SWSOFT™ PLESK™

SWsoft

Plesk Application Vault

Developer's Guide

Plesk 8.1 for Unix

SWSOFT™

(c) 1999-2007

Contents

C H A P T E R   1

Preface

In this section:

About This Guide

This guide describes Application Vault module used by Plesk to manage web applications. The guide contains details on the module's structure and functionality. It also provides you with information on how to use Application Vault, how to create a 'Site Application' package and install it to Plesk.

Who Should Read This Guide

This guide is useful for developers writing web applications that Plesk users may use on their sites. Also, it targets Plesk resellers who would like to supply customized versions of Plesk with extended sets of web applications.

Typographical Conventions

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 **QoS** tab.
	Titles of chapters, sections, and subsections.	Read the **Basic Administration** chapter.
Italics	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 *wildcard character* search.
`Monospace`	The names of style sheet selectors, files and directories, and CSS fragments.	The license file is located in the `httpdocs/common/license` directory.
Preformatted	On-screen computer output in your command-line sessions; source code in XML, C++, or other programming languages.	`# ls –al /files` `total 14470`
Preformatted Bold	What you type, contrasted with on-screen computer output.	`# cd /root/rpms/php`

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 to userdocs@swsoft.com. 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.

C H A P T E R 2

Concepts

This chapter presents an overview of the most common concepts related to Plesk Application Vault . It explains what Application Vault is designed for, how it is structured, both logically and physically, and what happens inside Plesk when one performs various operations on Application Vault.

In this section:

Overview

Beginning with version 7 and higher, Plesk ships with a set of installation packages of third-party applications that can be provided to the customer on demand. These web applications can be deployed on domains and then managed seamlessly from within Plesk Control Panel. Besides, Plesk provides its customers with means of deploying their own web applications on domains using the same mechanism. This mechanism is implemented in Application Vault, a module of Plesk.

Purpose

Application Vault presents a full-fledged repository of web applications with extended management capabilities. Application Vault stores installation packages of web applications on a server running Plesk v. 7.0 or later and has on-board mechanisms to install, configure, and delete these applications on the domain. These mechanisms perform the bulk of work, hiding the details of these tricky processes from the user.

The installation packages stored in Application Vault are arranged in the sets. These sets are then provided by Plesk suppliers along with Plesk. A typical set includes such applications as:

- File managers & Statistics systems (AutoIndex)
- Chats (gtchat, etc.), forums (phpBB, etc.), blog systems (pLog, WordPress, b2evolution, etc.)
- Photo galleries (nGallery, etc.)
- Web mail clients (Uebimiau, etc.)
- e-commerce systems (osCommerce)
- CRM systems (TUTOS, xrms, etc.)
- Content management systems (Drupal, Mambo, PostNuke, Typo, etc.)
- others.

Where Applicable

Application Vault ships with Plesk beginning with version 7.0. This module is designed as an optional extension of Plesk functionality. To activate it, you need to buy a licence. Application Vault has a Plesk-styled graphical user interface. If activated, Application Vault and all its operations are accessible via Plesk Control Panel.

Target Audience

- Application Vault was designed in order to provide a customer with a set of useful web applications and a simple mechanism of their deployment on a domain.
- Also, Application Vault targets Plesk resellers who would like to implement their own applications, or to use third-party ones, and to supply customized versions of Plesk with extended sets of web applications containing these ones.

Application Vault was designed to standardize the structure of application projects, which would guarantee their smooth installing/initial configuring/uninstalling by means of Plesk.

Application Vault Architecture

Before proceeding to the study of operations on application packages, look how Application Vault is arranged both logically and physically.

Logical Structure

Logically, Application Vault presents a distributed manager system designed to manage web applications on Plesk. The managing mechanism covers both web applications stored on the server side and ones already deployed on the domain side.
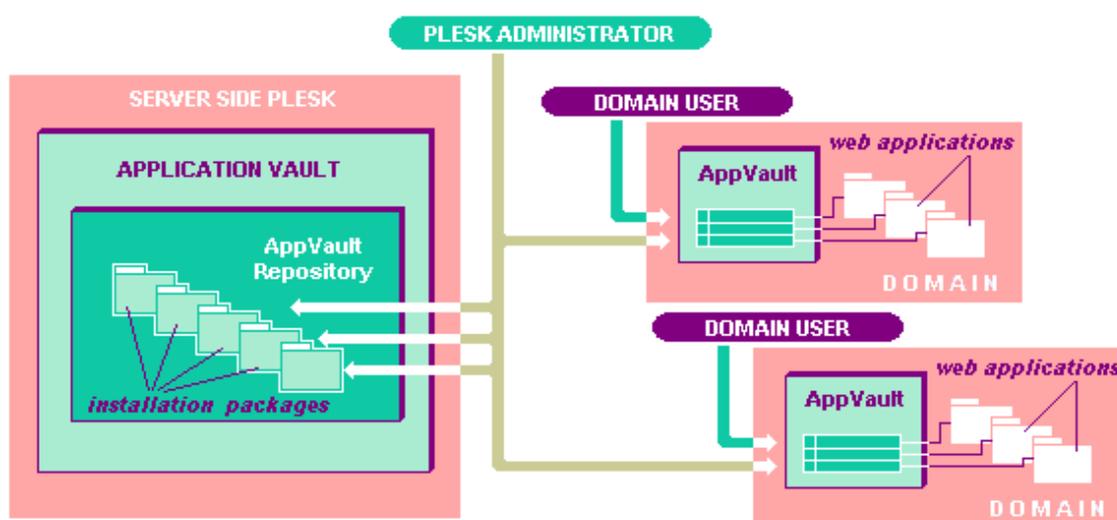
Figure 1: The logical structure of Application Vault

The server-side Application Vault is designed to store web applications packed into installation packages and entirely ready for the deployment on the domain side. Application Vault (AV) Repository allows the following operations on it:

- you can add packages to AV Repository (on page 20),
- you can edit package attributes (on page 21) (e.g. free/commercial),
- you can delete packages from AV Repository (on page 22),
- you can deploy a package on the specified domain (on page 23).

