

Parallels[®] Plesk Sitebuilder

Developer's Guide

Integration API

Plesk Sitebuilder 4.5

Copyright Notice

ISBN: N/A

Parallels

660 SW 39th Street

Suite 205

Renton, Washington 98057

USA

Phone: +1 (425) 282 6400

Fax: +1 (425) 282 6444

© Copyright 1999-2008,

Parallels, Inc.

All rights reserved

Distribution of this work or derivative of this work in any form is prohibited unless prior written permission is obtained from the copyright holder.

Product and service names mentioned herein are the trademarks of their respective owners.

Contents

Preface	5
About This Guide	5
Who Should Read This Guide	5
Typographical+SDK Conventions	5
Feedback	6
About Sitebuilder	7
Sitebuilder Overview	8
Generic Architecture	9
Object Model	11
Integration API Overview	15
Implementation	15
SOAP Architecture	16
SOAP Message	16
SOAP over HTTP	17
Tutorials	18
Creating User Account	19
PHP	20
ASP	21
Creating Site	22
PHP	24
ASP	25
Creating Host	26
PHP	27
ASP	28
Creating Plan	29
PHP	30
ASP	31
Modifying Plan	33
PHP	34
ASP	35
Managing Login Handler	37
Client Software Samples	39
Scenarios Details	39
Try-Before-Buy Scenario	40
Sign up First Scenario	41
Installation	41
PHP	42
ASP	43
Appendix 1. PHP Classes Details	44

Utils_SoapClient	44
setCredentialsHeader	44

Appendix 2. How to Invoke Method **45**

Appendix 3. PHP Snippets **46**

PHP Classes	47
Creating User Account Sample	48
Creating Site Sample.....	50
Creating Host Sample	52
Creating Plan Sample	54
Modifying Plan Sample.....	56
Business Scenarios Code	58

Appendix 4. ASP Snippets **65**

Core Functions	66
Creating User Account Sample	77
Creating Site Sample.....	79
Creating Host Sample	81
Create Plan Sample	83
Modifying Plan Sample.....	86
Business Scenarios Code	89

Preface

In this section:

About This Guide.....	5
Who Should Read This Guide	5
Typographical+SDK Conventions.....	5
Feedback	6

About This Guide

This guide shows how Plesk Sitebuilder integration API can be used in third-party applications.

Who Should Read This Guide

This guide is devoted to developers who want to integrate Plesk Sitebuilder with third-party control panels or applications, or create their own web interfaces to Plesk Sitebuilder (different from Administrator Panel).

Typographical+SDK Conventions

Before you start using this guide, it is important to understand the documentation conventions used in it.

The following kinds of formatting in the text identify special information.

Formatting convention	Type of Information	Example
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.	Go to the System tab.
	Titles of chapters, sections, and subsections.	Read the Basic Administration chapter.

<i>Italics</i>	Used to emphasize the importance of a point, to introduce a term or to designate a command line placeholder, which is to be replaced with a real name or value.	The system supports the so called <i>wildcard character</i> search.
Monospace	The names of commands, files, and directories.	The license file is located in the <code>http://docs/common/licenses</code> directory.
Preformatted	On-screen computer output in your command-line sessions; source code in XML, C++, or other programming languages.	<pre># ls -al /files total 14470</pre>
Preformatted Bold	What you type, contrasted with on-screen computer output.	<pre># cd /root/rpms/php</pre>
CAPITALS	Names of keys on the keyboard.	SHIFT, CTRL, ALT
[to step N]	Identifies that a reader must proceed to the N-th step of a procedure.	[to step 5]
step N(M)	Identifies that a reader must choose the M-th branch of the N-th step.	[to step 5(2)]

Source code highlighting conventions are described in the following table.

Formatting convention	Type of Information	Example
Preformatted orange	Comments in code.	<code>//Creates a DEB package</code>
Preformatted blue	User-defined variables.	<code>\$MyVariable</code>
Preformatted red	String values.	<code>'Hello, world!'</code>
Preformatted dark-blue	Words or word combinations reserved by a programming language.	<code>procedure CreateAccount()</code>

Feedback

If you have found a mistake in this guide, or if you have suggestions or ideas on how to improve this guide, please send your feedback using the online form at <http://www.parallels.com/en/support/usersdoc/>. Please include in your report the guide's title, chapter and section titles, and the fragment of text in which you have found an error.

About Sitebuilder

This chapter provides information essential for understanding what Plesk Sitebuilder is and how it is organized.

In this chapter:

Sitebuilder Overview	8
Generic Architecture.....	9
Object Model	11

Sitebuilder Overview

Parallels Plesk Sitebuilder is a web application designed for easy creating and publishing web sites on the Internet. Sitebuilder is developed to run on Unix/Linux and Windows platforms. Plesk Sitebuilder for Linux/Unix or Windows can be installed either as a stand-alone system with the ability to publish to any server or group of servers via FTP, or as a Plesk-integrated add-on with the ability for Plesk domain owners to access Sitebuilder to design, build and publish sites within the Plesk interface.

On the administrative side, Sitebuilder enables managing resources such as hosts, user accounts, plans, and web sites. On the site-owner side, Sitebuilder allows creating and publishing web sites, and updating site content. In accordance with these tasks, Plesk Sitebuilder system is logically split into two parts: sites creating Wizard and Administration panel.

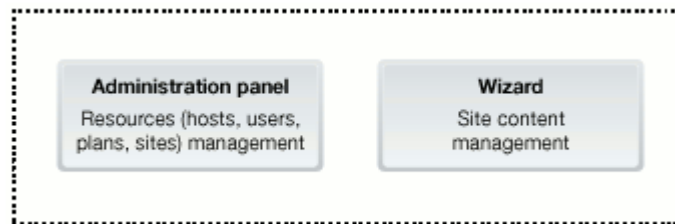


Figure 1: Sitebuilder parts

With the Administration panel, hosting providers can quickly and easily tailor Sitebuilder to their needs, for instance, by building a hierarchy of different plans, which define some subsets of Sitebuilder functionality available to different user levels, or by preparing page sets (site content templates), which help site owners with specific needs to start building sites.

The five-steps Wizard enables easily creating web sites and publishing them on the Internet. The Wizard is the main tool for site owners. Because of rich set of site design templates shipped in different color schemes, WYSIWYG editor, and a number of interactive modules, such as Blog, Image Gallery, Guestbook, eShop, the Wizard provides an effortless way for site owners to create, modify and update their websites without any technical skills or HTML knowledge.

The Sitebuilder administrator can make the Wizard also available for anonymous visitors, called guest users, who are, in fact, potential site owners. They can create trial sites in Sitebuilder but cannot publish them on the Internet. After a guest user creates a trial site, he or she is offered to purchase a hosting service to publish the site on the Internet.

Most of the Sitebuilder functionality is available through SOAP-based API, which makes possible automation of the administrative tasks and integration against external software.

Generic Architecture

Plesk Sitebuilder is a web application with a web server. The server exchanges data with user agents (Web Clients) via HTTP protocol, and with Web Services Clients via SOAP over HTTP. The latter type of communication is performed through the Service Facade component which provides the remote API capabilities.

Sitebuilder utilizes the common Model-View Controller design pattern, which separates it into the following architectural pieces:

- *Model*. Represents Sitebuilder domain model, the data on which the application operates. Manages the behavior and data of Sitebuilder, responds to requests for information about its state, and responds to instructions to change its state. The Model encapsulates the data access layer which provides communications with the database that stores the Sitebuilder data.
- *View*. Represents the Sitebuilder user interface. It is updated by the Controller's requests which use information provided by the Model. The component provides *Localization* and *Skining* capabilities, which allow creating multiple customizations of the Sitebuilder user interface.
- *Controller*. Interprets data sent by clients, and makes the appropriate requests to the Model and View so that they can be updated appropriately.

The following diagram depicts generic Sitebuilder architecture.

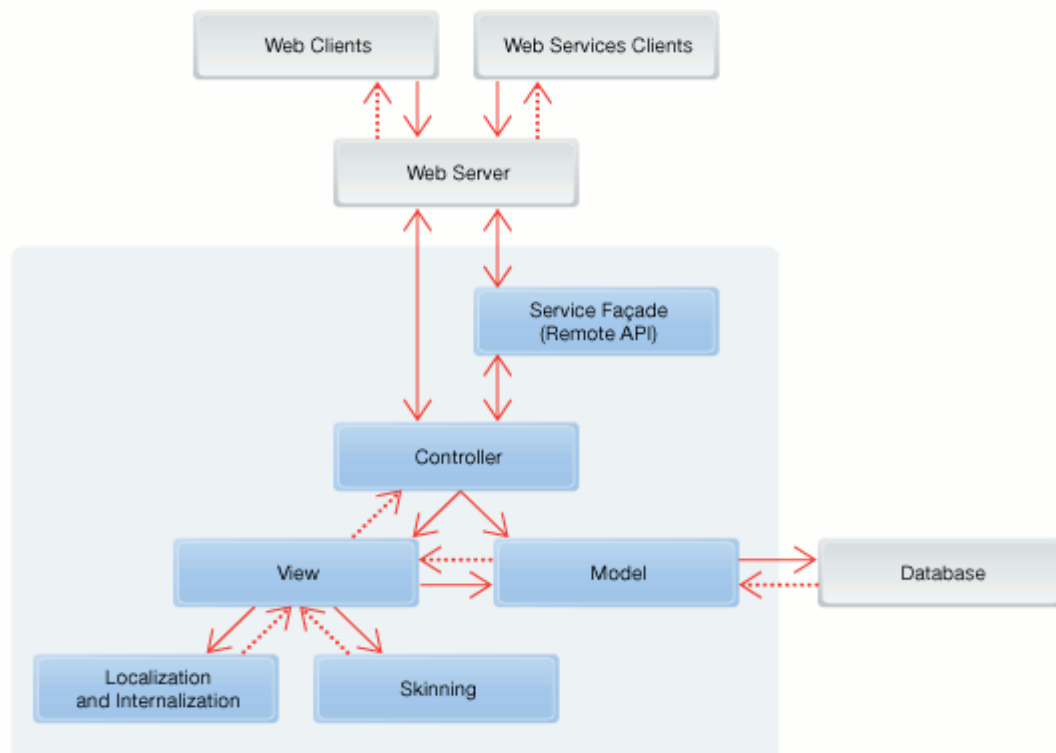


Figure 2: Generic architecture of Plesk Sitebuilder

Hence, Plesk Sitebuilder is designed to provide third-party developers with the following *integration* and *UI customization* capabilities:

- 1 Integrating Sitebuilder with third-party components via remote API using web services.

This guide is devoted to this very issue.

- 2 Customizing the ways the Sitebuilder user interface looks and speaks to a user, which implies, correspondingly, skinning and localizing Sitebuilder.
 - Skinning is customizing the Sitebuilder UI using skins (themes, in other terms) -- custom interface appearance styles.
 - Localization is customizing the Sitebuilder UI language using language packs.

Skinning and localizing Sitebuilder are considered in the corresponding guides within the current Sitebuilder SDK.

Object Model

This section is targeted to explain the Plesk Sitebuilder domain model, namely, to show which managed objects exist in Sitebuilder and how they are related to each other.





Plesk Sitebuilder domain model objects are as follows:

- *User*. A Sitebuilder user account representing one of the 3 hierarchical system roles (Admin, Reseller, Site Owner) which define in what way a user is authorized to interact with the system. Aside from this hierarchy stands a Guest User - any unauthenticated system actor allowed to create trial sites. Each User is assigned to a particular plan (see the definition below), and each registered user can have Sites.
- *Site*. A system entity representing a particular set of web pages as a whole, identified by a unique identifier, owned by a User, and presumably filled with User's content. Site can also be defined as an instance of a particular page set employing some design template.
- *Plan (Hosting Plan)*. A named set of limits, permissions and available resources to which a user is assigned. A user assigned to a plan can operate the objects provided by the plan, and within the limitations defined by the plan, too. One user can be assigned to only plan, while one plan can have many users assigned to it. Plans are organized into a hierarchy according to the hierarchy of users.

To understand the users and plans hierarchy, refer to the Figure 3.

- *License*. The system-level plan purchased from Parallels, the highest in the plans hierarchy.
- *Resource*. A Plan ingredient representing a system object available to a user according to a plan. Resources are *Hosts*, *Design Templates*, *Site Families*, *Page Sets* and *Modules*. To understand the user-determined relationship between plans and resources, refer to the Figure 4. *Host*. A physical or virtual operating environment which provides web hosting service for the users' sites. Host is identified by its name or IP address.
- *Page Set*. A named set of Web pages which represents web site structure and content template. It can be explained in other terms as site skeleton.
- *Design Template*. A template of web site design which could be applied to customize look and feel. Typically it has several color themes, header and menu variants.
- *Module*. An executable component which provides site dynamic content or extends web site functionality. Blog, forum, voting engine, and guestbook are examples of site modules that might be added to a site.
- *Site Family*. A resource which implies a predefined page set designed for the specific web site type, and exposes optional filter to the range of design templates suitable for this kind of web site.

The diagrams designed to illustrate the most important objects relationships in Plesk Sitebuilder use the following graphical conventions:

- 1...n means multiple relations or objects
- A  B means "A owns B"
- A  B means "A subordinates B", or "B obeys to A"
- A  B means "B inherits from A"
- A  B means "A utilizes B"

This diagram represents the hierarchical structure of users and plans in Sitebuilder.

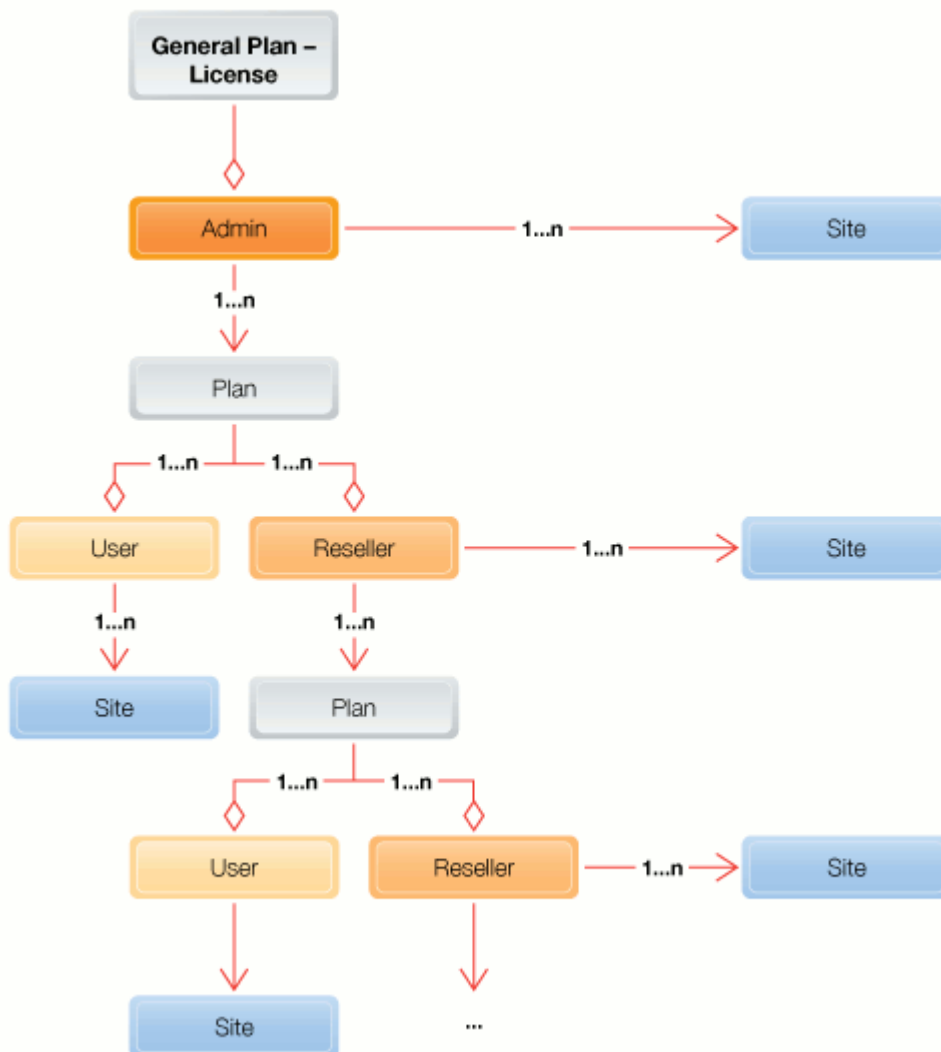


Figure 3: Hierarchy of Sitebuilder users and plans

Admin is a User automatically assigned to a General Plan - a License. Limited by the License, Admin creates his Sites and Plans, and assigns to these Plans another Users - Resellers and Site Owners. Both Resellers and Site Owners create their Sites according to the Plan to which each is assigned. Along with creating Sites, Reseller can also create his own Plans and Users, and assign these Users to his Plans. The Users that the Reseller creates are, again, Resellers and Site Owners who, again, create either their Sites and Plans (in case of a Reseller User), or their Sites only (in case of a Site Owner User). And the situation is repeated again and again, which results in creating multi-level hierarchy of Sitebuilder Plans and Users.

