



PHP SDK Guide

For Professional Use Only
Currently only available in English.

A usage Professional Uniquement
Disponible en Anglais uniquement pour l'instant.

Notice of Non-Liability

PayPal, Inc. and the authors assume no liability for errors or omissions, or for damages, resulting from the use of this Manual or the information contained in this Manual.



Contents

Preface	7
Chapter 1 Installation	9
Configuring PHP for the SDK	9
Supported Standards	9
Supported Human Languages	10
SDK Version Number	10
Minimum Hardware Requirements	10
Complementary Documentation	10
Downloading and Installing the SDK	11
Steps for Installing the PHP SDK	11
Chapter 2 SDK Components	13
API Services	13
Complete Class Documentation	14
Requesting API Credentials from PayPal	14
EWP Services	16
Obtaining EWP Access	16
Payment FORM Variables	17
PayPal Public Certificates	17
Profiles	17
Overview to Profile-related Classes	18
Profile Storage	18
Chapter 3 Example Program: Searching Transactions	21
Chapter 4 The SDK Web Console	23
Accessing the Console	23
Calling an API with the Web Console	23
Creating an API Profile	24
Calling an API	25

Appendix A	SDK Directories and Configuration Files31
	Configuration Files	32
	SDK Logging.	32
Appendix B	Updating and Building the SDK33
	Steps to Update the SDK	33



List of Tables

Table 1.1	Supported Standards	9
Table 1.2	Minimum System Hardware Requirements	10
Table 2.1	Summary of PHP SDK Profile-related Classes	18
Table A.1	PayPal SDK for PHP: Directories and Contents	31
Table A.2	Services_PayPal Subdirectories and Contents	31
Table A.3	SDK Logging Levels	32
Table B.1	Commands for Updating the PHP SDK.	33

Preface

This Document

This release of the *PayPal PHP Software Development Kit (SDK) Guide*, a document that describes the PayPal SDK for developers of business applications using the PayPal Web Services API, replaces the version released in July, 2005.

Intended Audience

This document is written for business application programmers familiar with web and application servers, databases, the relevant programming language, and Web Services application programming, including WSDL, XSD, and SOAP.

Organization of This Document

[Chapter 1, “Installation,”](#) describes the standards supported by the SDK, how to install it, and post-installation tasks.

[Chapter 2, “SDK Components,”](#) describes the core components of the SDK and how they work.

[Chapter 3, “Example Program: Searching Transactions,”](#) is a sample program that uses the SDK.

[Chapter 4, “The SDK Web Console,”](#) gives how-to information about using the SDK console test and learning tool.

[Appendix A, “SDK Directories and Configuration Files,”](#) details the configuration files that control the SDK.

[Appendix B, “Updating and Building the SDK,”](#) details how to update the SDK.

Notational Conventions

This document uses typefaces to identify the characteristics of text. These typefaces and the characteristics they imply are described below:

Typeface	How Used
<i>serif italics</i>	A document title.
	A term being discussed or defined. For example: A <i>file</i> is a readable or writable stream of characters ...
	Boolean values (not keywords). For example: The function returns <i>true</i> if it encounters an error.

Typeface	How Used
monospaced	<p>Pathnames or file names that appear in body text frames.</p> <p>Code-related names that appear in body text frames. Such names are used for functions, callbacks, arguments, data structures, and fields.</p> <p>For example: <code>AbstractResponseType</code> is the SOAP response type definition on which all PayPal API response methods are based.</p> <p>Components of Internet protocol requests and responses, such as HTTPS and FORM variables.</p> <p>For example: The PayPal system uses a <code>method=POST</code> request to return IPN status variables related to subscriptions, such as <code>txn_type</code>.</p>
Serif bold	<p>User interface names, such as window names or menu selections.</p> <p>For example: On the Profile page, click Email to confirm your email address.</p>
<i>San-serif oblique</i>	<p>Placeholders used in the context of a format or programming standard or formal descriptions of PayPal system syntax. Placeholders indicate values or names that the reader should provide.</p> <p>Example: For example, <code>amount</code> is the variable for a single-item shopping cart, but <code>amount_X</code> is the name of the variable for a multi-item shopping cart. <code>amount_3</code> is the item amount for the third item in a multiple-item shopping cart.</p>

To convey additional information, this document may also apply color and underlining to words or phrases that use the typefaces described above. Such use is described below:

Text attribute	How Used
xxxxxx	Hypertext link to a page in the current document or to another document in the set.
xxxxxx	Hypertext link to a URL or that initiates a web action, such as sending mail.

Documentation Problems

If you discover any errors in or have any problems with this documentation, please e-mail us by following the instructions below. Describe the error or problem as completely as possible and give us the document title, the date of the document (located at the foot of every page), and the page number or page range.

To contact Developer Technical Support about documentation problems:

1. Log in to your account at <https://developer.paypal.com/> by entering your email address and password in the **Member Log In** box
2. Click **Help Center** at the bottom of the box on the right side of the page.
3. Click **Email PayPal Technical Support**.
4. Complete the form.

1

Installation

This chapter details the software and hardware supported and required by the PayPal SDK, installation, and post-installation tasks.

