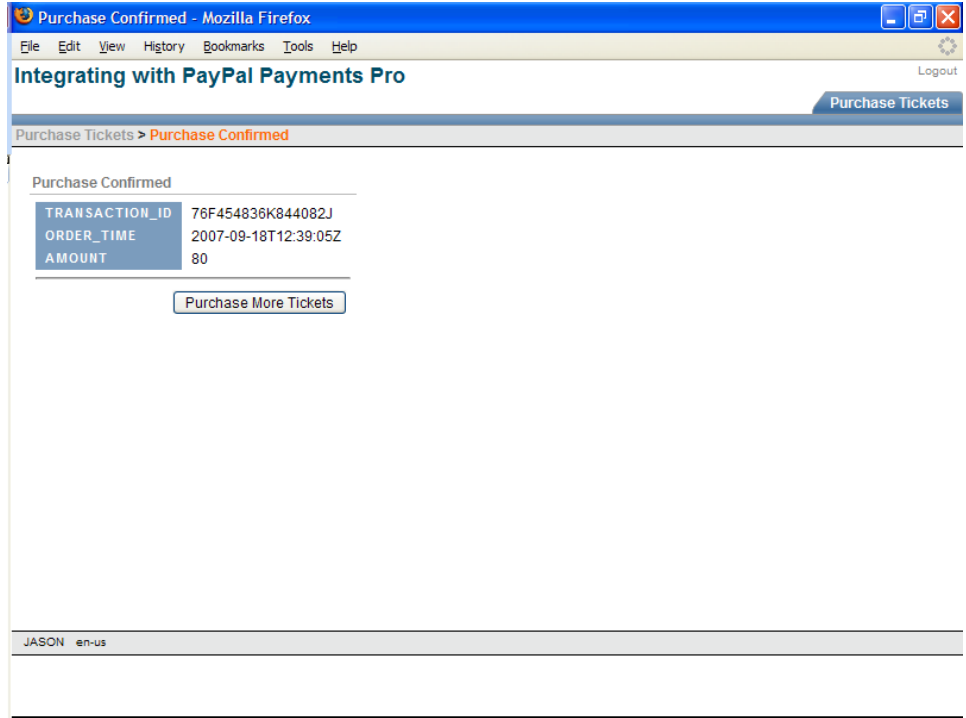
<http://htmldb.oracle.com/studio>

Oracle Application Express Forum on OTN

<http://htmldb.oracle.com/forums>

Oracle Application Express References

<http://htmldb.oracle.com/references>



## CONCLUSION

Oracle Application Express allows for building applications that integrate with payment solutions such as PayPal Website Payments Pro through the use of supplied PL/SQL packages like UTL\_HTTP. It is possible to use the same methodology to integrate with other payment solutions that use a name value pair API.



Integrating Application Express with PayPal Payments Pro  
September 2007  
Author: Jason Straub

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200  
[oracle.com](http://oracle.com)

Copyright © 2007, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle, JD Edwards, and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.