Plans are organized in such a hierarchy where, if moving top-down, the amount of Plan Resources is decreasing and increasing at the same time. The point is that Resellers do not create Plans by simply redistributing the parent Plan resources among them, some Resources (exactly, Hosts, Site Families, and Page Sets) can freely be created and used in addition to the inherited ones. This diagram illustrates how it happens:

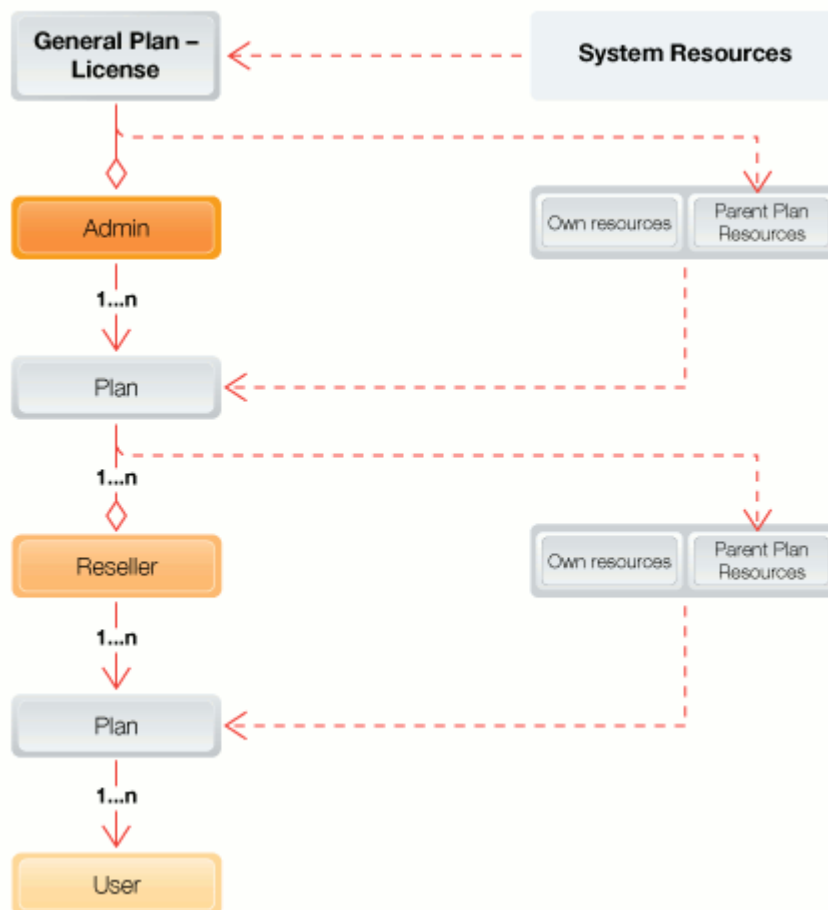


Figure 4: User-determined relationship between plans and resources

When creating a Plan, Admin operates with the pool of Resources composed of the Resources provided (or restricted) by the License (Parent Plan Resources on the diagram) and the Resources that he created himself (Own Resources). A Reseller assigned to this Plan, in turn, can create Plans from the Parent Plan Resources adding to them his Own Resources. And so on.

The following diagram explains how a Reseller creates a Plan adding his Own Resources to the ones inherited from his Parent Plan, and how a Site created upon this Plan utilizes the resources:

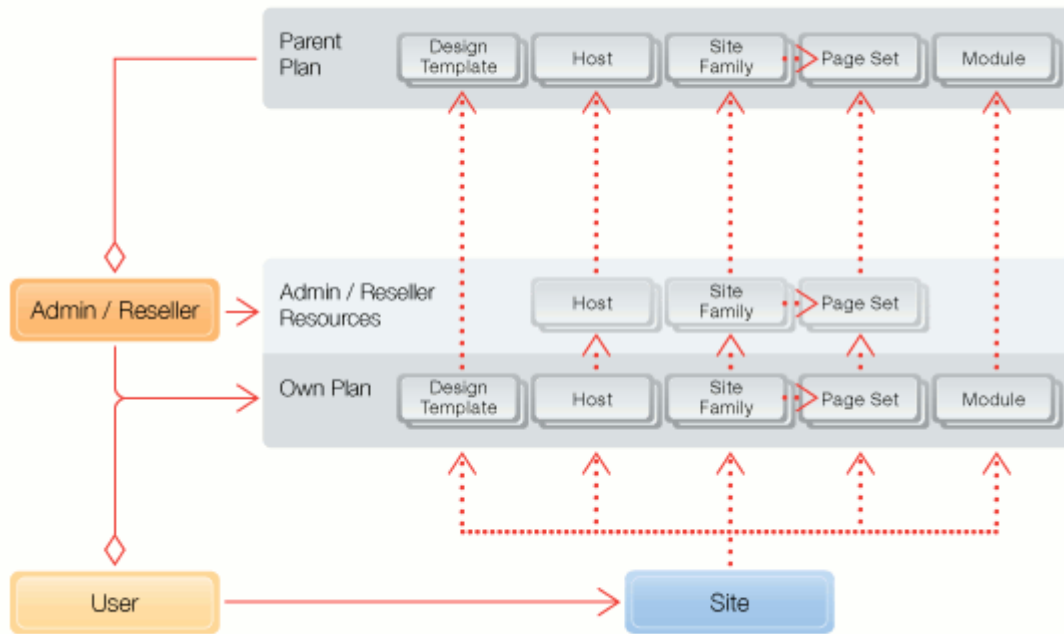


Figure 5: Reseller's plan created from Parent Plan Resources and Own Resources

Here, a Reseller (in fact, Admin is a Reseller, too, but of the highest level in the system) takes Design Templates, Hosts, Site Families, Page Sets and Modules provided by the Plan he is assigned to (Parent Plan Resources), adds to them Hosts, Site Families and Page Sets he created or somehow obtained (Admin/Reseller Resources), and creates his Own Plan from them. Then, a User assigned to this Plan can create a Site which will utilize all those resources the Reseller included to his Plan.

Integration API Overview

Plesk Sitebuilder integration API is a programming interface to various functions of Plesk Sitebuilder. Using this API, you can administrate Plesk Sitebuilder without graphical web interface (Administrator Panel), integrate Plesk Sitebuilder with third-party control panels or applications.

Access to Plesk Sitebuilder integration API is provided through the set of web services described by WSDL. Use Plesk Sitebuilder integration API on various operating systems via SOAP over HTTP.

For details on WSDL, refer to the www.w3.org/TR/wsdl/.

For details on SOAP, refer to the www.w3.org/TR/soap/.

In this chapter:

Implementation.....	15
---------------------	----

Implementation

Integration API uses the SOAP over HTTP protocol. That is why it is essential to say a couple of words about this protocol first. If you are familiar with the protocol, you can skip this subsection. The subsection does not describe details of SOAP. It provides you only with information essential for using integration API of Plesk Sitebuilder.

In this section:

SOAP Architecture	16
SOAP Message.....	16
SOAP over HTTP	17

SOAP Architecture

Simple Object Access Protocol (SOAP) provides access to remote objects. Examples of such objects are simple or Enterprise JavaBeans components and COM/COM+ objects, etc.

The following components of a typical SOAP communications architecture are:

- SOAP client
- SOAP server
- Actual service

The *client* is an application capable of generating and sending *requests* to a SOAP server over HTTP and retrieving responses from the server. Both request and response are SOAP messages. For details on SOAP messages, refer to the SOAP Message (on page 16) section.

The *server* is also an application that has capability to accept requests from clients, invoke the actual service, convert the response from the actual service to the SOAP-message structure and send it back to the client.

The *actual service* is functionality of Plesk Sitebuilder that can be used within the integration API. When the SOAP server invokes some method of actual service implementation, the method will perform its job and return the resulting information back to the SOAP server.

SOAP Message

SOAP message is XML-based data structured in a special way.

Each SOAP message is wrapped into the `soap:Envelope` tag and consists of *header* (`<soap:header>..</soap:header>`) and *body* (`<soap:body>..</soap:body>`) parts. Header is used by the integration API to identify a Plesk Sitebuilder user. Body contains attributes of a web service method.

A typical SOAP request can look as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <CredentialsSoapHeader
xmlns="http://swsoft.com/webservices/sb/4.5/SystemService">
      <Login>admin</Login>
      <Password>admin</Password>
    </CredentialsSoapHeader>
  </soap:Header>
  <soap:Body>
    <GetVersionApi
xmlns="http://swsoft.com/webservices/sb/4.5/SystemService" />
  </soap:Body>
</soap:Envelope>
```

A typical SOAP response can look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://swsoft.com/webservices/sb/4.5/SystemService">
  <SOAP-ENV:Body>
    <ns1:GetVersionApiResponse>
      <ns1:GetVersionApiResponse>
        <ns1:Major>4</ns1:Major>
        <ns1:Minor>0</ns1:Minor>
        <ns1:Build>2007062700</ns1:Build>
        <ns1:Revision>23623</ns1:Revision>
      </ns1:GetVersionApiResponse>
    </ns1:GetVersionApiResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP over HTTP

When you bind SOAP with HTTP or *operate SOAP over HTTP*, you actually add HTTP headers to the SOAP requests and responses. The request (listed above) with addition of HTTP headers looks as follows:

```
POST /ServiceFacade/4.5/SystemWebService.asmx HTTP/1.1
Host: sitebuilder4.demo.parallels.com
Connection: Keep-Alive
User-Agent: PHP SOAP 0.1
Content-Type: text/xml; charset=utf-8
SOAPAction:
"http://swsoft.com/webservices/sb/4.5/SystemService/GetVersionApi"
Content-Length: 480
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <CredentialsSoapHeader
xmlns="http://swsoft.com/webservices/sb/4.5/SystemService">
      <Login>admin</Login>
      <Password>admin</Password>
    </CredentialsSoapHeader>
  </soap:Header>
  <soap:Body>
    <GetVersionApi
xmlns="http://swsoft.com/webservices/sb/4.5/SystemService" />
  </soap:Body>
</soap:Envelope>
```

Tutorials

This chapter describes how to use the integration API for performing common Plesk Sitebuilder tasks. Each task is illustrated with code samples that can be used by third-party developers. For details on PHP, visit the <http://php.net>.

You can use any other languages that work with web services to perform the tasks. Each task comprises a sequence of integration API methods that are invoked with specific parameters. For details on how to invoke a method, refer to the How to Invoke Method (on page 45) section.

In this chapter:

Creating User Account	19
Creating Site	22
Creating Host	26
Creating Plan	29
Modifying Plan.....	33
Managing Login Handler	37

Creating User Account

This section describes how to create a Plesk Sitebuilder user account using the integration API.

In Plesk Sitebuilder, users are organized in a three-level hierarchy: *administrator*, *reseller*, and *site owner*.

For details on Plesk Sitebuilder users, refer to the **About Sitebuilder>Sitebuilder User Roles** of the Sitebuilder 4.5 Administrator's Guide.

Each Plesk Sitebuilder account is assigned to a specific plan. For details on the plans, refer to the **Serving Your Customers>Setting Up Service Plans** section of Sitebuilder 4.5 Administrator's Guide. For information on how to retrieve the plan identifier or list of available plans, refer to the **Web services>PlanWebService** section.

The accounts are created by methods of web service *AccountWebService*. *In Plesk Sitebuilder 4.5 for Linux/Unix, you cannot create administrator accounts via the integration API.*

Use the following methods to create Plesk Sitebuilder account:

- *CreateAccount*. Creates a user account and assigns it to a plan from the plan list of the user who invokes the method. For details on input and output parameters of the method, refer to the **Web services>AccountWebService>CreateAccount** section of Plesk Sitebuilder 4.5 Integration API Reference.
- *CreateAccountWithNewPlan*. First, it creates a user account and a copy of the plan assigned to the user who invokes the method. Then the plan is assigned to newly created user account.

Note: The plan is added to the plan list of the user who invoked the method.

For details on input and output parameters of the method, refer to the **Web services>AccountWebService>CreateAccountwithNewplan** section of Plesk Sitebuilder 4.5 Integration API Reference.

The section is illustrated with code samples written in PHP and ASP. To integrate Plesk Sitebuilder with your applications using PHP, refer to the **PHP** section (on page 20). Otherwise, refer to the **ASP** section (on page 21).

In this section:

PHP	20
ASP	21

PHP

You should include a set of PHP classes to invoke methods of integration API in your application. For the code of PHP classes, refer to the PHP Classes (on page 47) section.

Note: The classes should be declared before invoking integration API methods.

For details on the set of classes, refer to the PHP Classes Details (on page 44) section.

To create a Plesk Sitebuilder user account, the following steps should be taken:

1. Specify preferences common for all Plesk Sitebuilder web services.

The preferences are as follows:

- Web Services endpoint URL
- Login of a Plesk Sitebuilder user whose identity will be used to invoke methods of the integration API
- Password of the user

```
$serviceUrl = 'http://example.com/ServiceFacade/4.5/';  
$serviceLogin = 'serviceLogin';  
$servicePassword = 'servicePassword';
```

2. Create an instance of the SOAP client that invokes methods of web service *AccountWebService*.

```
$accountService = new  
Utils_SoapClient($serviceUrl.'/AccountWebService.asmx?WSDL', array());  
$accountService->setCredentialsHeader($serviceLogin,  
$servicePassword);
```

3. Create an instance of standard PHP class *stdClass* and specify the user account data. The data should contain user's credentials, his first name, last name, e-mail address, role, the ID of the plan, to which the account will be assigned and the boolean parameter defining if the user is able to change his password.

A sample of user account data can look as follows:

```
$struct = new stdClass();  
$struct->username = 'JohnDoe';  
$struct->password = '12345';  
$struct->firstName = 'John';  
$struct->lastName = 'Doe';  
$struct->email = 'my@example.com';  
$struct->role = 'Reseller';  
$struct->planId = '35b50410-da00-df8e-f6c4-1f0cb537a9a3';  
$struct->changePasswordAllowed = 'true';
```

4. Invoke one of the methods used for creating accounts. For details on the methods, refer to the [Creating User Account \(on page 19\)](#) section. The following example illustrates the use of method *CreateAccount*.

```
$result = $accountService->CreateAccount(new SoapVar($struct,
SOAP_ENC_OBJECT));
```

After the user account is created, you can retrieve preferences of the account. For instance, you can retrieve the ID of the plan to which the account is assigned. The PHP code of the operation looks as follows:

```
$accountId = $result->UserAccount->AccountId;
```

For full PHP code sample, refer to the [Creating User Account Sample \(on page 48\)](#) section.

ASP

This section contains examples of using the integration API that are written in ASP language.

To create a Plesk Sitebuilder user account, the following steps should be taken:

1. Specify common Plesk Sitebuilder preferences.

The preferences are as follows:

- Plesk Sitebuilder URL
- Login of a Plesk Sitebuilder user whose identity will be used to invoke methods of the integration API
- Password of the user

```
dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId
dim error
baseUrl="http://localhost:2006"
adminLogin="loginname"
adminPassword="password"
```

2. Specify account details. A sample of user account details can look as follows:

```
dim accountName, accountPassword, firstName, lastName, eMail, plan
accountName ="MyAccount"
accountPassword ="MyPassword"
firstName ="John"
lastName ="Doe"
eMail ="john@example.com"
plan ="35b50410-da00-df8e-f6c4-1f0cb537a9a3"
```

3. Add code of function *CreateAccount* from the list of core functions. These functions are written on ASP and are used to invoke Plesk Sitebuilder integration API methods.

For the list of all core functions, refer to the Core Functions (on page 66) section.

functions list.

4. Run the *CreateAccount* function.

```
AccountID = CreateAccount(accountName, accountPassword, firstName,
lastName, eMail, plan)
```

If the account is successfully created, the account ID value will be written to the `AccountID` variable.

For full ASP code sample, refer to the Creating User Account Sample (on page 77) section.

Creating Site

This section describes how to create a site in Plesk Sitebuilder using the integration API.

There are two types of sites in Plesk Sitebuilder: *trial* and *regular* sites. For details on site types, refer to the **Serving Your Customers>Managing Sites** section of Sitebuilder 4.5 Administrator's Guide.

To publish a site, you should specify publishing settings for the site. For details on the settings, refer to the **Serving Your Customers>Managing Sites>Registering Site in System** section of Sitebuilder 4.5 Administrator's Guide.

There are several publishing scenarios. They are as follows:

- *Standard*. This scenario uses the following schema: first, the host where a specific site will be published is created. Then the host is added to a plan. Then the plan is assigned to the user which site is going to be published. It is *highly recommended* to use this scenario.
- *FreeParallelsHosting*. In this scenario a user can publish sites to one of Parallels FTP servers. This scenario will be deprecated in next versions of Plesk Sitebuilder.
- *AnyFtp*. In this scenario a user can publish sites to the FTP server he defined in his publishing settings. This scenario will be deprecated in next versions of Plesk Sitebuilder.

The sites are created by methods of web service *SiteWebService*. The methods can be invoked by all users.

Use the following methods to create a Plesk Sitebuilder account:

- *CreateAnonymousSite*. Creates a trial site and adds it to list of sites of a specific reseller. If the reseller's ID is not specified, the site is owned by administrator. You can specify a site ID if needed, or the system will generate it automatically. Publishing settings are not specified in this method. For details on input and output parameters of the method, refer to the **Web services>SiteWebService>CreateAcnonyomousSite** section of Plesk Sitebuilder 4.5 Integration API Reference.
- *CreateSite*. Creates a site. Publishing settings are not specified in this method. The site can be trial or regular. If the site is regular, you should specify the site owner's ID. It can be retrieved by method *GetAccountByName* of service *AccountWebService*. For details on input and output parameters of the method, refer to the **Web services>SiteWebService>CreateSite** section of Plesk Sitebuilder 4.5 Integration API Reference.
- *CreateSite2*. Creates a site with specific publishing settings and owner. In this method you should also specify the ID of the host to which the site is published, if a user chose the *Publishing to location specified by provider* mode of publishing. The ID can be retrieved by method *GetHostByAddress* of service *HostWebService*. For details on input and output parameters of the method, refer to the **Web services>SiteWebService>CreateSite2** section of Plesk Sitebuilder 4.5 Integration API Reference.
- *CreateSiteWithHost*. First, the method creates the host with a specific ID add host to a user plan. Then it creates a site with standard FTP publishing scenario on the host. For details on input and output parameters of the method, refer to the **Web services>SiteWebService>CreateSiteWithHost** section of Plesk Sitebuilder 4.5 Integration API Reference.

The section is illustrated with code samples written in PHP and ASP. To integrate Plesk Sitebuilder with your applications using PHP, refer to the **PHP** section (on page 24). Otherwise, refer to the **ASP** section (on page 25).

In this section:

PHP	24
ASP	25

PHP

You should include a set of PHP classes to invoke methods of the integration API in your application. For the code of PHP classes, refer to the **PHP Classes** section (on page 47).

Note: The classes should be declared before invoking integration API methods.

For details on the set of classes, refer to the **PHP Classes Details** section (on page 44).

To create a Plesk Sitebuilder site, the following steps should be taken:

1. Specify preferences common for all Plesk Sitebuilder web services.

The preferences are as follows:

- Web Services endpoint URL
- Login of a Plesk Sitebuilder user whose identity will be used to invoke methods of the integration API
- Password of the user

```
$serviceUrl = 'http://example.com/ServiceFacade/4.5/';  
$serviceLogin = 'serviceLogin';  
$servicePassword = 'servicePassword';
```

2. Create an instance of the SOAP client that invokes methods of web service *SiteWebService*.