Configuring PHP for the SDK

Configure PHP as follows to use the PayPal SDK:

1. Disable safe mode features.
2. Do not use the `disabled_functions` and `disabled_classes` settings.
3. Enable the following extensions:
 - PHP Perl Compatible Regular Expressions extension for PHP 4.3.0+ and higher
 - PHP cURL extension for PHP 4.3.0+ and higher with SSL support
 - PHP OpenSSL extension for PHP 4.3.0+ and higher (for digital certificate transcoding)
4. Ensure that PEAR has the following required packages:
 - `NET_Url`
 - `NET_Socket`
 - `HTTP_Request`
 - `Log`

Supported Standards

The PayPal SDK has been verified to work with the following standards.

TABLE 1.1 *Supported Standards*

Standard	Version
PHP with PHP Extension and Application Repository (PEAR)	4.3.*
Simple Object Access Protocol (SOAP)	1.1
Web Services Description Language (WSDL)	1.1

Supported Human Languages

The PayPal SDK is available in U.S. English.

SDK Version Number

This guide describes PayPal PHP SDK version 2.0.

Minimum Hardware Requirements

The following table lists the minimum hardware requirements for using the PayPal SDK in development and test. Production systems might require more capacity, depending on their expected load.

TABLE 1.2 Minimum System Hardware Requirements

Component	Minimum Capacity
RAM	256 MB
CPU	Pentium 1 GHz
Disk space	50 MB

Complementary Documentation

The *PayPal Sandbox User Guide*, the *PayPal SDK Guides*, and the *PayPal Web Services API Reference* are complementary documentation.

NOTE: Users of the PayPal Web Services API and the PayPal SDK need all three of these guides.

- The *PayPal Sandbox User Guide* (https://www.paypal.com/en_US/pdf/PP_Sandbox_UserGuide.pdf) is for merchants and third-party developers who need to write and test their PayPal applications. It describes how to access PayPal's virtual testing environment (called "the Sandbox"), with information about the following:
 - How to get digital certificates for securely using the Sandbox, the live PayPal Web Services API service
 - Different types of business roles for effective use of the Sandbox and how to set up test accounts that correspond to those roles
 - Setting up Sandbox email to verify test PayPal transactions
 - Viewing PayPal-generated email sent to different business roles, depending on different types of transactions

- The *PayPal SDK Guides* (<https://www.paypal.com/sdk/>) describe PayPal's kit for writing applications based on PayPal's Web Services API. Supporting Java, Microsoft .NET Framework, and PHP with PEAR, the SDK shields the developer from explicitly dealing with SOAP request and response structures, who does not have to marshal and unmarshal SOAP XML. The SDK includes:
 - Administrative tools for managing PayPal API user profiles
 - Logging and other features for troubleshooting during test or production
 - A sample website integration that showcases many PayPal Web Services APIs and other features
 - Auto-updating to easily keep current with enhancements to PayPal Web Services
- The *PayPal Web Services API Reference* (https://www.paypal.com/en_US/pdf/PP_APIReference.pdf) is for programmers familiar with Web Services. It includes:
 - Technical details about the PayPal API
 - Data types, date/time formats, character sets, supported currency and country codes, and single-merchant and third-party authentication
 - WSDL and XSD definitions and structures for all PayPal API operations
 - SOAP request elements with required or allowable values
 - SOAP response structures with definitions of returned values
 - Error codes and messages

Downloading and Installing the SDK

The latest version of the PayPal SDK is available at <https://www.paypal.com/sdk/>. You can download a zipfile distribution.

The SDK consists of the following major components:

- SOAP libraries for the PayPal APIs
- The web console test tool
- API and EWP profile administration programs

Steps for Installing the PHP SDK

Follow steps below to install the PHP SDK.

NOTE: For complete step-by-step instructions for making your first API call with the SDK console, see the `QuickStart.txt` file in the SDK installation directory.

1. Be sure you have configured PHP and your web server as detailed in “[Configuring PHP for the SDK](#)” on page 9.
2. Unzip and extract the zipfile.

Throughout this document, the directory in which you choose to extract the SDK is referred to as follows:

SDK_root

3. Use pear to install the SDK SOAP libraries:

```
cd SDK_root/Services_PayPal  
pear install package.xml
```

4. Copy the WebConsole directory to your web server's document root. In this example, the user is running Apache on UNIX with the default document root:

```
cp -r WebConsole /usr/local/apache/htdocs/php-sdk
```

SDK Contents

For a description of the directories and configuration files included in the PayPal SDK distribution, see [Appendix A, “SDK Directories and Configuration Files.”](#)

2

SDK Components

The SDK is composed of *API Services*, *Encrypted Website Payments (EWP) Services*, and user *profiles* that define characteristics of the users of those services. This chapter describes these primary concepts, their core definitions, the prerequisite set-up steps for creating SDK profiles, and the location of complete SDK class documentation.

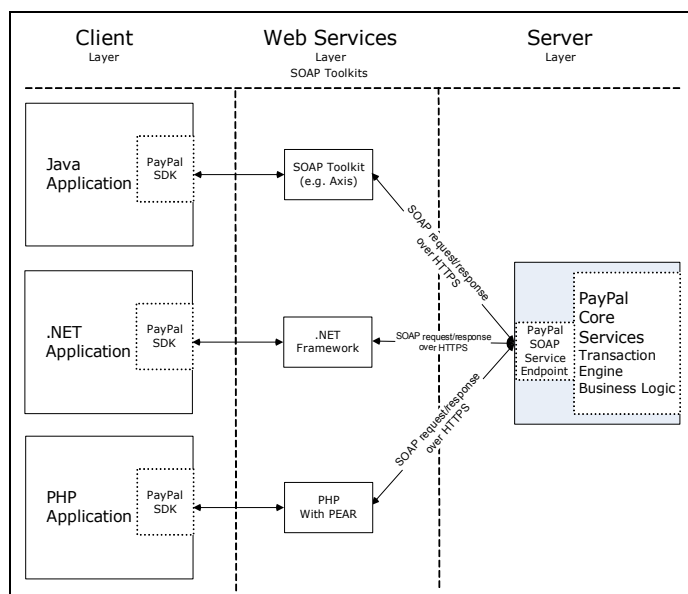
UTF-8 BOM and the SDK

IMPORTANT: Do not specify a UTF-8 byte-order mark (BOM) for any UTF-8 strings you use with the SDK.

API Services

Integrated access to the PayPal Web Services APIs is one of the main features of the PayPal SDK. As shown in [Figure 2.1, “PayPal SDK API Services: High-Level View” on page 13](#), the PayPal SDK uses the underlying platform’s SOAP toolkit to communicate with the PayPal API endpoint. Stub classes representing the requests, the responses, and their data are provided to set request parameters and read response values. These classes are auto-generated from the PayPal WSDL file and can be updated when upgrades to the PayPal WSDL are available (see [Appendix B, “Updating and Building the SDK”](#)).

FIGURE 2.1 PayPal SDK API Services: High-Level View



Complete Class Documentation

Complete documentation for all PayPal SDK interfaces, classes, methods, structures, and data types are included with the SDK distribution. To view the documentation, open the following file with your web browser:

`SDK_root/Services_PayPal/docs/apidoc/index.html`

If you have installed the PHP SDK with PEAR, the documentation is located in your PHP library's `doc` directory.

Requesting API Credentials from PayPal

API credentials consist of the following:

- API username that identifies you to the PayPal Web Services API service
- API password for that API username
- PayPal-issued digital certificate. To ensure security for your customers and your business, a public certificate and private key issued by PayPal are required for use of the PayPal Web Services API, SDK, and Sandbox. The certificate file is in PEM format, contains both your private key and your public certificate, and must be converted to PKCS12 format.

To obtain a PayPal Web Services API username, password, and digital certificate, you must first create a Business or Premier account and apply online in either the Sandbox or the live PayPal site, <https://www.paypal.com>.

To get a Sandbox certificate. PayPal automatically issues digital certificates for use with the Sandbox.

1. With your PayPal Business or Premier account email address, log in to Developer Central at <https://developer.paypal.com/>.
2. Launch the Sandbox.
3. Log in to the Sandbox
4. Navigate to **My Account > Profile**.
5. Click the **API Access** link.
6. Click the **API Certificate Request** link.
7. Follow the on-screen instructions to complete your certificate request.

To download your PayPal API test certificate:

8. Log in to Developer Central.
9. Click the **Test Certificates** tab.
10. Click **Download** at the lower right.

11. Save the file to your local disk.

To get a live PayPal certificate. The steps to get a live certificate are the same as those for a Sandbox certificate, except for Step 1:

- Login to <https://www.paypal.com/> to request the certificate.

You cannot immediately download a live certificate. For use in production with the live PayPal API service, PayPal must verify the authenticity of the credentials of a certificate request before issuing the certificate. PayPal notifies you by email when your certificate request has been approved; after you receive this email message, you can download the PayPal-issued certificate from your PayPal account.

Where to Store your Certificate Files. The certificate files do not need to be stored in any specific location as long as you and your applications know the path to the file. As with any digital certificate, you should exercise proper caution and follow common security practices to keep the certificate and keyfile safe.

EWP Services

The PayPal SDK's Encrypted Website Payments (EWP) Service generates encrypted HTML "Buy Now" button code that ensures the security, integrity, and authenticity of data posted to the PayPal site when a customer clicks the button. The SDK's cryptography algorithm first signs the data using the merchant's private key and then encrypts the button parameters using the PayPal public key included with the SDK. After receiving the POST, PayPal decrypts the message using the PayPal private key and verifies the signature using the public key provided by the merchant.

This section describes the steps to set up EWP and the SDK's tools for creating private keys and public certificates.

Obtaining EWP Access

For EWP, PayPal requires that you upload your public certificate to its website, so that the authenticity of the encrypted code can be verified. Your public certificate must be in PEM format.

Uploading Your Public Certificate

After you create a public certificate, you must upload it to PayPal. After uploading, PayPal displays a certificate ID that you need to use when you encrypt your button HTML.

To upload a public certificate to the PayPal website:

1. Log in to your Business or Premier PayPal account.
2. Click the **Profile** subtab.
3. Click the **Encrypted Payment Settings** link from the right-handed menu.
4. Click **Add**.
5. Click **Browse** and select the public certificate you want to upload.

The public certificate must be in PEM format.

If your public certificate is successfully uploaded, it will appear on the next screen under **Your Public Certificates**.

Your Public Certificates

PayPal will use your public certificate to decipher the encrypted content of your website buttons. You may add up to 6 different certificates.