```
$siteService = new  
Utils_SoapClient($serviceUrl.'/SiteWebService.asmx?WSDL', array());  
$siteService->setCredentialsHeader($serviceLogin, $servicePassword);
```

3. Create an instance of standard PHP class *stdClass* and specify the site properties. The properties differ depending on a method that will be used for site creation. For details on the methods, refer to the Creating Site (on page 22) section.

A sample of the properties can look as follows:

```
$struct = new stdClass();  
$struct->siteType = 'Regular';  
$struct->siteAlias = 'siteName';  
$struct->ownerId = 'a12b50410-da11-vf8e-m6c4-3f0cb537a9a3';  
$struct->publishingSettings = new stdClass();  
$struct->publishingSettings->Id = '112332';  
$struct->publishingSettings->Mode = 'Ftp';  
$struct->publishingSettings->StandardLocation = new stdClass();  
$struct->publishingSettings->StandardLocation->Address =  
'ftp://example.com';  
$struct->publishingSettings->StandardLocation->UserName =  
'myUserName';  
$struct->publishingSettings->StandardLocation->Password =  
'publishFtpPassword';
```

```

$struct->publishingSettings->StandardLocation->WorkingDirectory =
'/www/';
$struct->publishingSettings->StandardLocation->WebSiteUrl =
'http://example.com/website';
$struct->publishingSettings->StandardLocation->IsAnonymous = 'false';
$struct->publishingSettings->StandardLocation->Veid = 0;
$struct->publishingSettings->StandardLocation->VerifyStatus =
'Unverified';
$struct->publishingSettings->StandardLocation->LastVerifyAttempt =
'0000-00-00 00:00:00';
$struct->publishingSettings->StandardLocation->FallbackIP =
'192.0.2.1';
$struct->publishingSettings->StandardLocation->HostId = '12';

```

4. Invoke one of the methods used for creating sites. For details on the methods, refer to the **Creating Site** section (on page 22). The following example illustrates the use of method *CreateSite2*.

```

$result = $siteService->CreateSite2(new SoapVar($struct,
SOAP_ENC_OBJECT));

```

After the user account is created, you can retrieve preferences of the site. For instance, you can retrieve the ID of the site. The PHP code of the operation looks as follows:

```

$siteId = $result->CreateSite2Result->Id;

```

For full PHP code sample, refer to the **Creating Site Sample** section (on page 50).

ASP

This section contains examples of using the integration API that are written in ASP language.

To create a Plesk Sitebuilder site, the following steps should be taken:

1. Specify common Plesk Sitebuilder preferences.

The preferences are as follows:

- Plesk Sitebuilder URL
- Login of a Plesk Sitebuilder user whose identity will be used to invoke methods of integration API
- Password of the user

```

dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId
dim error
baseUrl="http://localhost:2006"
adminLogin="admin"
adminPassword="admin"

```

2. Specify the ID of a site owner account and the site name. A sample of the properties can look as follows:

```
SiteOwnerID ="a12b50410-da11-vf8e-m6c4-3f0cb537a9a3"
SiteName ="My Site"
```

3. Add code of function *CreateSite* from the list of core functions. These functions are written on ASP and are used to invoke Plesk Sitebuilder integration API methods.

For the list of all core functions, refer to the **Core Functions** section (on page 66).

4. Run the *CreateSite* function.

```
NewSiteID = CreateSite(SiteOwnerID, SiteName)
```

If the site is successfully created, the site ID value will be written to the `NewSiteID` variable.

For full ASP code sample, refer to the **Creating User Account Sample** section (on page 79).

Creating Host

Host is a computer that is connected to a network and used for site publication. In Plesk Sitebuilder, each host is given a unique name that cannot be empty. This section describes how to add a host to Plesk Sitebuilder hosts pool using the integration API.

Before adding a host, make sure that the host meets all requirements specified in the **Setting Up and Maintaining Sitebuilder>Specifying Host Used for Publishing Sites>Requirements to Host** section of Sitebuilder 4.5 Administrator's Guide.

Hosts are managed by methods of web service *HostWebService*. The method that adds a host to the hosts pool is called *CreateHost*. This method can be invoked only by administrators or resellers. To add a host, you should specify name of the host, host IP address or DNS name and settings of the SMTP server from which notifications to the site users will be sent. For details on input and output parameters of the method, refer to the **Web services>HostWebService>CreateHost** section of Plesk Sitebuilder 4.5 Integration API Reference.

The section is illustrated with code samples written in PHP and ASP. To integrate Plesk Sitebuilder with your applications using PHP, refer to the **PHP** section (on page 27). Otherwise, refer to the **ASP** section (on page 28).

In this section:

PHP	27
ASP.....	28

PHP

You should include a set of PHP classes to invoke methods of the integration API in your application. For the code of PHP classes, refer to the **PHP Classes** section (on page 47).

Note: The classes should be declared before invoking integration API methods.

For details on the set of classes, refer to the **PHP Classes Details** section (on page 44).

To add a host to Plesk Sitebuilder hosts pool, the following steps should be taken:

1. Specify preferences common for all Plesk Sitebuilder web services.

The preferences are as follows:

- Web Services endpoint URL
- Login of a Plesk Sitebuilder user whose identity will be used to invoke methods of the integration API
- Password of the user

```
$serviceUrl = 'http://example.com/ServiceFacade/4.5/';
$serviceLogin = 'serviceLogin';
$servicePassword = 'servicePassword';
```

2. Create an instance of the SOAP client that invokes methods of web service *HostWebService*.

```
$hostService = new
Utils_SoapClient($serviceUrl.'/HostWebService.asmx?WSDL', array());
$hostService->setCredentialsHeader($serviceLogin, $servicePassword);
```

3. Create an instance of standard PHP class *stdClass* and specify the host properties.

A sample of host properties can look as follows:

```
$struct = new stdClass();
$struct->hostName = 'myNewHost';
$struct->hostAddress = 'example.com';
$struct->SMTPPort = 0;
```

4. Invoke method *CreateHost*.

```
$result = $hostService->CreateHost(new SoapVar($struct,
SOAP_ENC_OBJECT));
```

After the host is created, you can retrieve preferences of the host. For instance, you can retrieve the host ID. The PHP code of the operation looks as follows:

```
$hostId = $result->CreateHostResult->Id;
```

For full PHP code sample, refer to the [Creating Host Sample](#) section (on page 52).

ASP

This section contains examples of using the integration API that are written in ASP language.

To add a host to Plesk Sitebuilder hosts pool, the following steps should be taken:

1. Specify common Sitebuilder preferences.

The preferences are as follows:

- Sitebuilder URL
- Login of a Sitebuilder user whose identity will be used to invoke methods of the integration API
- Password of the user

```
dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId
dim error
baseUrl="http://localhost:2006"
adminLogin="admin"
adminPassword="admin"
```

2. Specify the ID of a site owner account and the site name. A sample of the properties can look as follows:

```
HostName ="My new host"
HostAddress ="192.0.2.12"
```

3. Add code of function *CreateHost* from the list of core functions. These functions are written on ASP and are used to invoke Plesk Sitebuilder integration API methods.

For the list of all core functions, refer to the **Core Functions** section (on page 66).

4. Run the *CreateHost* function.

```
NewHostID = CreateHost(HostName, HostAddress)
```

If the host is successfully added, the host ID value will be written to the `NewHostID` variable.

For full ASP code sample, refer to the **Creating Host Sample** section (on page 81).

Creating Plan

This section describes how to create a plan in Plesk Sitebuilder using the integration API.

Each Plesk Sitebuilder account is assigned to a specific plan. For details on the plans, refer to the **Serving Your Customers>Setting Up Service Plans** section of Sitebuilder 4.5 Administrator's Guide.

The plans are created by methods *CreatePlan* of web service *PlanWebService* and *CreateAccountWithNewPlan* of web service *AccountWebService*. For details on method *CreateAccountWithNewPlan*, refer to the **Creating User Account** section (on page 19). The methods can be invoked by administrators and resellers. For details on input and output parameters of method *CreatePlan*, refer to the **Web services>PlanWebService>CreatePlan** section of Plesk Sitebuilder 4.5 Integration API Reference.

The section is illustrated with code samples written in PHP and ASP. To integrate Plesk Sitebuilder with your applications using PHP, refer to the **PHP** section (on page 30). Otherwise, refer to the **ASP** section (on page 31).

In this section:

PHP	30
ASP	31

PHP

You should include a set of PHP classes to invoke methods of the integration API in your application. For the code of PHP classes, refer to the **PHP Classes** section (on page 47).

Note: The classes should be declared before invoking integration API methods.

For details on the set of classes, refer to the **PHP Classes Details** section (on page 44).

To create a plan, the following steps should be taken:

1. Specify preferences common for all Plesk Sitebuilder web services.

The preferences are as follows:

- Web Services endpoint URL
- Login of a Plesk Sitebuilder user whose identity will be used to invoke methods of the integration API
- Password of the user

```
$serviceUrl = 'http://example.com/ServiceFacade/4.5/';  
$serviceLogin = 'serviceLogin';  
$servicePassword = 'servicePassword';
```

2. Create an instance of the SOAP client that invokes methods of web service *PlanWebService*.

```
$planService = new  
Utils_SoapClient($serviceUrl.'/PlanWebService.asmx?WSDL', array());  
$planService->setCredentialsHeader($serviceLogin, $servicePassword);
```

3. Create an instance of standard PHP class *stdClass* and specify the plan properties. The data contain plan name, description, restrictions on the content of the sites assigned to the plan and permissions of the users assigned to the plan.

A sample of plan properties can look as follows:

```
$struct = new stdClass();  
$struct->name = 'myNewPlan';  
$struct->description = 'This is a sample plan.';  
$struct->maxPagesNumber = 100;  
$struct->maxPagesRootLevel = 10;  
$struct->maxPagesLevel = 3;  
$struct->maxSitesNumber = 5;  
$struct->maxAccountsNumber = 15;  
$struct->maxHostsNumber = 15;  
$struct->isPersonal = true;  
$struct->isAnonymous = true;  
$struct->trialLifeTime = 2;  
$struct->trialLifeType = 'Days';  
$struct->templatesIds = new stdClass();
```

```

$struct->templatesIds->string = '15';
$struct->templatesIds->string = '19';
$struct->pagesetsIds = new stdClass();
$struct->pagesetsIds->string = '1';
$struct->pagesetsIds->string = '2';
$struct->modulesIds = new stdClass();
$struct->modulesIds->string = '2';
$struct->hostsIds = new stdClass();
$struct->hostsIds->string = '2';
$struct->families = new stdClass();
$struct->families->string = '1';
$struct->families->string = '3';
$struct->defaultFamily = '3';
$struct->isPublishingSettingsEditable = false;
$struct->isFtpPublishAvailable = true;
$struct->isXcopyPublishAvailable = false;
$struct->isVpsPublishAvailable = false;
$struct->isAdditionalSiteContentAllowed = false;
$struct->AdditionalSiteContent = '';
$struct->isUserManagementAllowed = true;
$struct->isSiteManagementAllowed = true;

```

4. Invoke method *CreatePlan*.

```

$result = $planService->CreatePlan(new SoapVar($struct,
SOAP_ENC_OBJECT));

```

After the plan is created, you can retrieve preferences of the plan. For instance, you can retrieve the plan ID. The PHP code of the operation looks as follows:

```

$hostId = $result->CreatePlanResult->planId;

```

For full PHP code sample, refer to the **Creating Plan Sample** section (on page 54).

ASP

This section contains examples of using the integration API that are written in ASP language.

To create a plan, the following steps should be taken:

1. Specify common Plesk Sitebuilder preferences.

The preferences are as follows:

- Plesk Sitebuilder URL
- Login of a Plesk Sitebuilder user whose identity will be used to invoke methods of the integration API
- Password of the user

```

dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId
dim error
baseUrl="http://localhost:2006"

```

```
adminLogin="admin"  
adminPassword="admin"
```

2. Specify the plan properties comprising plan name, description, restrictions on the content of the sites assigned to the plan and permissions of the users assigned to the plan. A sample of the properties can look as follows:

```
PlanName = "My new host"  
PlanDescription = "192.0.2.12"  
PMpagesLimit = "100"  
PMpagesrootLevel = "10"  
PMpagesLevel = "3"  
PMsitesNumber = "5"  
PMaccountsNumber = "15"  
PMhostsNumber = "15"  
PisPersonal = "true"  
PisAnonymous = "false"  
PTrialLifeTime = "2"  
PTrialLifeType = "Days"  
PTemplatesID = "15"  
PPageSetsID = "1"  
PModulesID = "1"  
PHostsID = "2"  
PFamiliesID = "1"  
PDefaultFamilyID = "3"  
PisPublishSettingsEditable = "false"  
PisFTPPublishAvailable = "true"  
PisXcopyPublishAvailable = "false"  
PisVPSPublishAvailable = "false"  
PisAdditionalSiteContentAllowed = "false"  
PAdditionalSiteContent = ""  
PisUserManagementAllowed = "true"  
PisSiteManagementAllowed = "true"
```

3. Add code of function *CreatePlan* from the list of core functions. These functions are written on ASP and are used to invoke Plesk Sitebuilder integration API methods.

For the list of all core functions, refer to the **Core Functions** section (on page 66).

4. Run the *CreatePlan* function.

```
NewPlanID = CreatePlan(PlanName, PlanDescription, PMpagesLimit,  
PMpagesrootLevel, PMpagesLevel,  
PMsitesNumber, PMaccountsNumber, PMhostsNumber, PisPersonal,  
PisAnonymous, PTrialLifeTime,  
PTrialLifeType, PTemplatesID, PPageSetsID, PModulesID, PHostsID,  
PFamiliesID, PDefaultFamilyID,  
PisPublishSettingsEditable, PisFTPPublishAvailable,  
PisXcopyPublishAvailable, PisVPSPublishAvailable,  
PisAdditionalSiteContentAllowed,  
PAdditionalSiteContent, PisUserManagementAllowed,  
PisSiteManagementAllowed)
```

If the plan is successfully created, the plan ID value will be written to the `NewPlanID` variable.

For full ASP code sample, refer to the **Creating Host Sample** section (on page 83).

Modifying Plan

This section describes how to modify a plan in Plesk Sitebuilder using the integration API.

Each Plesk Sitebuilder account is assigned to a specific plan. For details on the plans, refer to the **Serving Your Customers>Setting Up Service Plans** section of Sitebuilder 4.5 Administrator's Guide.

The plans are modified by method *UpdatePlan* of web service *PlanWebService*. The method can be invoked by administrators and resellers. For details on input and output parameters of method *UpdatePlan*, refer to the **Web services>PlanWebService>UpdatePlan** section of Plesk Sitebuilder 4.5 Integration API Reference.

The section is illustrated with code samples written in PHP and ASP. To integrate Plesk Sitebuilder with your applications using PHP, refer to the **PHP** section (on page 34). Otherwise, refer to the **ASP** section (on page 35).

In this section:

PHP	34
ASP	35

PHP

You should include a set of PHP classes to invoke methods of the integration API in your application. For the code of PHP classes, refer to the **PHP Classes** section (on page 47).

Note: The classes should be declared before invoking integration API methods.

For details on the set of classes, refer to the **PHP Classes Details** section (on page 44).

To create a plan, the following steps should be taken:

1. Specify preferences common for all Plesk Sitebuilder web services.

The preferences are as follows:

- Web Services endpoint URL
- Login of a Plesk Sitebuilder user whose identity will be used to invoke methods of the integration API
- Password of the user

```
$serviceUrl = 'http://example.com/ServiceFacade/4.5/';  
$serviceLogin = 'serviceLogin';  
$servicePassword = 'servicePassword';
```

2. Create an instance of the SOAP client that invokes methods of web service *PlanWebService*.