Cert ID	Certifying Authority	Expiration Date
MGH3RXHNMXZSL	/C=US/ST=NEBRASKA/L=OMAHA/O=TESTING/OU=IT/CN=PAU/Email=your email	Jul. 22, 2005 20:43:24 PDT

Payment FORM Variables

For details about allowable FORM name/value pairs for use with EWP, see the “HTML Samples” appendix of the *Website Payments Standard Integration Guide*, available at https://www.paypal.com/en_US/pdf/PP_WebsitePaymentsStandard_IntegrationGuide.pdf.

PayPal Public Certificates

The PayPal SDK includes PayPal’s public certificates for both `www.paypal.com` and `sandbox.paypal.com`. See the `SDK_root/certSDK_root/Services_PayPal/cert` directory.

Profiles

Before the SDK can be used, it must know the profile of the user accessing its services. A *profile* is a collection of information about a merchant or developer who uses the PayPal SDK. The SDK has two kinds of profiles: API profiles and Encrypted Website Payments (EWP) profiles.

1. An *API profile* is associated with API Services and includes:

- A PayPal API username and password
- The path to the API certificate in PEM format as downloaded from PayPal
- The optional name of a third-party who authorizes the caller to invoke PayPal APIs on his behalf. This third-party is called a *subject*.
- The PayPal environment for processing API calls: `live` or `sandbox`.

NOTE: For information about obtaining an API username, password, and digital certificate, see “[Requesting API Credentials from PayPal](#)” on page 14.

2. An *EWP profile* is associated with EWP Services includes:

- The path to the merchant’s local copy of that public certificate
- The private key password for that public certificate
- The path to a merchant’s private key file for digitally signing data
- The URL to which the button form POSTs
- The optional URL of a payment button image. The default is PayPal’s standard Buy Now button.

NOTE: For more information about how EWP works, see the *Website Payments Standard Integration Guide*, available at https://www.paypal.com/en_US/pdf/PP_WebsitePaymentsStandard_IntegrationGuide.pdf.

Overview to Profile-related Classes

The primary interfaces and classes for SDK profiles are described in [Table 2.1, “Summary of PHP SDK Profile-related Classes.”](#)

TABLE 2.1 *Summary of PHP SDK Profile-related Classes*

Interface/Class	Descriptions
APIProfile	This class defines the basic information that PayPal needs to know about a user of the PayPal Web Service APIs. Developers must create an instance of APIProfile for each account that accesses the APIs. For single-merchant developers, only a single APIProfile instance is needed. PayPal provides a default implementation suitable for the needs of most SDK developers. However, you are free to write a custom implementation if you need additional functionality the default class does not offer.
EWPPProfile	This class defines the basic information that PayPal needs to know about a user of PayPal’s Encrypted Website Payments (EWP) service. Developers must create an instance of EWPPProfile for each account that generates the encrypted button code. PayPal provides a basic implementation suitable for the needs of most SDK developers. However, you are free to write a custom implementation if you need functionality the default class does not offer.
ProfileHandler	The ProfileHandler class manages the storage and retrieval of APIProfile and EWPPProfile objects. PayPal provides two basic implementations: ProfileHandler_Array and ProfileHandler_File to serialize and deserialize the data in profiles to and from a file on the disk. NOTE: For security, the ProfileHandler_File implementation does not store an API user’s API password.

Profile Storage

To save you from having to enter profile data every time you use the SDK, the SDK includes a mechanism to save and retrieve profiles. This a feature of the SDK you might want to customize.

ProfileHandler_File, the default storage mechanism, is rudimentary: it serializes profile data in memory to a file on disk. Although file storage can be somewhat insecure and error-prone, it is provided for quick implementation of the SDK after it is installed. You might want to store your profiles in a database, in secure XML files, or some other storage mechanism.

For program development, the SDK includes an alternative profile handler:

ProfileHandler_Array. This handler accepts an array of values representing data that

would be loaded from the data store by the default handler `ProfileHandler_File` but without actually storing those values on disk or relying on the other profile management functions.

Best Practice

Although the SDK can be used without a profile handler to store and retrieve profiles, best practices and common security measures encourage the use of a separate class for this function. Early in development, you should decide the best mechanism for your own profile management.

Configuring the SDK with Your Profile Handler

To configure the SDK with your own profile storage mechanism, do the following:

1. Create a class that extends the default abstract class `ProfileHandler_File`.
2. Be sure your custom class follows the naming conventions detailed in the configuration file described in [“Configuration Files” on page 32](#):
 - Your handler class must be named `ProfileHandler_yourNameHere`.
 - Your handler class must be in a file named `yourNameHere.php`.
 - Your handler class must be stored in the directory you specify in the `custom_handler_dir` system property.
 - Set the PayPal SDK system property `custom_handler_dir` to the full path of the directory holding your custom class.

3

Example Program: Searching Transactions

The PayPal SDK includes example code in the *SDK_root/Services_PayPal/docs/examples* directory.

This chapter presents one of them: an example Java program to show how easy it is to use the PayPal SDK. The sample program uses the *TransactionSearch* API operation to search for PayPal transactions and displays the response (the transactions that match the search criteria). The line numbers on the left of the example are for reference.