```
$planService = new  
Utils_SoapClient($serviceUrl.'/PlanWebService.asmx?WSDL', array());  
$planService->setCredentialsHeader($serviceLogin, $servicePassword);
```

3. Create an instance of standard PHP class *stdClass* and specify new plan properties. The data contain plan name, description, restrictions on the content of the sites assigned to the plan and permissions of the users assigned to the plan.

A sample of plan properties can look as follows:

```
$struct = new stdClass();  
$struct->planId = '35b50410-da00-df8e-f6c4-1f0cb537a9a3';  
$struct->name = 'myNewNewPlan';  
$struct->description = 'This is a modified plan.';  
$struct->maxPagesNumber = 100;  
$struct->maxPagesRootLevel = 10;  
$struct->maxPagesLevel = 3;  
$struct->maxSitesNumber = 5;  
$struct->maxAccountsNumber = 15;  
$struct->maxHostsNumber = 15;  
$struct->trialLifeTime = 2;  
$struct->trialLifeType = 'Days';  
$struct->isPublishingSettingsEditable = false;  
$struct->isFtpPublishAvailable = true;
```

```

$struct->isXcopyPublishAvailable = false;
$struct->isVpsPublishAvailable = false;
$struct->isAdditionalSiteContentAllowed = false;
$struct->isUserManagementAllowed = true;
$struct->isSiteManagementAllowed = true;

```

4. Invoke method *UpdatePlan*.

```

$result = $planService->UpdatePlan(new SoapVar($struct,
SOAP_ENC_OBJECT));

```

For full PHP code sample, refer to the Modifying Plan Sample (on page 56) section.

ASP

This section contains examples of using the integration API that are written in ASP language.

To modify a plan, the following steps should be taken:

1. Specify common Plesk Sitebuilder preferences.

The preferences are as follows:

- Plesk Sitebuilder URL
- Login of a Plesk Sitebuilder user whose identity will be used to invoke methods of the integration API
- Password of the user

```

dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId
dim error
baseUrl="http://localhost:2006"
adminLogin="admin"
adminPassword="admin"

```

2. Specify the plan ID and the plan properties that should be changed. A sample of the properties can look as follows:

```

PlanId = "35b50410-da00-df8e-f6c4-1f0cb537a9a3"
PlanName = "My new host"
PlanDescription = "192.0.2.12"
PMpagesLimit = "100"
PMpagesrootLevel = "10"
PMpagesLevel = "3"
PMsitesNumber = "5"
PMaccountsNumber = "15"
PMhostsNumber = "15"
PTrialLifeTime = "2"

```

```
PTrialLifeType = "Days"  
PTemplatesID = "15"  
PPageSetsID = "1"  
PModulesID = "1"  
PHostsID = "2"  
PFamiliesID = "1"  
PDefaultFamilyID = "3"  
PIsPublishSettingsEditable = "false"  
PIsFTPPublishAvailable = "true"  
PIsXcopyPublishAvailable = "false"  
PIsVPSPublishAvailable = "false"  
PIsAdditionalSiteContentAllowed = "false"  
PAdditionalSiteContent = ""  
PIsUserManagementAllowed = "true"  
PIsSiteManagementAllowed = "true"
```

3. Add code of function *UpdatePlan* from the list of core functions. These functions are written on ASP and are used to invoke Plesk Sitebuilder integration API methods.

For the list of all core functions, refer to the **Core Functions** section (on page 66).

4. Run the *UpdatePlan* function.

```
UpdatePlan(PlanId, PlanName, PlanDescription, PMpagesLimit,  
PMpagesrootLevel, PMpagesLevel, PmsitesNumber, PMaccountsNumber,  
PMhostsNumber, PTrialLifeTime, PTrialLifeType,  
PTemplatesID, PPageSetsID, PModulesID, PHostsID, PFamiliesID,  
PDefaultFamilyID, PIsPublishSettingsEditable, PIsFTPPublishAvailable,  
PIsXcopyPublishAvailable, PIsVPSPublishAvailable,  
PIsAdditionalSiteContentAllowed, PAdditionalSiteContent,  
PIsUserManagementAllowed, PIsSiteManagementAllowed)
```

For full ASP code sample, refer to the **Creating Host Sample** section (on page 86).

Managing Login Handler

Login handler is a utility that redirects a user who already logged in to the Plesk Sitebuilder Administrator Panel or Wizard.

For this, send a GET or POST query to the handler address:

- `http://<sb-example-host.com>/ExternalLogin.ashx` (Plesk Sitebuilder for Windows) or
- `http://<sb-example-host.com>/external_login.php` (Plesk Sitebuilder for Linux/Unix)

Here `http://<sb-example-host.com>/` means your Plesk Sitebuilder root URL.

Query parameters

- *Login*. User login ID. Required.
- *Password*. User password. Required.
- *SiteID*. ID of the site to open in the Wizard after successful login. Optional.
- *ShowAdmin*. If specified (and SiteID is not specified), redirects to the Administrator Panel on successful login. Optional
- *ReturnUrl*. URL to redirect to in case of successful login. Used if neither SiteID nor ShowAdmin are present in the query. Optional.
- *FailUrl*. URL to redirect to in case of login failure. Optional, but highly recommended to specify the parameter.
- *Skin*. ID of skin to select in the new session. Can be one of the following:
 - AquaCompact
 - LonghornAero
 - LonghornBlack
 - PleskBlue
 - PleskReflections
 - PleskSea
 - PleskViolet
 - WinXPBlue
 - WinXPOLivegreen
 - WinXPReloadedCompact
 - WinXPSilver
 - LonghornYellow
 - PleskClassicSilver
 - PleskNature
 - PleskSilver
- *Language*. Session GUI language by standard culture code, for example "en-US", or "ru-RU". Optional.

Request examples

If dealing with Plesk Sitebuilder for Linux/Unix, use the URLs in the following formats:

To log in to the Administrator Panel as administrator:

- `http://<sb-example-host.com>/external_login.php?Login=admin&Password=admin`
- To log in to the Wizard as site owner:
`http://<sb-example-host.com>/external_login.php?Login=siteowner&Password=siteowner`
- To log in to the Administrator Panel as site owner:
`http://<sb-example-host.com>/external_login.php?Login=siteowner&Password=siteowner&ShowAdmin=true`
- To log in to the Administrator Panel with a specific language and skin:
`http://<sb-example-host.com>/external_login.php?Login=admin&Password=admin&Skin=AquaCompact&Language=de-DE`
- To log in as site owner directly to the page of editing a particular site (specified by its ID):
`http://<sb-example-host.com>/external_login.php?Login=siteowner&Password=siteowner&SiteID=b2e11f743e10475dcaa60f81b271caaa`

If dealing with Plesk Sitebuilder for Windows, use the URLs in the following formats:

To log in to the Administrator Panel as administrator: `http://<sb-example-host.com>/ExternalLogin.ashx?Login=admin&Password=admin`

- To log in to the Wizard as site owner:
`http://<sb-example-host.com>/ExternalLogin.ashx?Login=siteowner&Password=siteowner`
- To log in to the Administrator Panel as site owner:
`http://<sb-example-host.com>/ExternalLogin.ashx?Login=siteowner&Password=siteowner&ShowAdmin=true`
- To log in to the Administrator Panel with a specific language and skin:
`http://<sb-example-host.com>/ExternalLogin.ashx?Login=admin&Password=admin&Skin=AquaCompact&Language=de-DE`
- To log in as site owner directly to the page of editing a particular site (specified by its ID):
`http://<sb-example-host.com>/ExternalLogin.ashx?Login=siteowner&Password=siteowner&SiteID=b2e11f743e10475dcaa60f81b271caaa`

Note: It is recommended to avoid the POST request in order to save the sensitive info in browser navigation history.

Client Software Samples

This chapter contains code of two standard business scenarios: Try-Before-Buy and Sign up First.

The **Installation** (on page 41) sub-section tells where to take the code samples from, and how to make the code work with Plesk Sitebuilder so that they could implement the scenarios.

The **Scenarios Details** (on page 39) section explains scenarios business logic.

In this chapter:

Scenarios Details	39
Installation.....	41

Scenarios Details

The Try-Before-Buy scenario is used when an unregistered (in Plesk Sitebuilder) user created a site using Plesk Sitebuilder Wizard and wants to take ownership of the site. The Sign up First scenario is used when a user registers a Plesk Sitebuilder account first, and then creates a site.

In this section:

Try-Before-Buy Scenario.....	40
Sign up First Scenario.....	41

Try-Before-Buy Scenario

- 1 Anonymous user opens Plesk Sitebuilder wizard, creates a trial site, and opens the Publish step of the Wizard.

At the Publish step, the user clicks the "register" link and fills in the registration form containing the following fields:

- Login name
- First name/Last name
- Email address
- Password
- Desired plan
- Submit button

- 2 User submits the form.

- 3 System validates the submitted data, creates a Plesk Sitebuilder user, assigns the plan to the user, changes status of the site created by the user from trial to regular, gives the user ownership of the site, and creates links offering the user to redirect to Plesk Sitebuilder Administrator Panel or to the Wizard page with credentials of the created Plesk Sitebuilder account.

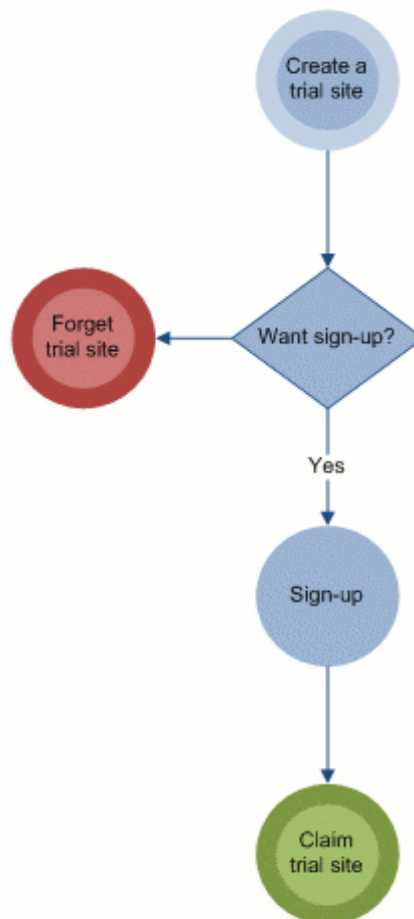


Figure 6: Try-Before-Buy scenario

Sign up First Scenario

- 1 Anonymous user opens and fills the registration form containing the following fields:
 - Login name
 - First name/Last name
 - Email address
 - Password
 - Desired plan (Simple without modules, Advanced with modules)
- 2 The user submits the form.
- 3 System checks that the data entered is valid, creates a Plesk Sitebuilder user, assigns plan to the user, creates a regular site, gives the user ownership of this site and creates links offering the user to redirect to Plesk Sitebuilder Administrator Panel or to the Wizard page with credentials of the created Plesk Sitebuilder account.



Figure 7: Sign Up First Scenario

Installation

This section explains how to run Try-Before-Buy and Sign up First scenarios.

In this section:

PHP	42
ASP.....	43

PHP

➤ **To install the integration script, perform the following steps:**

1. Select a working directory for the integration script, and place the script (`TestSbIntegration.php`) to it.

You have here two options:

- If you have the Plesk Sitebuilder 4.5 SDK installed, take the `TestSbIntegration.php` file from the `%SBSDK%\Samples\Integration\PHP` directory. (`%SBSDK%` denotes the directory of Plesk Sitebuilder SDK installation, by default it is `C:\Program Files\Parallels\Plesk Sitebuilder 4.5\SDK\.`)
- Copy the application source code and save it as the `TestSbIntegration.php` to the working directory.

For full code of the application, refer to the **Business Scenarios Code** section (on page 58).

2. Create the required plans in the Plesk Sitebuilder Administrator Panel.
3. Open the `TestSbIntegration.php` in a text editor and edit the following lines (392-395):

```
$locationUrl = 'http://<sitebuilder-host>';  
$servicesDirectory = '/ServiceFacade/4.5/';  
$adminLogin = 'myLogin';  
$adminPassword = 'myPassowrd';
```

4. In Plesk Sitebuilder Administrator Panel, open **Server -> Trial Site Settings -> Publish step message** and edit the message with the following text, specifying the correct path to the `TestSbIntegration.php` file:

```
<p align=center>Your temporary site will be available for a  
%lifetime at %sitepreviewlink</p><hr><p align=center>If you  
want to save it, please <a  
href"http://<yourserver>/<yourpath>/TestSbIntegration.php?siteId=%site  
id">register</a></p>
```

ASP

➤ **To install the integration script, perform the following steps:**

1. Complete Plesk Sitebuilder setup and register the administrator account.
2. Configure IIS as follows:
 1. Create mapping between an IIS virtual folder and the working directory where the script will be deployed. For this, create a virtual folder in the IIS and then create a matching working folder, or map the IIS virtual folder to the already existing physical folder where the script must be deployed.
 2. Select the **Run scripts (such as ASP)** checkbox at the **Virtual Directory Access** permissions page.
3. Place the script to the working directory.

You have here two options:

- If you have the Plesk Sitebuilder 4.5 SDK installed, take the `TestSbIntegration.asp` file from the `%SBSDK%\Samples\Integration\ASP` directory. (`%SBSDK%` denotes the directory of Plesk Sitebuilder SDK installation, by default it is `C:\Program Files\Parallels\Plesk Sitebuilder 4.5\SDK\.`)
- Copy the application source code and save it as the `TestSbIntegration.php` to the working directory.

For full code of the application, refer to the **Business Scenarios Code** section (on page 89).

4. Create the required plans in the Plesk Sitebuilder Administrator Panel.
5. Open `TestSbIntegration.asp` in a text editor and edit the following lines (11-13):

```
baseUrl="http://<sitebuilder-host>" 'base URL to Sitebuilder without end slash
adminLogin="admin" 'administrator login, it is required to authenticate API calls
adminPassword="admin" 'administrator password, it is required to authenticate API calls
```

6. In Plesk Sitebuilder Administrator Panel, open **Server -> Trial Site Settings -> Publish step message** and edit the message with the following text, specifying the correct path to `TestIntegration.asp`:

```
<p align=center>Your temporary site will be available for a
sb:lifetime at sb:sitepreviewlink</p><hr><p align=center>If you want
to save it, please <a
href="http://<yourserver>/<yourpath>/TestSbIntegration.asp?siteId=sb:s
iteid">register</a></p>
```

Appendix 1. PHP Classes Details

To invoke methods of the integration API using PHP, you should include the set of PHP classes in the code of your applications. The set consists of the following classes:

- Uutils_SoapClient (on page 44)

In this chapter:

Uutils_SoapClient	44
-------------------------	----

Uutils_SoapClient

The class extends standard PHP class SoapClient. It is used to create a SOAP client. For details on SOAP clients, refer to the SOAP Architecture (on page 16) section. The class includes the following functions:

setCredentialsHeader (on page 44)

In this section:

setCredentialsHeader.....	44
---------------------------	----

setCredentialsHeader

This function forms the header of a SOAP message. For details on SOAP headers, refer to the SOAP Message (on page 16) section.

Arguments

login

Data type: *string*. The Plesk Sitebuilder user's login.

password

Data type: *string*. The Plesk Sitebuilder user's password.

Return value

None.

Appendix 2. How to Invoke Method

To invoke a specified method of a web service, you need to do the following:

- 1 Find a web service you want to use. List of all web services is found in the **Web Services** section of the **Integration API Reference**. Each web service is represented by XSD schema. Location of the schema is specified by **WSDL location** parameter.
- 2 Find a method you want to invoke.
- 3 Construct a request packet using XSD representation or a sample request packet displayed for each method. In a sample packet's header, you need to substitute the **Host** and **Content-Length** values with your own. In the sample packet's body, each attribute value is substituted with a type of the attribute . You need to change them into values you need.

Send the request packet and get a response packet using a SOAP client. To look at samples of SOAP client applications, refer to the Client Software Samples (on page 39) section.

Appendix 3. PHP Snippets

In this chapter:

PHP Classes	47
Creating User Account Sample	48
Creating Site Sample	50
Creating Host Sample	52
Creating Plan Sample	54
Modifying Plan Sample.....	56
Business Scenarios Code	58

PHP Classes

```
/**
 * Extended SOAP client for simplified setup of credentials header
 *
 * @package Utils
 * @copyright (c) 2004-2008, Parallels
 */
class Utils_SoapClient extends SoapClient {
    protected $_targetNamespace;

    /**
     * Constructor
     *
     * @param mixed $wsdl
     * @param array $options
     */
    public function __construct($wsdl, $options) {
        parent::__construct($wsdl, $options);
        // detect target namespace $xml =
        simplexml_load_file($wsdl);
        $this->_targetNamespace = (string)
        $xml['targetNamespace'];
    }

    /**
     * Set credentials header
     *
     * @param string $login
     * @param string $password
     */
    public function setCredentialsHeader($login, $password) {
        $header = new SoapHeader($this->_targetNamespace,
            'CredentialsSoapHeader',
            new SoapVar(
                array(
                    'Login' => $login,
                    'Password' => $password,
                ),
                SOAP_ENC_OBJECT,
                'CredentialsSoapHeader',
                $this->_targetNamespace
            )
        );

        $this->__setSoapHeaders(array($header));
    }
}
```

Creating User Account Sample

```

<?php
/**
 * Extended SOAP client for simplified setup of credentials header
 *
 * @package Utils
 * @copyright (c) 2004-2008, Parallels
 */
class Utils_SoapClient extends SoapClient {
    protected $_targetNamespace;

    /**
     * Constructor
     *
     * @param mixed $wsdl
     * @param array $options
     */
    public function __construct($wsdl, $options) {
        parent::__construct($wsdl, $options);
        // detect target namespace $xml =
simplexml_load_file($wsdl);
        $this->_targetNamespace = (string)
$xml['targetNamespace'];
    }

    /**
     * Set credentials header
     *
     * @param string $login
     * @param string $password
     */
    public function setCredentialsHeader($login, $password) {
        $header = new SoapHeader($this->_targetNamespace,
            'CredentialsSoapHeader',
            new SoapVar(
                array(
                    'Login' => $login,
                    'Password' => $password,
                ),
                SOAP_ENC_OBJECT,
                'CredentialsSoapHeader',
                $this->_targetNamespace
            )
        );

        $this->__setSoapHeaders(array($header));
    }
}

$serviceUrl = 'http://example.com/ServiceFacade/4.5/';
$serviceLogin = 'serviceLogin';
$servicePassword = 'servicePassword';
$accountService = new
Utils_SoapClient($serviceUrl.'/AccountWebService.asmx?WSDL', array());
$accountService->setCredentialsHeader($serviceLogin,
$servicePassword);

```

```
$struct = new stdClass();
$struct->username = 'myUserName';
$struct->password = 'myPassword';
$struct->firstName = 'myFirstName';
$struct->lastName = 'myLastName';
$struct->email = 'my@example.com';
$struct->role = 'Reseller';
$struct->planId = 1;

$result = $accountService->CreateAccount(new SoapVar($struct,
SOAP_ENC_OBJECT));
$accountId = $result->UserAccount->AccountId;
?>
```

Creating Site Sample

```

<?php
/**
 * Extended SOAP client for simplified setup of credentials header
 *
 * @package Utils
 * @copyright (c) 2004-2008, Parallels
 */
class Utils_SoapClient extends SoapClient {
    protected $_targetNamespace;

    /**
     * Constructor
     *
     * @param mixed $wsdl
     * @param array $options
     */
    public function __construct($wsdl, $options) {
        parent::__construct($wsdl, $options);
        // detect target namespace $xml =
simplexml_load_file($wsdl);
        $this->_targetNamespace = (string)
$xml['targetNamespace'];
    }

    /**
     * Set credentials header
     *
     * @param string $login
     * @param string $password
     */
    public function setCredentialsHeader($login, $password) {
        $header = new SoapHeader($this->_targetNamespace,
            'CredentialsSoapHeader',
            new SoapVar(
                array(
                    'Login' => $login,
                    'Password' => $password,
                ),
                SOAP_ENC_OBJECT,
                'CredentialsSoapHeader',
                $this->_targetNamespace
            )
        );

        $this->__setSoapHeaders(array($header));
    }
}

}$serviceUrl = 'http://example.com/ServiceFacade/4.5/';
$serviceLogin = 'serviceLogin';
$servicePassword = 'servicePassword';
$siteService = new
Utils_SoapClient($serviceUrl.'/SiteWebService.asmx?WSDL', array());
$siteService->setCredentialsHeader($serviceLogin, $servicePassword);
$struct = new stdClass();
$struct->siteType = 'Regular';

```

```
$struct->siteAlias = 'siteName';
$struct->ownerId = 'a12b50410-da11-vf8e-m6c4-3f0cb537a9a3';
$struct->publishingSettings = new stdClass();
$struct->publishingSettings->Id = '112332';
$struct->publishingSettings->Mode = 'Ftp';
$struct->publishingSettings->StandardLocation = new stdClass();
$struct->publishingSettings->StandardLocation->Address =
'ftp://example.com';
$struct->publishingSettings->StandardLocation->UserName =
'myUserName';
$struct->publishingSettings->StandardLocation->Password =
'publishFtpPassword';
$struct->publishingSettings->StandardLocation->WorkingDirectory =
'/www/';
$struct->publishingSettings->StandardLocation->WebSiteUrl =
'http://example.com/website';
$struct->publishingSettings->StandardLocation->IsAnonymous = 'false';
$struct->publishingSettings->StandardLocation->Veid = 0;
$struct->publishingSettings->StandardLocation->VerifyStatus =
'Unverified';
$struct->publishingSettings->StandardLocation->LastVerifyAttempt =
'0000-00-00 00:00:00';
$struct->publishingSettings->StandardLocation->FallbackIP =
'192.0.2.1';
$struct->publishingSettings->StandardLocation->HostId = '12';
$result = $siteService->CreateSite2(new SoapVar($struct,
SOAP_ENC_OBJECT));
$siteId = $result->CreateSite2Result->Id;
?>
```

Creating Host Sample

```
<?php
/**
 * Extended SOAP client for simplified setup of credentials header
 *
 * @package Utils
 * @copyright (c) 2004-2008, Parallels
 */
class Utils_SoapClient extends SoapClient {
    protected $_targetNamespace;

    /**
     * Constructor
     *
     * @param mixed $wsdl
     * @param array $options
     */
    public function __construct($wsdl, $options) {
        parent::__construct($wsdl, $options);
        // detect target namespace $xml =
simplexml_load_file($wsdl);
        $this->_targetNamespace = (string)
$xml['targetNamespace'];
    }

    /**
     * Set credentials header
     *
     * @param string $login
     * @param string $password
     */
    public function setCredentialsHeader($login, $password) {
        $header = new SoapHeader($this->_targetNamespace,
            'CredentialsSoapHeader',
            new SoapVar(
                array(
                    'Login' => $login,
                    'Password' => $password,
                ),
                SOAP_ENC_OBJECT,
                'CredentialsSoapHeader',
                $this->_targetNamespace
            )
        );

        $this->__setSoapHeaders(array($header));
    }
}

}$serviceUrl = 'http://example.com/ServiceFacade/4.5/';
$serviceLogin = 'serviceLogin';
$servicePassword = 'servicePassword';
$hostService = new
Utils_SoapClient($serviceUrl.'/HostWebService.asmx?WSDL', array());
$hostService->setCredentialsHeader($serviceLogin, $servicePassword);
$struct = new stdClass();
$struct->hostName = 'myNewHost';
```

```
$struct->hostAddress = 'example.com';  
$struct->SMTPPort = 0;  
$result = $hostService->CreateHost(new SoapVar($struct,  
SOAP_ENC_OBJECT));  
$hostId = $result->CreateHostResult->Id;  
?>
```

Creating Plan Sample

```
<?php
/**
 * Extended SOAP client for simplified setup of credentials header
 *
 * @package Utils
 * @copyright (c) 2004-2008, Parallels
 */
class Utils_SoapClient extends SoapClient {
    protected $_targetNamespace;

    /**
     * Constructor
     *
     * @param mixed $wsdl
     * @param array $options
     */
    public function __construct($wsdl, $options) {
        parent::__construct($wsdl, $options);
        // detect target namespace $xml =
simplexml_load_file($wsdl);
        $this->_targetNamespace = (string)
$xml['targetNamespace'];
    }

    /**
     * Set credentials header
     *
     * @param string $login
     * @param string $password
     */
    public function setCredentialsHeader($login, $password) {
        $header = new SoapHeader($this->_targetNamespace,
            'CredentialsSoapHeader',
            new SoapVar(
                array(
                    'Login' => $login,
                    'Password' => $password,
                ),
                SOAP_ENC_OBJECT,
                'CredentialsSoapHeader',
                $this->_targetNamespace
            )
        );

        $this->__setSoapHeaders(array($header));
    }
}

}$serviceUrl = 'http://example.com/ServiceFacade/4.5/';
$serviceLogin = 'serviceLogin';
$servicePassword = 'servicePassword';
$planService = new
Utils_SoapClient($serviceUrl.'/PlanWebService.asmx?WSDL', array());
$planService->setCredentialsHeader($serviceLogin, $servicePassword);
$struct = new stdClass();
$struct->name = 'myNewPlan';
```

```
$struct->description = 'This is a sample plan.';
$struct->maxPagesNumber = 100;
$struct->maxPagesRootLevel = 10;
$struct->maxPagesLevel = 3;
$struct->maxSitesNumber = 5;
$struct->maxAccountsNumber = 15;
$struct->maxHostsNumber = 15;
$struct->isPersonal = true;
$struct->isAnonymous = true;
$struct->trialLifeTime = 2;
$struct->trialLifeType = 'Days';
$struct->templatesIds = new stdClass();
$struct->templatesIds->string = '15';
$struct->templatesIds->string = '19';
$struct->pagesetsIds = new stdClass();
$struct->pagesetsIds->string = '1';
$struct->pagesetsIds->string = '2';
$struct->modulesIds = new stdClass();
$struct->modulesIds->string = '2';
$struct->hostsIds = new stdClass();
$struct->hostsIds->string = '2';
$struct->families = new stdClass();
$struct->families->string = '1';
$struct->families->string = '3';
$struct->defaultFamily = '3';
$struct->isPublishingSettingsEditable = false;
$struct->isFtpPublishAvailable = true;
$struct->isXcopyPublishAvailable = false;
$struct->isVpsPublishAvailable = false;
$struct->isAdditionalSiteContentAllowed = false;
$struct->AdditionalSiteContent = '';
$struct->isUserManagementAllowed = true;
$struct->isSiteManagementAllowed = true;
$result = $planService->CreatePlan(new SoapVar($struct,
SOAP_ENC_OBJECT));
$hostId = $result->CreatePlanResult->planId;
?>
```

Modifying Plan Sample

```
<?php
/**
 * Extended SOAP client for simplified setup of credentials header
 *
 * @package Utils
 * @copyright (c) 2004-2008, Parallels
 */
class Utils_SoapClient extends SoapClient {
    protected $_targetNamespace;

    /**
     * Constructor
     *
     * @param mixed $wsdl
     * @param array $options
     */
    public function __construct($wsdl, $options) {
        parent::__construct($wsdl, $options);
        // detect target namespace $xml =
simplexml_load_file($wsdl);
        $this->_targetNamespace = (string)
$xml['targetNamespace'];
    }

    /**
     * Set credentials header
     *
     * @param string $login
     * @param string $password
     */
    public function setCredentialsHeader($login, $password) {
        $header = new SoapHeader($this->_targetNamespace,
            'CredentialsSoapHeader',
            new SoapVar(
                array(
                    'Login' => $login,
                    'Password' => $password,
                ),
                SOAP_ENC_OBJECT,
                'CredentialsSoapHeader',
                $this->_targetNamespace
            )
        );

        $this->__setSoapHeaders(array($header));
    }
}

}$serviceUrl = 'http://example.com/ServiceFacade/4.5/';
$serviceLogin = 'serviceLogin';
$servicePassword = 'servicePassword';
$planService = new
Utils_SoapClient($serviceUrl.'/PlanWebService.asmx?WSDL', array());
$planService->setCredentialsHeader($serviceLogin, $servicePassword);
$struct = new stdClass();
$struct->planId = '35b50410-da00-df8e-f6c4-1f0cb537a9a3';
```

```
$struct->name = 'myNewNewPlan';
$struct->description = 'This is a modified plan.';
$struct->maxPagesNumber = 100;
$struct->maxPagesRootLevel = 10;
$struct->maxPagesLevel = 3;
$struct->maxSitesNumber = 5;
$struct->maxAccountsNumber = 15;
$struct->maxHostsNumber = 15;
$struct->trialLifeTime = 2;
$struct->trialLifeType = 'Days';
$struct->isPublishingSettingsEditable = false;
$struct->isFtpPublishAvailable = true;
$struct->isXcopyPublishAvailable = false;
$struct->isVpsPublishAvailable = false;
$struct->isAdditionalSiteContentAllowed = false;
$struct->isUserManagementAllowed = true;
$struct->isSiteManagementAllowed = true;
$result = $planService->UpdatePlan(new SoapVar($struct,
SOAP_ENC_OBJECT));
?>
```

Business Scenarios Code

```
<?php
/**
 * Extended SOAP client for simplified setup of credentials header
 *
 * @package Utils
 * @copyright (c) 2004-2008, Parallels
 */
class Utils_SoapClient extends SoapClient {
    protected $_targetNamespace;

    /**
     * Constructor
     *
     * @param mixed $wsdl
     * @param array $options
     */
    public function __construct($wsdl, $options) {
        parent::__construct($wsdl, $options);
        // detect target namespace $xml =
        simplexml_load_file($wsdl);
        $this->_targetNamespace = (string)
        $xml['targetNamespace'];
    }

    /**
     * Set credentials header
     *
     * @param string $login
     * @param string $password
     */
    public function setCredentialsHeader($login, $password) {
        $header = new SoapHeader($this->_targetNamespace,
            'CredentialsSoapHeader',
            new SoapVar(
                array(
                    'Login' => $login,
                    'Password' => $password,
                ),
                SOAP_ENC_OBJECT,
                'CredentialsSoapHeader',
                $this->_targetNamespace
            )
        );

        $this->__setSoapHeaders(array($header));
    }
}

/**
 * Error handler class
 *
 * @package Utils
 * @copyright (c) 2004-2008, Parallels
 */
```

```

class Utils_ErrorHandler {

    /**
     * Exceptions handler
     *
     * @param Exception $exception
     */
    public static function exceptionHandler($exception) {
        $fullMessage = "Type: " . get_class($exception) . "<br>\n"
            . "Message: {$exception->getMessage()}<br>\n"
            . (('SoapFault' == get_class($exception))
                ? "Fault code: {$exception->faultcode}<br>\n"
                : "Details: {$exception->detail}<br>\n"
            )
            . "File: {$exception->getFile()}<br>\n"
            . "Line: {$exception->getLine()}<br>\n"
            . "Code: {$exception->getCode()}";
        self::_print($fullMessage, 'Exception');
    }

    /**
     * PHP error handler
     *
     * @param int $code
     * @param string $message
     * @param string $file
     * @param int $line
     * @param array $context
     */
    public static function phpErrorHandler($code, $message, $file,
        $line, $context) {
        if (0 == ini_get('error_reporting')) {
            return;
        }

        $fullMessage = "Message: $message<br>\n"
            . "File: $file<br>\n"
            . "Line: $line<br>\n"
            . "Code: $code";

        self::_print($fullMessage, 'PHP error');
    }

    /**
     * Print message with title
     *
     * @param string $title
     * @param string $message
     */
    protected static function _print($message, $title = '') {
        echo "<fieldset><legend><font
color='red'>$title</font></legend>$message</fieldset>";
        exit(1);
    }
}

/**
 * API usage scenario
 */

```

```

* @package Utils/ApiUsage
* @copyright (c) 2004-2008, Parallels
*/
class Utils_ApiUsage_Scenario {
    protected $_locationUrl;
    protected $_servicesDirectory;
    protected $_servicesUrl;
    protected $_adminLogin;
    protected $_adminPassword;

    /**
     * Set scenario configuration
     *
     * @param string $servicesUrl
     * @param string $adminLogin
     * @param string $adminPassword
     */
    public function setConfiguration($locationUrl,
    $servicesDirectory, $adminLogin, $adminPassword) {
        $this->locationUrl = $locationUrl;
        $this->_servicesDirectory = $servicesDirectory;
        $this->_servicesUrl = $locationUrl . $servicesDirectory;
        $this->_adminLogin = $adminLogin;
        $this->_adminPassword = $adminPassword;
    }

    /**
     * Get web service instance
     *
     * @param string $serviceHandler
     * @return Utils_SoapClient
     */
    private function _getWebService($serviceHandler) {
        $service = new Utils_SoapClient($this->_servicesUrl . '/'
    . $serviceHandler, array());
        $service->setCredentialsHeader($this->_adminLogin, $this-
    >_adminPassword);
        return $service;
    }

    /**
     * Get list of plans
     *
     * @return array
     */
    private function _getPlansList() {
        $planService = $this-
    >_getWebService('/PlanWebService.asmx?WSDL');
        $struct = new stdClass();
        $struct->accountId = '';

        $result = $planService->GetAvailablePlans(new
    SoapVar($struct, SOAP_ENC_OBJECT));
        if (0 == $result->GetAvailablePlansResult->TotalCount) {
            throw new Exception('List of plans is empty. Please
    create at least one plan.');
```

```

        ? array($result->GetAvailablePlansResult->Items-
>PlanValue)
        : $result->GetAvailablePlansResult->Items-
>PlanValue;
    $planList = array();
    foreach ($planValues as $planValue) {
        $planList[$planValue->PlanId] = $planValue->Name;
    }

    return $planList;
}

private function _renderHeader() {
    echo '
<html>
<head>
    <title>Creating Plesk Sitebuilder Account</title>
    <style>
        body, td {
            background-color: #FFFFFF;
            color:#000000;
            font-family: Tahoma;
            font-size: 10pt;
        }

        table {
            border-collapse: collapse;
            border-spacing: 0px;
        }

        td {
            border: 1px solid #D1D1D1;
        }

        input, select {
            width: 100%;
        }

        .submitInput {
            width: 100px;
            height: 40px;
            font-weight: bold;
            font-size: 14pt;
        }
    </style>
</head>
<body>
    <center>
        <h1>Sample code for Plesk Sitebuilder 4.5 for Linux/Unix
integration</h1>
    </center>
';
}

private function _renderInfo() {
    echo '
Copyright &copy; 2008 by <a target="_blank"
href="http://www.parallels.com">Parallels.</a><br/>
<b>NOTE:</b> This code is provided as is, without any warranty, either
express or implied.

```

```

Please do not use this sample in a production environment.<br/>
<b>WARNING:</b> Administrator credentials are sent in clear text.<br/>
<br/>
';
    }

    private function _renderFooter() {
        echo '
</body>
</html>
';
    }

    private function _renderForm() {
        $plans = $this->_getPlansList();
        echo '
<form method="post">
    <input name="action" type="hidden" value="create">
    <input name="siteId" type="hidden" value="' . @$_GET['siteId'] .
'">
    <table cellpadding="5" cellspacing="0" width="100%">
        <tr>
            <td width="50%">First name</td>
            <td><input name="accountFirstName" type="text"
value="John"></td>
        </tr>
        <tr>
            <td>Last name</td>
            <td><input name="accountLastName" type="text"
value="Smith"></td>
        </tr>
        <tr>
            <td>User name</td>
            <td><input name="accountUserName" type="text"
value="testuser"></td>
        </tr>
        <tr>
            <td>Password</td>
            <td><input name="accountPassword" type="text"
value="testpassword"></td>
        </tr>
        <tr>
            <td>E-mail</td>
            <td><input name="accountEmail" type="text"
value="john@example.com"></td>
        </tr>
        <tr>
            <td>Plan</td>
            <td>
                <select name="planId">
';

                foreach ($plans as $planId => $planName) {
                    echo "<option value='$planId'>$planName</option>";
                }

                echo '
                    </select>
                </td>
            </tr>
';
    }

```