EXAMPLE 3.1 Annotated Example PHP Program

Code	Comment
1. <?php	Invoke PHP
2.	
3. require_once './request_base.php';	Include a module that sets up a basic API request. See below.
4.	
5. \$ts =& Services_PayPal::getType('TransactionSearchRequestType');	This application performs a transaction search. Here it creates a new <i>TransactionSearchRequestType</i> object called <i>ts</i> .
6. \$ts->setStartDate(date('Y-m-d') . 'T00:00:00-0700');	Use the <i>setStartDate()</i> method to set the value of <i>ts</i> object's required element <i>StartDate</i> .
7.	
8. \$response = \$caller->TransactionSearch(\$ts);	Call the <i>TransactionSearch</i> API by referencing the caller object with the operation <i>TransactionSearch</i> and the <i>ts</i> object. The caller object was set in <i>request_base.php</i> .
9. var_dump(\$response);	Display the results. See <i>SDK_root/doc</i> for full list of available result properties to display.

EXAMPLE 3.2 Annotated request_base.php

Code	Comment
1. <?php	
2. require_once 'Services/PayPal.php';	Import the base SDK services packages
3. require_once 'Services/PayPal/Profile/Handler/Array.php';	Import Profile Handler array routines
4. require_once 'Services/PayPal/Profile/API.php';	Import the default API profile routines
5.	
6. // Settings.	
7. \$certfile = dirname(__FILE__) . '/sdk-seller_cert.pem';	Set the path to the API certificate

EXAMPLE 3.2 *Annotated request_base.php*

Code	Comment
8. \$certpass = '';	The private key for API certificates is not implemented at this time.
9. \$apiusername = 'sdk-seller_api1.sdk.com';	
10. \$apipassword = '12345678';	
11. \$subject = null;	If this program were run on behalf a third party, subject must be set to the email address of that third-party.
12. \$environment = 'Sandbox';	Because this is a test application, it will be run against the PayPal sandbox endpoint, not the live PayPal API endpoint.
13.	
14. \$handler =& ProfileHandler_Array::getInstance(array(Populate a ProfileHandler array object with the values previously set
15. 'username' => \$apiusername,	
16. 'certificateFile' => \$certfile,	
17. 'subject' => \$subject,	
18. 'environment' => \$environment));	
19.	
20. \$profile =& APIProfile::getInstance(\$apiusername, \$handler);	
21. \$profile->setAPIPassword(\$apipassword);	Set the API password object
22.	
23. \$caller =& Services_PayPal::getCallerServices(\$profile);	Create a caller object
24.	
25. if (Services_PayPal::isError(\$caller))	Some simple error handling.
26. {	
27. print "Could not create CallerServices instance: ". \$caller->getMessage();	
28. exit;	
29. }	

4

The SDK Web Console

The PayPal SDK includes a self-contained console for testing basic features of the SDK. With the console, you can:

- Manage your API profiles
- Generate PHP code for making API calls and make actual API calls

The web console displays the API code it generates. You can copy and paste the generated code into a PHP file for use on your website.

The web console can also make actual PayPal Web Services API calls. It is a good tool for testing that your environment is properly configured.

This chapter includes the following sections:

- [Accessing the Console](#)
- [Calling an API with the Web Console](#)

Accessing the Console

You access the web console with your browser. The exact URL depends on where your web server administrator installed the web console. If the default instructions in this guide were followed, the URL is like the following:

`http://webServerName/php-sdk/WebConsole/index.php`

Calling an API with the Web Console

Here is the general process for calling an API with the web console:

1. Create an API profile.
2. Select the active API profile.
3. Display an API operation's request template.
4. Manually fill-in the values of the template's required and optional elements.

NOTE: For complete details about required and optional request elements, consult the *PayPal Web Services API Reference*.

5. Display the generated PHP code and run it.

The remainder of this section presents an example of these steps to call the TransactionSearch API.

Creating an API Profile

The first step in calling APIs with the console is to create an API profile. For the complete definition of an API profile, see [“Profiles” on page 17](#).

In this example, the user has already obtained an API username, password, and PayPal-issued digital certificate for use with the Sandbox, as detailed in [“Requesting API Credentials from PayPal” on page 14](#).

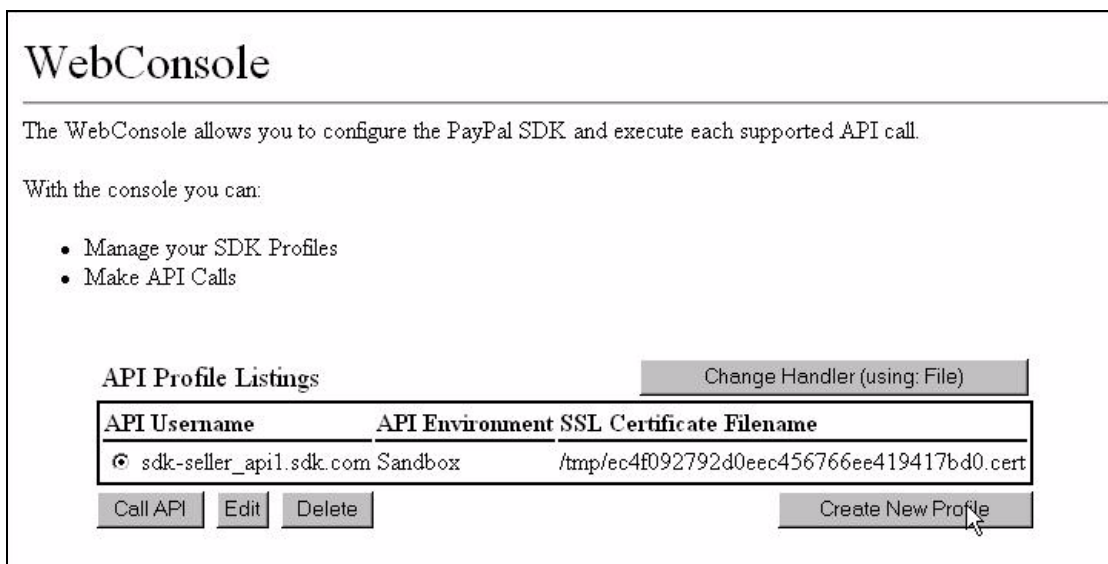
The complete details about the API user in this example are as follows:

- The API username is `developer-fictional.com`.
- The API user has already downloaded a digital certificate from PayPal.
- The API call will be processed against the Sandbox, not the live PayPal API service.