```

</table>
<br/>
<table cellpadding="5" cellspacing="0" width="100%">
  <tr>
    <td width="50%">Site name</td>
    <td><input name="siteName" type="text" value="Test
site"></td>
  </tr>
</table>
<br/>
<center><input class="submitInput" type="submit"
value="Create"></center>
</form>
';
}

private function _renderSuccess($userName, $userPassword,
$siteId) {
    $baseUrl = $this->_locationUrl . '/external_login.php?'
        . 'Login=' . $userName
        . '&Password=' . $userPassword;
    $wizardUrl = $baseUrl . '&SiteId=' . $siteId;
    $adminUrl = $baseUrl . '&ShowAdmin=true';
    echo "You have successfully registered. ";
    echo "Please <a href='$wizardUrl'>click here</a> to edit
your site at Wizard";
    echo " or <a href='$adminUrl'>click here</a> to enter
Plesk Sitebuilder siteowner's panel.";
}

/**
 * Process form
 */
private function _processForm() {
    $result = new stdClass();
    $struct = new stdClass();
    $struct->username = $_POST['accountUserName'];
    $struct->password = $_POST['accountPassword'];
    $struct->firstName = $_POST['accountFirstName'];
    $struct->lastName = $_POST['accountLastName'];
    $struct->email = $_POST['accountEmail'];
    $struct->role = 'SiteOwner';
    $struct->planId = $_POST['planId'];
    $accountService = $this->
>_getService('AccountWebService.asmx?WSDL');
    $result = $accountService->CreateAccount(new
SoapVar($struct, SOAP_ENC_OBJECT));
    $accountId = $result->UserAccount->AccountId;
    $siteService = $this->
>_getService('SiteWebService.asmx?WSDL');
    if (!empty($_POST['siteId'])) {
        $struct = new stdClass();
        $struct->ownerId = $accountId;
        $struct->siteId = $_POST['siteId'];
        $struct->alias = $_POST['siteName'];
        $siteService->TakeOwnershipOfAnonymousSite(new
SoapVar($struct, SOAP_ENC_OBJECT));
        $siteId = $_POST['siteId'];
    } else {

```

```

        $struct = new stdClass();
        $struct->ownerId = $accountId;
        $struct->siteType = 'Regular';
        $struct->siteAlias = $_POST['siteName'];
        $result = $siteService->CreateSite(new
SoapVar($struct, SOAP_ENC_OBJECT));
        $siteId = $result->CreateSiteResult->Id;
    }

    $this->_renderSuccess($_POST['accountUserName'],
$_POST['accountPassword'], $siteId);
}

/**
 * Run scenario
 */
public function run() {
    if ('create' == @$_POST['action']) {
        $this->renderHeader();
        $this->_processForm();
        $this->_renderFooter();
    } else {
        $this->_renderHeader();
        $this->_renderInfo();
        $this->_renderForm();
        $this->_renderFooter();
    }
}
}

// set up customized error handlers
set_error_handler(array('Utils_ErrorHandler', 'phpErrorHandler'));
set_exception_handler(array('Utils_ErrorHandler',
'exceptionHandler'));
// create application and run it
$scenario = new Utils_ApiUsage_Scenario();

// set up configuration parameters
$locationUrl = 'http://localhost:2006/';
$servicesDirectory = '/ServiceFacade/4.5/';
$adminLogin = 'admin';
$adminPassword = 'admin';

$scenario->setConfiguration($locationUrl, $servicesDirectory,
$adminLogin, $adminPassword);
$scenario->run();

```

Appendix 4. ASP Snippets

In this chapter:

Core Functions.....	66
Creating User Account Sample	77
Creating Site Sample	79
Creating Host Sample	81
Create Plan Sample	83
Modifying Plan Sample.....	86
Business Scenarios Code	89

Core Functions

```

<%
  Function CreateHost(HostName, HostAddress)
  Const SMTPPort = 0;

  dim url, xmlhttp, XMLDOM, XMLNode
  ' Call web service using HTTP-POST
  url = baseUrl & "/ServiceFacade/4.5/HostWebService.asmx"
  Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
  Call xmlhttp.Open("POST", url, False)
  Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
  dim body
  body = "<?xml version=""1.0"" encoding=""utf-8""?>"
  body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
  body = body & " <soap12:Header>"
  body = body & " <CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/HostService"">"
  body = body & " <Login>" & adminLogin & "</Login>"
  body = body & " <Password>" & adminPassword & "</Password>"
  body = body & " </CredentialsSoapHeader>"
  body = body & " </soap12:Header>"
  body = body & " <soap12:Body>"
  body = body & " <CreateHost
xmlns=""http://swsoft.com/webservices/sb/4.5/HostService"">"
  body = body & " <hostName>" & HostName & "</hostName>"
  body = body & " <hostAddress>" & HostAddress & "</hostAddress>"
  body = body & " <SMTPPort>" & SMTPPort & "</SMTPPort>"
  body = body & " </CreateHost >"
  body = body & " </soap12:Body>"
  body = body & " </soap12:Envelope>"

  Call xmlhttp.send(body)
  ' Parse result
  Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
  XMLDOM.Load(xmlhttp.responseBody)
  Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
  if Not XMLNode Is Nothing then
    Dim XMLdata
    Set XMLdata = XMLNode.firstChild
    if Not XMLdata is Nothing then
      if XMLdata.nodeName = "soap:Fault" then
        response.Write "FAULT!<br><pre>" & XMLdata.text & "</pre>"
        CreateHost=""
      else
        Set XMLdata =
XMLDOM.SelectSingleNode("//CreateHostResponse/CreateHostResult/Id")
        CreateHost = XMLdata.text
      end if
    else
      CreateHost=""
      response.Write "Error parsing SOAP response"
    end if
  end if
end Function

```

```

end if
End Function

Function CreateAccount (accountName, accountPassword, firstName,
lastName, eMail, planId)
dim url, xmlhttp, XMLDOM, XMLNode
' Call web service using HTTP-POST
url = baseUrl & "/ServiceFacade/4.5/AccountWebService.asmx"
Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
Call xmlhttp.Open("POST", url, False)
Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
dim body
body = "<?xml version=""1.0"" encoding=""utf-8""?>"
body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
body = body & "<soap12:Header>"
body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/AccountService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<CreateAccount
xmlns=""http://swsoft.com/webservices/sb/4.5/AccountService"">"
body = body & "<ownerAccountId></ownerAccountId>"
body = body & "<username>" & accountName & "</username>"
body = body & "<password>" & accountPassword & "</password>"
body = body & "<firstName>" & firstName & "</firstName>"
body = body & "<lastName>" & lastName & "</lastName>"
body = body & "<email>" & eMail & "</email>"
body = body & "<role>SiteOwner</role>"
body = body & "<planId>" & planId & "</planId>"
body = body &
"<changePasswordAllowed>true</changePasswordAllowed>"
body = body & "</CreateAccount>"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

Call xmlhttp.send(body)
' Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseBody)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
CreateAccount = XMLDOM.xml 'XMLNode.xml
if Not XMLNode Is Nothing then
Dim XMLdata
Set XMLdata = XMLNode.firstChild
if Not XMLdata is Nothing then
if XMLdata.nodeName = "soap:Fault" then
response.Write "FAULT!<br><pre>" & XMLData.text & "</pre>"
CreateAccount=""
else
Set XMLData =
XMLDOM.SelectSingleNode("//CreateAccountResponse/UserAccount/AccountId
")

```

```

        CreateAccount = XMLData.text
    end if
else
    CreateAccount=""
    response.Write "Error parsing SOAP response"
end if
end if
End Function

Function TakeOwnership(accountId, siteId, siteAlias)
    dim url, xmlhttp, XMLDOM, XMLNode
    ' Call web service using HTTP-POST
url = baseUrl & "/ServiceFacade/4.5/SiteWebService.aspx"
    Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
    Call xmlhttp.Open("POST", url, False)
    Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
    dim body
    body = "<?xml version=""1.0"" encoding=""utf-8""?>"
    body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
    body = body & " <soap12:Header>"
    body = body & " <CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/SiteService"">"
    body = body & " <Login>" & adminLogin & "</Login>"
    body = body & " <Password>" & adminPassword & "</Password>"
    body = body & " </CredentialsSoapHeader>"
    body = body & "</soap12:Header>"
    body = body & "<soap12:Body>"
    body = body & " <TakeOwnershipOfAnonymousSite
xmlns=""http://swsoft.com/webservices/sb/4.5/SiteService"">"
    body = body & " <siteId>" & siteId & "</siteId>"
    body = body & " <ownerId>" & accountId & "</ownerId>"
    body = body & " <alias>" & siteAlias & "</alias>"
    body = body & " </TakeOwnershipOfAnonymousSite>"
    body = body & "</soap12:Body>"
    body = body & "</soap12:Envelope>"

    Call xmlhttp.send(body)
    ' Parse result
    TakeOwnership=false
    Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
    XMLDOM.Load(xmlhttp.responseBody)
    Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
    if Not XMLNode Is Nothing then
        Dim XMLdata
        Set XMLdata = XMLNode.firstChild
        if Not XMLdata is Nothing then
            if XMLdata.nodeName = "soap:Fault" then
                response.Write "FAULT!<br><pre>" & XMLData.text & "</pre>"
            else
                TakeOwnership=true
            end if
        else
            response.Write "Error parsing SOAP response"
        end if
    end if
end if

```

```

End Function

Function ListPlans()
    dim url, xmlhttp, XMLDOM, XMLNodeList, x, str
    ' Call web service using HTTP-POST url = baseUrl &
    "/ServiceFacade/4.5/PlanWebService.asmx" Set xmlhttp =
    Server.CreateObject("MSXML2.ServerXMLHTTP") Call xmlhttp.Open("POST",
    url, False)
    Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
    charset=utf-8")
    dim body
    body = "<?xml version=""1.0"" encoding=""utf-8""?>"
    body = body & "<soap12:Envelope
    xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
    xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
    xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
    body = body & "<soap12:Header>"
    body = body & "<CredentialsSoapHeader
    xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService"">"
    body = body & "<Login>" & adminLogin & "</Login>"
    body = body & "<Password>" & adminPassword & "</Password>"
    body = body & "</CredentialsSoapHeader>"
    body = body & "</soap12:Header>"
    body = body & "<soap12:Body>"
    body = body & "<GetAvailablePlans
    xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService"">"
    body = body & "<accountId></accountId>"
    body = body & "</GetAvailablePlans>"
    body = body & "</soap12:Body>"
    body = body & "</soap12:Envelope>"
    Call xmlhttp.send(body)
    ' Parse result
    Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
    XMLDOM.Load(xmlhttp.responseBody)
    Set XMLNodeList = XMLDOM.selectNodes("//PlanValue")

    For x = 1 To XMLNodeList.length
        str = str & "<option value="" & XMLNodeList.Item(x -
    1).selectSingleNode("PlanId").text & """">" & XMLNodeList.Item(x -
    1).selectSingleNode("Name").text
    Next
    ListPlans=str
End Function

Function CreateSite(ownerId, siteAlias)
    dim url, xmlhttp, XMLDOM, XMLNode
    ' Call web service using HTTP-POST
    url = baseUrl & "/ServiceFacade/4.5/SiteWebService.asmx"
    Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
    Call xmlhttp.Open("POST", url, False)
    Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
    charset=utf-8")
    dim body
    body = "<?xml version=""1.0"" encoding=""utf-8""?>"

```

```

body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
body = body & "<soap12:Header>"
body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/SiteService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<CreateSite
xmlns=""http://swsoft.com/webservices/sb/4.5/SiteService"">"
body = body & "<siteType>Regular</siteType>"
body = body & "<siteAlias>" & siteAlias & "</siteAlias>"
body = body & "<ownerId>" & ownerId & "</ownerId>"
body = body & "</CreateSite >"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

Call xmlhttp.send(body)
' Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseBody)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
if Not XMLNode Is Nothing then
    Dim XMLdata
    Set XMLdata = XMLNode.firstChild
    if Not XMLdata is Nothing then
        if XMLdata.nodeName = "soap:Fault" then
            response.Write "FAULT!<br><pre>" & XMLdata.text & "</pre>"
            CreateSite=""
        else
            Set XMLdata =
XMLDOM.SelectSingleNode("//CreateSiteResponse/CreateSiteResult/Id")
            CreateSite = XMLdata.text
        end if
    else
        CreateSite=""
        response.Write "Error parsing SOAP response"
    end if
end if
End Function

Function CreatePlan(PlanName, PlanDescription, PMpagesLimit,
PMpagesrootLevel,PMpagesLevel, PMSitesNumber, PMAccountsNumber,
PMhostsNumber,PisPersonal, PisAnonymous, PTrialLifeTime,
PTrialLifeType,
PTemplatesID, PPageSetsID, PModulesID, PHostsID, PFamiliesID,
PDefaultFamilyID, PIsPublishSettingsEditable, PIsFTPPublishAvailable,
PIsXcopyPublishAvailable, PIsVPSPublishAvailable,
PIsAdditionalSiteContentAllowed, PAdditionalSiteContent,
PIsUserManagementAllowed, PIsSiteManagementAllowed)
    dim url, xmlhttp, XMLDOM, XMLNode
    ' Call web service using HTTP-POST
url = baseUrl & "/ServiceFacade/4.5/PlanWebService.asmx"
    Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
    Call xmlhttp.Open("POST", url, False)

```

```

Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
dim body
body = "<?xml version=""1.0"" encoding=""utf-8""?>"
body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
body = body & "<soap12:Header>"
body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<CreatePlan
xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService""> "
body = body & "<name>" & PlanName & "</name>"
body = body & "<description>" & PlanDescription & "</description>"
body = body & "<maxPagesNumber>" & PMpagesLimit & "</maxPagesNumber>"
body = body & "<maxPagesRootLevel>" & PMpagesrootLevel &
"</maxPagesRootLevel>"
body = body & "<maxPagesLevel>" & PMpagesLevel & "</maxPagesLevel>"
body = body & "<maxSitesNumber>" & PmsitesNumber &
"</maxSitesNumber>"
body = body & "<maxAccountsNumber>" & PmaxAccountsNumber &
"</maxAccountsNumber>"
body = body & "<maxHostsNumber>" & PMhostsNumber &
"</maxHostsNumber>"
body = body & "<isPersonal>" & PisPersonal & "</isPersonal>"
body = body & "<isAnonymous>" & PisAnonymous & "</isAnonymous>"
body = body & "<trialLifeTime>" & PtrialLifeTime & "</trialLifeTime>"
body = body & "<trialLifeType>" & PtrialLifeType & "</trialLifeType>"
body = body & "<templatesIds> <string>" & PtemplatesID &
"</string></templatesIds>"
body = body & "<pagesetsIds><string>" & PpageSetsID &
"</string></pagesetsIds>"
body = body & "<modulesIds> <string>" & PmodulesID &
"</string></modulesIds>"
body = body & "<hostsIds> <string>" & PhostsID &
"</string></hostsIds>"
body = body & "<families> <string>" & PfamiliesID &
"</string></families>"
body = body & "<defaultFamily>" & PdefaultFamilyID &
"</defaultFamily>"
body = body & "<isPublishingSettingsEditable>" &
PisPublishSettingsEditable & "</isPublishingSettingsEditable>"
body = body & "<isFtpPublishAvailable>" & PIsFTPPublishAvailable &
"</isFtpPublishAvailable>"
body = body & "<isXcopyPublishAvailable>" & PIsXcopyPublishAvailable
& "</isXcopyPublishAvailable>"
body = body & "<isVpsPublishAvailable>" & PIsVSPublishAvailable &
"</isVpsPublishAvailable>"
body = body & "<isAdditionalSiteContentAllowed>" &
PisAdditionalSiteContentAllowed & "</isAdditionalSiteContentAllowed>"
body = body & "<additionalSiteContent>" & PAdditionalSiteContent &
"</additionalSiteContent>"
body = body & "<isUserManagementAllowed>" & PIsUserManagementAllowed
& "</isUserManagementAllowed>"

```

```

body = body & "<isSiteManagementAllowed>" & PIsSiteManagementAllowed
& "</isSiteManagementAllowed>"
body = body & "</CreateHost >"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

Call xmlhttp.send(body)
' Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseBody)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
if Not XMLNode Is Nothing then
    Dim XMLdata
    Set XMLdata = XMLNode.firstChild
    if Not XMLdata is Nothing then
        if XMLdata.nodeName = "soap:Fault" then
            response.Write "FAULT!<br><pre>" & XMLdata.text & "</pre>"
            CreatePlan=""
        else
            Set XMLdata =
XMLDOM.SelectSingleNode("//CreateHostResponse/CreatePlanResult/PlanId"
)
            CreatePlan = XMLdata.text
        end if
    else
        CreatePlan=""
        response.Write "Error parsing SOAP response"
    end if
end if
End Function