The user has already started the console, as described in [“Accessing the Console” on page 23](#).

As shown in [Figure 4.1, “Adding an API Profile: Step 1,”](#) to add a new API profile, the user clicks **Create New Profile**.

FIGURE 4.1 Adding an API Profile: Step 1

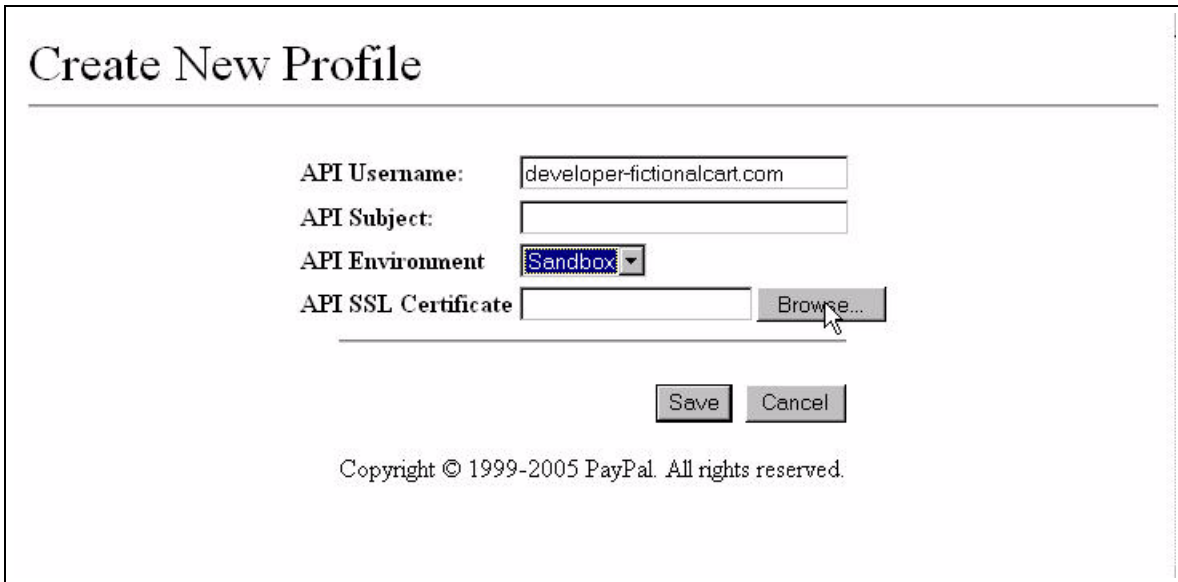


In the displayed form (see [Figure 4.2, “Adding an API Profile: Step 2”](#)), the user enters or selects the following information:

- The API username
- No **Subject**: the user is creating this profile in order to make API calls himself, not on behalf of a third-party, so the subject is left blank.

- The environment: **Sandbox**
- For the certificate, the user browses his local disk and selects the downloaded PayPal PEM file.

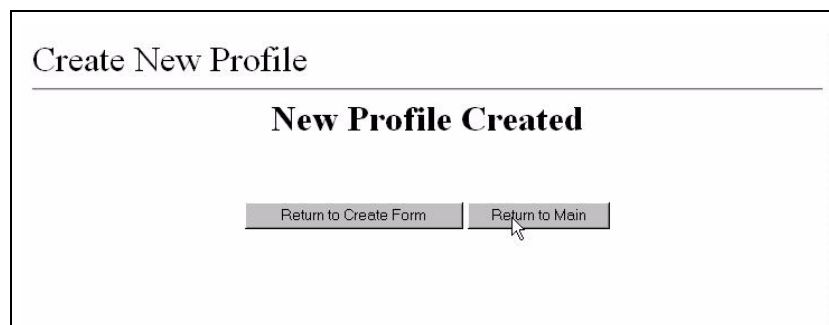
FIGURE 4.2 Adding an API Profile: Step 2



The user clicks **Save** to create the profile.

On the displayed page shown in [Figure 4.3](#), “Adding an API Profile: Step 3,” the user clicks **Return to Main** to view the web console front page.

FIGURE 4.3 Adding an API Profile: Step 3



Calling an API

Before calling an API, the user must select the API profile for whom the call will be made. In [Figure 4.4](#), “Selecting the Active API Profile,” the user selects the `developer-fictionalcart.com` profile and clicks **Call API**.

FIGURE 4.4 Selecting the Active API Profile

WebConsole

The WebConsole allows you to configure the PayPal SDK and execute each supported API call.

With the console you can:

- Manage your SDK Profiles
- Make API Calls

API Profile Listings Change Handler (using: File)

API Username	API Environment	SSL Certificate Filename
<input checked="" type="radio"/> developer-fictionalcart.com	Sandbox	/tmp/d39f72c7fd75ab8397791d0de04c3678.cert
<input type="radio"/> sdk-seller_api1.sdk.com	Sandbox	/tmp/ec4f092792d0eec456766ee419417bd0.cert

Call API Edit Delete Create New Profile

EWP Profile Listings Change Handler

Private Key File	SSL Certificate Filename
None	

Edit Delete Create New Profile

On the page displayed in [Figure 4.5](#), “Entering the API Password for Active Profile,” the user enters the API password for the `developer-fictionalcart.com` API user and clicks **Save**.

FIGURE 4.5 Entering the API Password for Active Profile

WebConsole API Tester

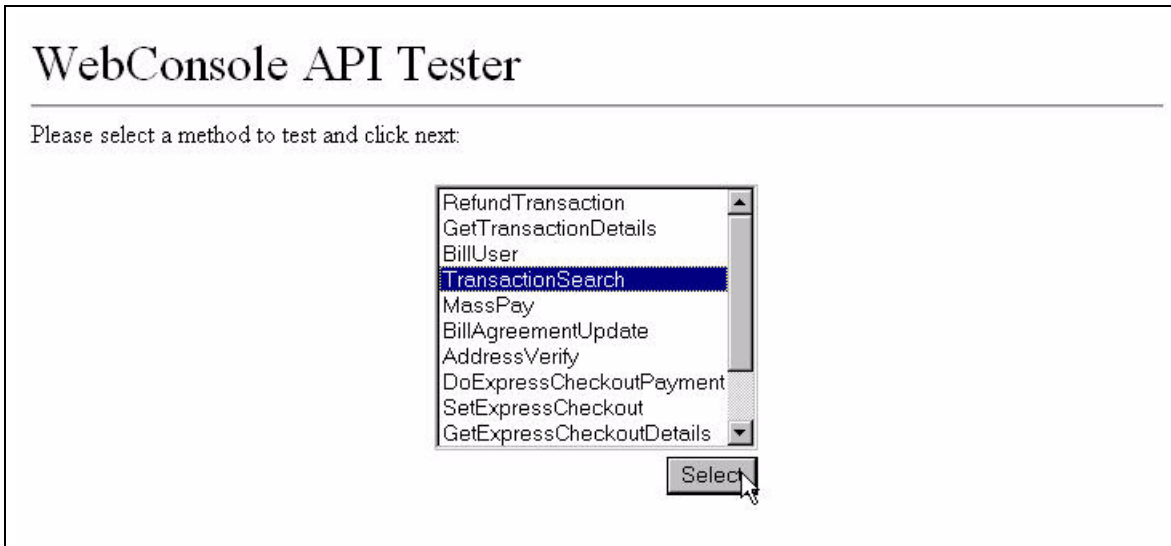
Please enter the password for the current API profile (developer-fictionalcart.com):

Save

Copyright © 1999-2005 PayPal. All rights reserved

In [Figure 4.6](#), “Selecting the API Operation,” the user selects the `TransactionSearch` method.

FIGURE 4.6 Selecting the API Operation



The web console displays the TransactionSearch request template, as shown in Figure 4.7, “Filling In the Operation Fields.” The request template contains all possible elements that can be included in the TransactionSearch operation. However, not all values are required; some are optional. For TransactionSearch only the StartDate element is required.

IMPORTANT: *The PayPal Web Services API Reference details the required or optional elements for all operations. To populate the values of the request template, consult the definitions of elements in PayPal Web Services API Reference to determine which element values you need to supply.*

FIGURE 4.7 Filling In the Operation Fields

WebConsole API Tester

Please fill out the required values. Complex types which are subtypes of the current type will be available on following screens. Items marked in **bold** are required elements, and elements with checkboxes next to them are optional complex types. If you would like to use them in your request, please check the box before continuing on to the following screen.

TransactionSearch	
Status	<input type="text"/>
CurrencyCode	<input type="text"/>
Amount	<input type="checkbox"/> <i>Complex Type (next page)</i>
TransactionClass	<input type="text"/>
InvoiceID	<input type="text"/>
AuctionItemNumber	<input type="text"/>
PayerName	<input type="checkbox"/> <i>Complex Type (next page)</i>
TransactionID	<input type="text"/>
ReceiptID	<input type="text"/>
Receiver	<input type="text"/>
Payer	<input type="text"/>
EndDate	<input type="text"/>
StartDate	<input type="text" value="2005-05-17T21:08:51Z"/>
MessageID	<input type="text"/>
ErrorLanguage	<input type="text"/>
DetailLevel	<input type="text"/>

After filling in the desired fields, the user clicks **Next**.

The web console then displays the generated PHP code, as shown in [Figure 4.8](#), “Viewing the Generated API Call Code.”

FIGURE 4.8 Viewing the Generated API Call Code

WebConsole API Tester

The following is an automatically generated PHP script based on your input which should allow you to test the SDK. Cut and paste this code into a PHP file to use it.