Function UpdatePlan(PlanId, PlanName, PlanDescription, PMpagesLimit,
PMpagesrootLevel,PMpagesLevel, PMSitesNumber, PMaccountsNumber,
PMhostsNumber, PTrialLifeTime, PTrialLifeType,
PTemplatesID, PPageSetsID, PModulesID, PHostsID, PFamiliesID,
PDefaultFamilyID, PIsPublishSettingsEditable, PIsFTTPublishAvailable,
PIsXcopyPublishAvailable, PIsVPSPublishAvailable,
PIsAdditionalSiteContentAllowed, PAdditionalSiteContent,
PIsUserManagementAllowed, PIsSiteManagementAllowed)
    dim url, xmlhttp, XMLDOM, XMLNode
    ' Call web service using HTTP-POST
url = baseUrl & "/ServiceFacade/4.5/PlanWebService.asmx"
    Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
    Call xmlhttp.Open("POST", url, False)
    Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
    dim body
    body = "<?xml version=""1.0"" encoding=""utf-8""?>"
    body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
    body = body & "<soap12:Header>"
    body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService"">"

```

```

body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<CreatePlan
xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService""> "
body = body & "<planId>" & PlanId & "</planId>"
body = body & "<name>" & PlanName & "</name>"
body = body & "<description>" & PlanDescription & "</description>"
body = body & "<maxPagesNumber>" & PMpagesLimit & "</maxPagesNumber>"
body = body & "<maxPagesRootLevel>" & PMpagesrootLevel &
"</maxPagesRootLevel>"
body = body & "<maxPagesLevel>" & PMpagesLevel & "</maxPagesLevel>"
body = body & "<maxSitesNumber>" & PMsitesNumber &
"</maxSitesNumber>"
body = body & "<maxAccountsNumber>" & PMaccountsNumber &
"</maxAccountsNumber>"
body = body & "<maxHostsNumber>" & PMhostsNumber &
"</maxHostsNumber>"
body = body & "<trialLifeTime>" & PTrialLifeTime & "</trialLifeTime>"
body = body & "<trialLifeType>" & PTrialLifeType & "</trialLifeType>"
body = body & "<templatesIds> <string>" & PTemplatesID &
"</string></templatesIds>"
body = body & "<pagesetsIds><string>" & PPageSetsID &
"</string></pagesetsIds>"
body = body & "<modulesIds> <string>" & PModulesID &
"</string></modulesIds>"
body = body & "<hostsIds> <string>" & PHostsID &
"</string></hostsIds>"
body = body & "<families> <string>" & PFamiliesID &
"</string></families>"
body = body & "<defaultFamily>" & PDefaultFamilyID &
"</defaultFamily>"
body = body & "<isPublishingSettingsEditable>" &
PIsPublishSettingsEditable & "</isPublishingSettingsEditable>"
body = body & "<isFtpPublishAvailable>" & PIsFTPPublishAvailable &
"</isFtpPublishAvailable>"
body = body & "<isXcopyPublishAvailable>" & PIsXcopyPublishAvailable
& "</isXcopyPublishAvailable>"
body = body & "<isVpsPublishAvailable>" & PIsVPSPublishAvailable &
"</isVpsPublishAvailable>"
body = body & "<isAdditionalSiteContentAllowed>" &
PIsAdditionalSiteContentAllowed & "</isAdditionalSiteContentAllowed>"
body = body & "<additionalSiteContent>" & PAdditionalSiteContent &
"</additionalSiteContent>"
body = body & "<isUserManagementAllowed>" & PIsUserManagementAllowed
& "</isUserManagementAllowed>"
body = body & "<isSiteManagementAllowed>" & PIsSiteManagementAllowed
& "</isSiteManagementAllowed>"
body = body & "</CreateHost >"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

Call xmlhttp.send(body)
' Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseBody)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
if Not XMLNode Is Nothing then

```

```

Dim XMLdata
Set XMLdata = XMLNode.firstChild
if Not XMLdata is Nothing then
    if XMLdata.nodeName = "soap:Fault" then
        Err.Raise 100, "UpdatePlan", XMLdata.Text
    end if
else
    Err.Raise 100, "UpdatePlan", "Error parsing SOAP response"
end if
end if
End function

<%
function RenderSuccess (login, password, siteId)
    dim adminUrl, wizardUrl
    adminUrl=baseUrl & "/ExternalLogin.ashx?Login=" & login &
"&Password=" & password
    wizardUrl=adminUrl & "&SiteID=" & siteId
    adminUrl=adminUrl & "&ShowAdmin=true"
%>
You have successfully registered. Please <a
href="<%=wizardUrl%>">click here</a> to edit your site at Wizard or <a
href="<%=adminUrl%>">click here</a> to enter Sitebuilder siteowner's
panel.
<%
End function
%>

<%
function RenderForm
%>
<h3>
    SAMPLE CODE FOR SITEBUILDER 4.5 FOR WINDOWS INTEGRATION</h3>
Copyright&copy; 2008 by <a href="parallels.com">Parallels Inc</a><br>
NOTE: This code is provided as is, without any warranty, either
express or implied
Please do not use this sample in a production environment WARNING!
Administrator
credentials are sent in clear text. It is by default assumed that this
script will
be run on the same machine as the Sitebuilder installation. In case
you want to
administer the Sitebuilder remotely using this Web Service API, make
sure to secure
the communication (for example, by using HTTPS protocol instead of
HTTP).
<form method="post">
    <input type="hidden" name="siteId" value="<% response.Write
newSiteId%>" />
    <table>
        <tr>
            <td>

```

```

                Account Name:
            </td>
            <td>
                <input type="text" name="accountName"
value="TestUserFromASP"></td>
            </tr>
            <tr>
                <td>
                    Account Password:
                </td>
                <td>
                    <input type="text" name="accountPassword"
value="111111"></td>
            </tr>
            <tr>
                <td>
                    First Name:
                </td>
                <td>
                    <input type="text" name="firstName"
value="Bob"></td>
            </tr>
            <tr>
                <td>
                    Last Name:
                </td>
                <td>
                    <input type="text" name="lastName"
value="Smith"></td>
            </tr>
            <tr>
                <td>
                    E-Mail:</td>
                <td>
                    <input type="text" name="eMail"
value="none@nowhere.org"></td>
            </tr>
            <tr>
                <td>
                    Plan:
                </td>
                <td>
                    <select name="plan">
                        <%response.Write ListPlans() %>
                    </select>
                </td>
            </tr>
            <tr><td colspan="2"><hr /></td></tr>
            <tr>
                <td>
                    Site alias:
                </td>
                <td>
                    <input type="text" name="siteAlias"
value="Test Site">
                </td>
            </tr>
            <tr>
                <td colspan="2">

```

```
                                <input type="submit" name="action"
value="create"></td>
                                </tr>
                                </table>
</form>
<%
End function
%>
```

Creating User Account Sample

```

<%
dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId
dim error
baseUrl="http://localhost:2006"
adminLogin="admin"
adminPassword="admin"

dim accountName, accountPassword, firstName, lastName, eMail, plan
accountName = "MyAccount"
accountPassword = "MyPassword"
firstName = "John"
lastName = "Doe"
eMail = "john@example.com"
plan = "1"

AccountID = CreateAccount(accountName, accountPassword, firstName,
lastName, eMail, plan)
'
'Function that is used to create Plesk Sitebuilder accounts.
'
Function CreateAccount (accountName, accountPassword, firstName,
lastName, eMail, planId)
dim url, xmlhttp, XMLDOM, XMLNode
'Call web service using HTTP-POST
url = baseUrl & "/ServiceFacade/4.5/AccountWebService.asmx"
Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
Call xmlhttp.Open("POST", url, False)
Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
dim body
body = "<?xml version=""1.0"" encoding=""utf-8""?>"
body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
body = body & "<soap12:Header>"
body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/AccountService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<CreateAccount
xmlns=""http://swsoft.com/webservices/sb/4.5/AccountService"">"
body = body & "<ownerAccountId></ownerAccountId>"
body = body & "<username>" & accountName & "</username>"
body = body & "<password>" & accountPassword & "</password>"
body = body & "<firstName>" & firstName & "</firstName>"
body = body & "<lastName>" & lastName & "</lastName>"
body = body & "<email>" & eMail & "</email>"
body = body & "<role>SiteOwner</role>"
body = body & "<planId>" & planId & "</planId>"
body = body &
"<changePasswordAllowed>>true</changePasswordAllowed>"

```

```
body = body & "</CreateAccount>"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

Call xmlhttp.send(body)
'Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseBody)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
CreateAccount = XMLDOM.xml 'XMLNode.xml
if Not XMLNode Is Nothing then
    Dim XMLdata
    Set XMLdata = XMLNode.firstChild
    if Not XMLdata is Nothing then
        if XMLdata.nodeName = "soap:Fault" then
            response.Write "FAULT!<br><pre>" & XMLdata.text & "</pre>"
            CreateAccount=""
        else
            Set XMLdata =
XMLDOM.SelectSingleNode("//CreateAccountResponse/UserAccount/AccountId
")
            CreateAccount = XMLdata.text
        end if
    else
        CreateAccount=""
        response.Write "Error parsing SOAP response"
    end if
end if
End Function
%>
```

Creating Site Sample

```

<%
dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId
dim error
baseUrl="http://localhost:2006"
adminLogin="admin"
adminPassword="admin"
SiteOwnerID ="133"
SiteName ="My Site"
NewSiteID = CreateSite(SiteOwnerID, SiteName)
'
'Function that is used to create a Plesk Sitebuilder site.
'

Function CreateSite(ownerId, siteAlias)
dim url, xmlhttp, XMLDOM, XMLNode
' Call web service using HTTP-POST
url = baseUrl & "/ServiceFacade/4.5/SiteWebService.asmx"
Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
Call xmlhttp.Open("POST", url, False)
Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
dim body
body = "<?xml version=""1.0"" encoding=""utf-8""?>"
body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
body = body & "<soap12:Header>"
body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/SiteService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<CreateSite
xmlns=""http://swsoft.com/webservices/sb/4.5/SiteService"">"
body = body & "<siteType>Regular</siteType>"
body = body & "<siteAlias>" & siteAlias & "</siteAlias>"
body = body & "<ownerId>" & ownerId & "</ownerId>"
body = body & "</CreateSite >"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

Call xmlhttp.send(body)
'Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.ResponseBody)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
if Not XMLNode Is Nothing then
Dim XMLdata
Set XMLdata = XMLNode.firstChild
if Not XMLdata is Nothing then
if XMLdata.nodeName = "soap:Fault" then
response.Write "FAULT!<br><pre>" & XMLData.text & "</pre>"

```

```
        CreateSite=""
    else
        Set XMLData =
XMLDOM.SelectSingleNode("//CreateSiteResponse/CreateSiteResult/Id")
        CreateSite = XMLData.text
    end if
    else
        CreateSite=""
        response.Write "Error parsing SOAP response"
    end if
end if
End Function
%>
```

Creating Host Sample

```

<%
dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId
dim error
baseUrl="http://localhost:2006"
adminLogin="admin"
adminPassword="admin"
HostName ="My new host"
HostAddress ="192.0.2.12"
NewHostID = CreateHost(HostName, HostAddress)
'
' This function is used to add a host to Plesk Sitebuilder.
'
Function CreateHost(HostName, HostAddress)
Const SMTPPort = 0;

    dim url, xmlhttp, XMLDOM, XMLNode
    'Call web service using HTTP-POST
    url = baseUrl & "/ServiceFacade/4.5/HostWebService.asmx"
    Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
    Call xmlhttp.Open("POST", url, False)
    Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml; charset=utf-8")
    dim body
    body = "<?xml version=""1.0"" encoding=""utf-8""?>"
    body = body & "<soap12:Envelope"
    xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
    xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
    xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
    body = body & " <soap12:Header>"
    body = body & " <CredentialsSoapHeader"
    xmlns=""http://swsoft.com/webservices/sb/4.5/HostService"">"
    body = body & " <Login>" & adminLogin & "</Login>"
    body = body & " <Password>" & adminPassword & "</Password>"
    body = body & " </CredentialsSoapHeader>"
    body = body & " </soap12:Header>"
    body = body & " <soap12:Body>"
    body = body & " <CreateHost"
    xmlns=""http://swsoft.com/webservices/sb/4.5/HostService"">"
    body = body & " <hostName>" & HostName & "</hostName>"
    body = body & " <hostAddress>" & HostAddress & "</hostAddress>"
    body = body & " <SMTPPort>" & SMTPPort & "</SMTPPort>"
    body = body & " </CreateHost >"
    body = body & " </soap12:Body>"
    body = body & " </soap12:Envelope>"

    Call xmlhttp.send(body)
    'Parse result
    Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
    XMLDOM.Load(xmlhttp.responseBody)
    Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
    if Not XMLNode Is Nothing then
        Dim XMLdata
        Set XMLdata = XMLNode.firstChild
        if Not XMLdata is Nothing then
            if XMLdata.nodeName = "soap:Fault" then

```

```
        response.Write "FAULT!<br><pre>" & XMLData.text & "</pre>"
        CreateHost=""
    else
        Set XMLData =
XMLDOM.SelectSingleNode("//CreateHostResponse/CreateHostResult/Id")
        CreateHost = XMLData.text
    end if
    else
        CreateHost=""
        response.Write "Error parsing SOAP response"
    end if
end if
End Function %>
```

Create Plan Sample

```
<%  
dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId  
dim error  
baseUrl="http://localhost:2006"  
adminLogin="admin"  
adminPassword="admin"  
PlanName = "My new host"  
PlanDescription = "192.0.2.12"  
PMpagesLimit = "100"  
PMpagesrootLevel = "10"  
PMpagesLevel = "3"  
PMsitesNumber = "5"  
PMaccountsNumber = "15"  
PMhostsNumber = "15"  
PisPersonal = "true"  
PisAnonymous = "false"  
PTrialLifeTime = "2"  
PTrialLifeType = "Days"  
PTemplatesID = "15"  
PPageSetsID = "1"  
PModulesID = "1"  
PHostsID = "2"  
PFamiliesID = "1"  
PDefaultFamilyID = "3"  
PisPublishSettingsEditable = "false"  
PisFTPPublishAvailable = "true"  
PisXcopyPublishAvailable = "false"  
PisVPSPublishAvailable = "false"  
PisAdditionalSiteContentAllowed = "false"  
PAdditionalSiteContent = ""  
PisUserManagementAllowed = "true"  
PisSiteManagementAllowed = "true"  
NewPlanID = CreatePlan(PlanName, PlanDescription, PMpagesLimit,  
PMpagesrootLevel, PMpagesLevel, PMsitesNumber, PMaccountsNumber,  
PMhostsNumber, PisPersonal, PisAnonymous, PTrialLifeTime,  
PTrialLifeType,  
PTemplatesID, PPageSetsID, PModulesID, PHostsID, PFamiliesID,  
PDefaultFamilyID, PisPublishSettingsEditable, PisFTPPublishAvailable,  
PisXcopyPublishAvailable, PisVPSPublishAvailable,  
PisAdditionalSiteContentAllowed, PAdditionalSiteContent,  
PisUserManagementAllowed, PisSiteManagementAllowed)  
Function CreatePlan(PlanName, PlanDescription, PMpagesLimit,  
PMpagesrootLevel, PMpagesLevel, PMsitesNumber, PMaccountsNumber,  
PMhostsNumber, PisPersonal, PisAnonymous, PTrialLifeTime,  
PTrialLifeType,  
PTemplatesID, PPageSetsID, PModulesID, PHostsID, PFamiliesID,  
PDefaultFamilyID, PisPublishSettingsEditable, PisFTPPublishAvailable,  
PisXcopyPublishAvailable, PisVPSPublishAvailable,  
PisAdditionalSiteContentAllowed, PAdditionalSiteContent,  
PisUserManagementAllowed, PisSiteManagementAllowed)  
    dim url, xmlhttp, XMLDOM, XMLNode  
    'Call web service using HTTP-POST  
    url = baseUrl & "/ServiceFacade/4.5/PlanWebService.asmx"  
    Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")  
    Call xmlhttp.Open("POST", url, False)
```

```

Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
dim body
body = "<?xml version=""1.0"" encoding=""utf-8""?>"
body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
body = body & "<soap12:Header>"
body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<CreatePlan
xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService""> "
body = body & "<name>" & PlanName & "</name>"
body = body & "<description>" & PlanDescription & "</description>"
body = body & "<maxPagesNumber>" & PMpagesLimit & "</maxPagesNumber>"
body = body & "<maxPagesRootLevel>" & PMpagesrootLevel &
"</maxPagesRootLevel>"
body = body & "<maxPagesLevel>" & PMpagesLevel & "</maxPagesLevel>"
body = body & "<maxSitesNumber>" & PmsitesNumber &
"</maxSitesNumber>"
body = body & "<maxAccountsNumber>" & PmaxaccountsNumber &
"</maxAccountsNumber>"
body = body & "<maxHostsNumber>" & PMhostsNumber &
"</maxHostsNumber>"
body = body & "<isPersonal>" & PisPersonal & "</isPersonal>"
body = body & "<isAnonymous>" & PisAnonymous & "</isAnonymous>"
body = body & "<trialLifeTime>" & PtrialLifeTime & "</trialLifeTime>"
body = body & "<trialLifeType>" & PtrialLifeType & "</trialLifeType>"
body = body & "<templatesIds> <string>" & PtemplatesID &
"</string></templatesIds>"
body = body & "<pagesetsIds><string>" & PpageSetsID &
"</string></pagesetsIds>"
body = body & "<modulesIds> <string>" & PmodulesID &
"</string></modulesIds>"
body = body & "<hostsIds> <string>" & PhostsID &
"</string></hostsIds>"
body = body & "<families> <string>" & PfamiliesID &
"</string></families>"
body = body & "<defaultFamily>" & PdefaultFamilyID &
"</defaultFamily>"
body = body & "<isPublishingSettingsEditable>" &
PisPublishSettingsEditable & "</isPublishingSettingsEditable>"
body = body & "<isFtpPublishAvailable>" & PisFTPPublishAvailable &
"</isFtpPublishAvailable>"
body = body & "<isXcopyPublishAvailable>" & PisXcopyPublishAvailable
& "</isXcopyPublishAvailable>"
body = body & "<isVpsPublishAvailable>" & PisVSPublishAvailable &
"</isVpsPublishAvailable>"
body = body & "<isAdditionalSiteContentAllowed>" &
PisAdditionalSiteContentAllowed & "</isAdditionalSiteContentAllowed>"
body = body & "<additionalSiteContent>" & PadditionalSiteContent &
"</additionalSiteContent>"
body = body & "<isUserManagementAllowed>" & PisUserManagementAllowed
& "</isUserManagementAllowed>"

```

```
body = body & "<isSiteManagementAllowed>" & PIsSiteManagementAllowed
& "</isSiteManagementAllowed>"
body = body & "</CreateHost >"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

Call xmlhttp.send(body)
'Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseBody)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
if Not XMLNode Is Nothing then
    Dim XMLdata
    Set XMLdata = XMLNode.firstChild
    if Not XMLdata is Nothing then
        if XMLdata.nodeName = "soap:Fault" then
            response.Write "FAULT!<br><pre>" & XMLdata.text & "</pre>"
            CreatePlan=""
        else
            Set XMLData =
XMLDOM.SelectSingleNode("//CreateHostResponse/CreatePlanResult/PlanId"
)
            CreatePlan = XMLData.text
        end if
    else
        CreatePlan=""
        response.Write "Error parsing SOAP response"
    end if
end if
End Function %>
```

Modifying Plan Sample

```
dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId
dim error
baseUrl="http://localhost:2006"
adminLogin="admin"
adminPassword="admin"
PlanId = "35b50410-da00-df8e-f6c4-1f0cb537a9a3"
PlanName = "My new host"
PlanDescription = "192.0.2.12"
PMpagesLimit = "100"
PMpagesrootLevel = "10"
PMpagesLevel = "3"
PMsitesNumber = "5"
PMaccountsNumber = "15"
PMhostsNumber = "15"
PTrialLifeTime = "2"
PTrialLifeType = "Days"
PTemplatesID = "15"
PPageSetsID = "1"
PModulesID = "1"
PHostsID = "2"
PFamiliesID = "1"
PDefaultFamilyID = "3"
PISPublishSettingsEditable = "false"
PISFTPPublishAvailable = "true"
PISXcopyPublishAvailable = "false"
PISVPSPublishAvailable = "false"
PISAdditionalSiteContentAllowed = "false"
PAdditionalSiteContent = ""
PISUserManagementAllowed = "true"
PISSiteManagementAllowed = "true"

UpdatePlan(PlanId, PlanName, PlanDescription, PMpagesLimit,
PMpagesrootLevel, PMpagesLevel, PMsitesNumber, PMaccountsNumber,
PMhostsNumber, PTrialLifeTime, PTrialLifeType,
PTemplatesID, PPageSetsID, PModulesID, PHostsID, PFamiliesID,
PDefaultFamilyID, PISPublishSettingsEditable, PISFTPPublishAvailable,
PISXcopyPublishAvailable, PISVPSPublishAvailable,
PISAdditionalSiteContentAllowed, PAdditionalSiteContent,
PISUserManagementAllowed, PISSiteManagementAllowed)
Function UpdatePlan(PlanId, PlanName, PlanDescription, PMpagesLimit,
PMpagesrootLevel, PMpagesLevel, PMsitesNumber, PMaccountsNumber,
PMhostsNumber, PTrialLifeTime, PTrialLifeType,
PTemplatesID, PPageSetsID, PModulesID, PHostsID, PFamiliesID,
PDefaultFamilyID, PISPublishSettingsEditable, PISFTPPublishAvailable,
PISXcopyPublishAvailable, PISVPSPublishAvailable,
PISAdditionalSiteContentAllowed, PAdditionalSiteContent,
PISUserManagementAllowed, PISSiteManagementAllowed)
    dim url, xmlhttp, XMLDOM, XMLNode
    ' Call web service using HTTP-POST
    url = baseUrl & "/ServiceFacade/4.5/PlanWebService.asmx"
    Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
    Call xmlhttp.Open("POST", url, False)
```

```

Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
dim body
body = "<?xml version=""1.0"" encoding=""utf-8""?>"
body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
body = body & "<soap12:Header>"
body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<CreatePlan
xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService""> "
body = body & "<planId>" & PlanId & "</planId>"
body = body & "<name>" & PlanName & "</name>"
body = body & "<description>" & PlanDescription & "</description>"
body = body & "<maxPagesNumber>" & PMpagesLimit & "</maxPagesNumber>"
body = body & "<maxPagesRootLevel>" & PMpagesrootLevel &
"</maxPagesRootLevel>"
body = body & "<maxPagesLevel>" & PMpagesLevel & "</maxPagesLevel>"
body = body & "<maxSitesNumber>" & PMsitesNumber &
"</maxSitesNumber>"
body = body & "<maxAccountsNumber>" & PMaccountsNumber &
"</maxAccountsNumber>"
body = body & "<maxHostsNumber>" & PMhostsNumber &
"</maxHostsNumber>"
body = body & "<trialLifeTime>" & PTrialLifeTime & "</trialLifeTime>"
body = body & "<trialLifeType>" & PTrialLifeType & "</trialLifeType>"
body = body & "<templatesIds> <string>" & PTemplatesID &
"</string></templatesIds>"
body = body & "<pagesetsIds><string>" & PPageSetsID &
"</string></pagesetsIds>"
body = body & "<modulesIds> <string>" & PModulesID &
"</string></modulesIds>"
body = body & "<hostsIds> <string>" & PHostsID &
"</string></hostsIds>"
body = body & "<families> <string>" & PFamiliesID &
"</string></families>"
body = body & "<defaultFamily>" & PDefaultFamilyID &
"</defaultFamily>"
body = body & "<isPublishingSettingsEditable>" &
PisPublishSettingsEditable & "</isPublishingSettingsEditable>"
body = body & "<isFtpPublishAvailable>" & PIsFTTPublishAvailable &
"</isFtpPublishAvailable>"
body = body & "<isXcopyPublishAvailable>" & PIsXcopyPublishAvailable
& "</isXcopyPublishAvailable>"
body = body & "<isVpsPublishAvailable>" & PIsVSPublishAvailable &
"</isVpsPublishAvailable>"
body = body & "<isAdditionalSiteContentAllowed>" &
PisAdditionalSiteContentAllowed & "</isAdditionalSiteContentAllowed>"
body = body & "<additionalSiteContent>" & PAdditionalSiteContent &
"</additionalSiteContent>"
body = body & "<isUserManagementAllowed>" & PIsUserManagementAllowed
& "</isUserManagementAllowed>"

```

```
body = body & "<isSiteManagementAllowed>" & PIsSiteManagementAllowed
& "</isSiteManagementAllowed>"
body = body & "</CreateHost >"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

Call xmlhttp.send(body)
'Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseBody)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
if Not XMLNode Is Nothing then
    Dim XMLdata
    Set XMLdata = XMLNode.firstChild
    if Not XMLdata is Nothing then
        if XMLdata.nodeName = "soap:Fault" then
            Err.Raise 100, "UpdatePlan", XMLdata.Text
        end if
    else
        Err.Raise 100, "UpdatePlan", "Error parsing SOAP response"
    end if
end if
End function
```

Business Scenarios Code

```

<%@ language="VBScript" %>
<html>
<head>
    <title>Creating Site Builder Account</title>
</head>
<body>
<%
dim baseUrl, adminLogin, adminPassword, newAccountId, newSiteId
dim error
baseUrl="http://localhost:2006"
adminLogin="admin"
adminPassword="admin"

newSiteId=Request("siteId")
if Request("action") = "create" then
    newAccountId=CreateAccount(Request("accountName"),
Request("accountPassword"), Request("firstName"), Request("lastName"),
Request("eMail"), Request("plan"))
    if newAccountId <> "" then
        if newSiteId<>"" then
            if true=TakeOwnership(newAccountId, newSiteId,
Request("siteAlias")) then
                RenderSuccess Request("accountName"),
Request("accountPassword"), newSiteId
            end if
        else
            newSiteId=CreateSite(newAccountId,Request("siteAlias"))
            if newSiteId<>"" then
                end if
            end if
        end if
    else
        response.Write RenderForm
    end if
%>
</body>
</html>
<%
Function CreateAccount (accountName, accountPassword, firstName,
lastName, eMail, planId)
dim url, xmlhttp, XMLDOM, XMLNode
' Call web service using HTTP-POST
url = baseUrl & "/ServiceFacade/4.5/AccountWebService.asmx"
Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
Call xmlhttp.Open("POST", url, False)
Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
dim body
body = "<?xml version=""1.0"" encoding=""utf-8""?>"
body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
body = body & "<soap12:Header>"

```

```

body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/AccountService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<CreateAccount
xmlns=""http://swsoft.com/webservices/sb/4.5/AccountService"">"
body = body & "<ownerAccountId></ownerAccountId>"
body = body & "<username>" & accountName & "</username>"
body = body & "<password>" & accountPassword & "</password>"
body = body & "<firstName>" & firstName & "</firstName>"
body = body & "<lastName>" & lastName & "</lastName>"
body = body & "<email>" & eMail & "</email>"
body = body & "<role>SiteOwner</role>"
body = body & "<planId>" & planId & "</planId>"
body = body &
"<changePasswordAllowed>true</changePasswordAllowed>"
body = body & "</CreateAccount>"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

Call xmlhttp.send(body)
' Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseBody)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
CreateAccount = XMLDOM.xml 'XMLNode.xml
if Not XMLNode Is Nothing then
    Dim XMLdata
    Set XMLdata = XMLNode.firstChild
    if Not XMLdata is Nothing then
        if XMLdata.nodeName = "soap:Fault" then
            response.Write "FAULT!<br><pre>" & XMLdata.text & "</pre>"
            CreateAccount=""
        else
            Set XMLData =
XMLDOM.SelectSingleNode("//CreateAccountResponse/UserAccount/AccountId
")
            CreateAccount = XMLData.text
            end if
        else
            CreateAccount=""
            response.Write "Error parsing SOAP response"
            end if
        end if
    end if
End Function
Function TakeOwnership(accountId, siteId, siteAlias) dim url, xmlhttp,
XMLDOM, XMLNode ' Call web service using HTTP-POST url = baseUrl &
"/ServiceFacade/4.5/SiteWebService.aspx" Set xmlhttp =
Server.CreateObject("MSXML2.ServerXMLHTTP") Call xmlhttp.Open("POST",
url, False)
Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
dim body
body = "<?xml version=""1.0"" encoding=""utf-8""?>"

```

```

body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
body = body & "<soap12:Header>"
body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/SiteService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<TakeOwnershipOfAnonymousSite
xmlns=""http://swsoft.com/webservices/sb/4.5/SiteService"">"
body = body & "<siteId>" & siteId & "</siteId>"
body = body & "<ownerId>" & accountId & "</ownerId>"
body = body & "<alias>" & siteAlias & "</alias>"
body = body & "</TakeOwnershipOfAnonymousSite>"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

```

```

Call xmlhttp.send(body)
' Parse result
TakeOwnerShip=false
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseBody)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
if Not XMLNode Is Nothing then
    Dim XMLdata
    Set XMLdata = XMLNode.firstChild
    if Not XMLdata is Nothing then
        if XMLdata.nodeName = "soap:Fault" then
            response.Write "FAULT!<br><pre>" & XMLdata.text & "</pre>"
        else
            TakeOwnerShip=true
        end if
    else
        response.Write "Error parsing SOAP response"
    end if
end if
End Function

```

```

Function ListPlans()
    dim url, xmlhttp, XMLDOM, XMLNodeList, x , str
    ' Call web service using HTTP-POST url = baseUrl &
    "/ServiceFacade/4.5/PlanWebService.aspx" Set xmlhttp =
    Server.CreateObject("MSXML2.ServerXMLHTTP") Call xmlhttp.Open("POST",
    url, False)
    Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
    charset=utf-8")
    dim body
    body = "<?xml version=""1.0"" encoding=""utf-8""?>"
    body = body & "<soap12:Envelope
    xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
    xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
    xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
    body = body & "<soap12:Header>"

```

```

body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<GetAvailablePlans
xmlns=""http://swsoft.com/webservices/sb/4.5/PlanService"">"
body = body & "<accountId></accountId>"
body = body & "</GetAvailablePlans>"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"
Call xmlhttp.send(body)
' Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseBody)
Set XMLNodeList = XMLDOM.selectNodes("//PlanValue")

For x = 1 To XMLNodeList.length
    str = str & "<option value="" & XMLNodeList.Item(x -
1).selectSingleNode("PlanId").text & """" & XMLNodeList.Item(x -
1).selectSingleNode("Name").text
Next
ListPlans=str
End Function

Function CreateSite(ownerId, siteAlias)
dim url, xmlhttp, XMLDOM, XMLNode
' Call web service using HTTP-POST
url = baseUrl & "/ServiceFacade/4.5/SiteWebService.aspx"
Set xmlhttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
Call xmlhttp.Open("POST", url, False)
Call xmlhttp.setRequestHeader("Content-Type", "application/soap+xml;
charset=utf-8")
dim body
body = "<?xml version=""1.0"" encoding=""utf-8""?>"
body = body & "<soap12:Envelope
xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
xmlns:soap12=""http://www.w3.org/2003/05/soap-envelope"">"
body = body & "<soap12:Header>"
body = body & "<CredentialsSoapHeader
xmlns=""http://swsoft.com/webservices/sb/4.5/SiteService"">"
body = body & "<Login>" & adminLogin & "</Login>"
body = body & "<Password>" & adminPassword & "</Password>"
body = body & "</CredentialsSoapHeader>"
body = body & "</soap12:Header>"
body = body & "<soap12:Body>"
body = body & "<CreateSite
xmlns=""http://swsoft.com/webservices/sb/4.5/SiteService"">"
body = body & "<siteType>Regular</siteType>"
body = body & "<siteAlias>" & siteAlias & "</siteAlias>"
body = body & "<ownerId>" & ownerId & "</ownerId>"
body = body & "</CreateSite >"
body = body & "</soap12:Body>"
body = body & "</soap12:Envelope>"

Call xmlhttp.send(body)

```

```

' Parse result
Set XMLDOM = Server.CreateObject("Microsoft.XMLDOM")
XMLDOM.Load(xmlhttp.responseText)
Set XMLNode = XMLDOM.SelectSingleNode("//soap:Body")
if Not XMLNode Is Nothing then
    Dim XMLdata
    Set XMLdata = XMLNode.firstChild
    if Not XMLdata is Nothing then
        if XMLdata.nodeName = "soap:Fault" then
            response.Write "FAULT!<br><pre>" & XMLdata.text & "</pre>"
            CreateSite=""
        else
            Set XMLData =
XMLDOM.SelectSingleNode("//CreateSiteResponse/CreateSiteResult/Id")
            CreateSite = XMLData.text
        end if
    else
        CreateSite=""
        response.Write "Error parsing SOAP response"
    end if
end if
End Function

%>
<%
function RenderSuccess (login, password, siteId)
    dim adminUrl, wizardUrl
    adminUrl=baseUrl & "/ExternalLogin.ashx?Login=" & login &
"&Password=" & password
    wizardUrl=adminUrl & "&SiteID=" & siteId
    adminUrl=adminUrl & "&ShowAdmin=true"
%>
You have successfully registered. Please <a
href="<%=wizardUrl%>">click here</a> to edit your site at Wizard or <a
href="<%=adminUrl%>">click here</a> to enter Plesk Sitebuilder
siteowner's panel.
<%
End function
%>

<%
function RenderForm
%>
<h3>
    SAMPLE CODE FOR PLESK SITEBUILDER INTEGRATION</h3>
Copyright&copy; 2008 by <a href="parallels.com">Parallels</a><br>
NOTE: This code is provided as is, without any warranty, either
express or implied
Please do not use this sample in a production environment WARNING!
Administrator
credentials are sent in clear text. It is by default assumed that this
script will
be run on the same machine as the Plesk Sitebuilder installation. In
case you want to
administer the Plesk Sitebuilder remotely using this Web Service API,
make sure to secure
the communication (for example, by using HTTPS protocol instead of
HTTP).

```

```

<form method="post">
  <input type="hidden" name="siteId" value="<% response.Write
newSiteId%>" />
  <table>
    <tr>
      <td>
        Account Name:
      </td>
      <td>
        <input type="text" name="accountName"
value="TestUserFromASP"></td>
    </tr>
    <tr>
      <td>
        Account Password:
      </td>
      <td>
        <input type="text" name="accountPassword"
value="111111"></td>
    </tr>
    <tr>
      <td>
        First Name:
      </td>
      <td>
        <input type="text" name="firstName"
value="Bob"></td>
    </tr>
    <tr>
      <td>
        Last Name:
      </td>
      <td>
        <input type="text" name="lastName"
value="Smith"></td>
    </tr>
    <tr>
      <td>
        E-Mail:</td>
      <td>
        <input type="text" name="eMail"
value="none@nowhere.org"></td>
    </tr>
    <tr>
      <td>
        Plan:
      </td>
      <td>
        <select name="plan">
          <%response.Write ListPlans() %>
        </select>
      </td>
    </tr>
    <tr><td colspan="2"><hr /></td></tr>
    <tr>
      <td>
        Site alias:
      </td>
      <td>

```

```
                <input type="text" name="siteAlias"
value="Test Site">
            </td>
        </tr>
        <tr>
            <td colspan="2">
                <input type="submit" name="action"
value="create"></td>
            </tr>
        </table>
</form>
<%
End function
%>
```