Run this code
Change arguments
Build another snippet

```

<?php
require_once 'Services/PayPal.php';
require_once 'Services/PayPal/Profile/Handler/File.php';
require_once 'Services/PayPal/Profile/API.php';

$handler =& ProfileHandler_File::getInstance(array (
    'path' => '/tmp',
));

if (Services_PayPal::isError($handler)) {
    var_dump($handler->getMessage());
    exit;
}

$profile =& APIProfile::getInstance('ec4f092792d0eec456766ee419417bd0', $handler);

if (Services_PayPal::isError($profile)) {
    var_dump($profile->getMessage());
    exit;
}

$profile->setAPIPassword('12345678');

$caller =& Services_PayPal::getCallerServices($profile);

if (Services_PayPal::isError($caller)) {
    var_dump($caller->getMessage());
    exit;
}
    
```

Finally, the user decides to run the generated code. The results of the call are shown in [Figure 4.9, “Viewing the API Response.”](#)

To determine if the call was successful, the user scrolls down the displayed page to see the `Ack` response code **Success**.

FIGURE 4.9 Viewing the API Response

The screenshot shows the 'WebConsole API Tester' interface. At the top, there are three buttons: 'Run again', 'Change arguments', and 'Build another snippet'. Below these buttons, the API response is displayed in a text area. The response is a JSON object representing a transaction search result. The 'Ack' field is highlighted with a green circle, showing the value 'Success'. The 'CorrelationID' field is also visible, showing a long alphanumeric string.

```

object (transactionsearchresponsetype) (11) {
  ["PaymentTransactions"]=>
  NULL
  ["_elements"]=>
  array(7) {
    ["Timestamp"]=>
    array(3) {
      ["required"]=>
      bool(false)
      ["type"]=>
      string(8) "dateTime"
    }
  }
  ["_attributes"]=>
  array(0) {
  }
  ["_attributeValues"]=>
  array(0) {
  }
  ["_namespace"]=>
  string(22) "urn:ebay:api:PayPalAPI"
  ["Timestamp"]=>
  string(20) "2005-05-17T21:11:41Z"
  ["Ack"]=>
  string(7) "Success"
  ["CorrelationID"]=>
  NULL
  ["Errors"]=>
  NULL
  ["Version"]=>
  string(8) "1.000000"
  ["Build"]=>
  string(6) "1.0006"
}
  
```



SDK Directories and Configuration Files

The PHP SDK components are organized into different subdirectories, as shown in [Table A.1](#), “PayPal SDK for PHP: Directories and Contents.”

TABLE A.1 *PayPal SDK for PHP: Directories and Contents*

Directory	Description
Services_PayPal	Primary SDK code. Subdirectories are described in Table A.2 , “Services_PayPal Subdirectories and Contents.”
WebConsole	Web console programs. Must be copied to web server’s document root.

TABLE A.2 *Services_PayPal Subdirectories and Contents*

Directory in Services_PayPal	Description
docs	SDK documentation: examples, sample XML, and API documentation
PayPal	Main SDK classes, configuration files, and ancillary files
PayPal/cert	PayPal public certificates
PayPal/conf	SDK system properties configuration file. See “Configuration Files” on page 32 .
PayPal/Profile	APIProfile and EWPPProfile implementations and the ProfileHandler definitions, with two default profile handlers: ProfileHandler_File and ProfileHandler_Array.
PayPal/SDK	This directory and SDK.php are used to rebuild the SDK from the PayPal WSDL. For information, see Appendix B , “Updating and Building the SDK.”
PayPal/SOAP	PEAR::SOAP implementation. No PayPal-specific code.
PayPal/Type	Auto-generated data type files from which the PayPal type classes inherit, and two parent classes: XSDType and XSDSimpleType.
PayPal/wsd1	PayPal WSDL/XSD files and paypal-endpoints.php

Configuration Files

The PayPal SDK is distributed with the configuration properties file
`SDK_root/Services_PayPal/PayPal/conf/paypal-sdk.php.dist`.

If you want to change the configuration, copy this file to:

`SDK_root/Services_PayPal/PayPal/conf/paypal-sdk.php`

SDK Logging

Setting Log Levels

The SDK varies the amount of detail it logs according to three logging levels.

Set the value of the `log_level` parameter in the
`SDK_root/Services_PayPal/PayPal/conf/paypal-sdk.php` file.

TABLE A.3 SDK Logging Levels

Level	Description
PEAR_LOG_ERR	Log only severe errors
PEAR_LOG_INFO	Default log level. Date/time of API operation, operation name, elapsed time, success or failure indication.
PEAR_LOG_DEBUG	<p>Full text of SOAP requests and responses and other debugging messages.</p> <p>NOTE: Because SOAP requests and responses are asynchronous, the recording of requests and responses might appear out of sequence in the log file.</p> <p>Because DEBUG logging can degrade the performance of the SDK, be careful about using it for day-to-day operation.</p>

Logfile Location

In the `log_dir` parameter, set the name of the directory where the SDK should write its logfile.



Updating and Building the SDK

IMPORTANT: *Before running this command, be sure to set your `PATH` environment variable with the location of the PHP executable and the PEAR bin directory.*

Steps to Update the SDK

Follow these steps to update the SDK:

1. Change directories to `SDK_root`.
2. Run one of the following commands. If you have installed the PHP SDK with PEAR, these commands will be located in your PHP installation's `bin` directory.

TABLE B.1 *Commands for Updating the PHP SDK*

Microsoft Windows	<code>paypal-sdk-update.bat update</code>
UNIX	<code>paypal-sdk-update update</code>