Note: In the current implementation, upgrading packages in its pure sense is not supported in Plesk. If a later (or a different) version of a web application is added to AV repository, it will be put beside the existing package.

All operations on the server-side Application Vault lay within Plesk Administrator's area of responsibility.

The domain-side Application Vault is represented by the *list of deployed applications* registered on a domain. The following operations over this list are allowed from Plesk Control Panel:

- an application can be added to the list (on page 23) (which happens when installing the application to a domain);
- you can reconfige a web application (on page 25) available in this list;
- an application can be deleted from the list (on page 26) (which entails uninstalling the application from the domain).

Note: Upgrading packages is not supported in the current implementation of Plesk. To fill the gap, Application Vault allows the deployment of as many similar applications on the same domain as necessary.

All operations on the domain-side Application Vault are accessible both for Domain User and Plesk Administrator.

Physical structure

Though the logical structure (on page 9) of Application Vault is presented by server-side and domain-side divisions, in fact, Application Vault is fully located on the server running Plesk. Its physical structure includes the following units:

- *AV Repository* stores physical application packages;
- *persistent data* describes the repository and successful installations. The persistent data is also used to display and manage the contents of Application Vault both on the server and on domains.

The Application Vault (AV) Repository is arranged as a hierarchy of folders being a part of Plesk folder system. The multi-level structure of AV Repository looks as follows:

Level 0	`<plesk_root_dir>/`
📂	This level matches the root directory of Plesk. Here `<plesk_root_dir>` stands for the fully qualified path of a directory where Plesk Server Administrator is installed. In Unix, this path is normally as follows: `/usr/local/psa`, in Debian it is `/opt/psa`.

	Level 1	`var/cgitory/`
	📂	This is the level of the root directory of Plesk Application Vault.

		Level 2	`<app_name>-<app_version>-<app_release>/`
		📂	This level shows the contents of AV root directory. In case there is not an application package stored in the AV repository, the `<plesk_root_dir>/var/cgitory` folder is empty. Otherwise it stores one to many folders, each created for a single package.

			Level 3	`scripts/, forms/, apps/, uninstall/, docs/, info/, screenshots/`
			📂	This level shows the contents of an installation package stored in the AV repository. It contains 7 folders, some of which may be missing. The detailed description of the package structure can be found here (on page 27).

Installation Package Naming Convention

The name of the application package folder should be formed according to the following rule:

```
<application name>-<product version>-<release version>
```

E.g.

```
phpAds-2.0.5-7512
phpBB-2.0.17-7118
phpBook-1.50-8011
```

This format allows the storage of different product and release versions of the same application in AV Repository.

The persistent data on the contents of the stored packages and successful installations is stored in Plesk database.

Application Vault Operations

Besides storing installation packages, Application Vault solves two large tasks, that is:

- it manages all operations on installation packages stored in AV Repository,
- it manages all operations on the web applications it has deployed on domains.

To perform these tasks, Application Vault has two built-in managers:

- *SiteAppPackageManager* is designed to manage the installation package flow (on page 18) within AV Repository,
- *SiteAppManager* implements the deployment/reconfiguring of web applications on the domain side.

AV Repository Operations

Application Vault (AV) Repository is designed to store web applications on the form of installation packages. AV Repository is managed by *SiteAppPackageManager* that supplements AV Repository with new packages, allows changing the package status, and removes unnecessary packages from AV Repository.

Uploading an Installation Package to AV Repository

An installation package can get to AV Repository with the help of SiteAppPackageManager only, whose chief tasks are: to verify the structure of a distribution package incoming to AV Repository, to initiate unpacking of its contents to the repository, and to register the unpacked contents (an installation package) in AV repository.

When adding a new installation package to the repository, the first step is the physical upload of the RPM/SH/DEB package to the server running Plesk. This can be done via a special form provided by Plesk, or by other Plesk-non-specific means, e.g. via CLI. An RPM/SH/DEB package can be considered as a wrapper package that contains an installation package itself plus (and it is very important!) the folder structure that specifies where the installation package will be located in AV Repository. This folder structure will be fully copied to the repository by internal means of the "wrapper" package.

Having copied the RPM/SH/DEB package to the server, SiteAppPackageManager checks whether this file format is supported and whether SiteAppPackageManager should proceed to its processing.

Then SiteAppPackageManager initiates unpacking of the "wrapper" package to the repository: the folder structure and the contents of the "wrapper" package are copied to the root folder of Application Vault.

When unpacked, the new installation package needs to be registered in the repository. First SiteAppPackageManager verifies whether the new installation package contains the `info.xml` file in its `/info` folder, and whether the data shown in this file (the application name, its product and release versions) matches the name of the installation package. If the `info.xml` file is not found or the checkup fails, the upload rolls back (i.e. the application folder and all its subfolders are deleted from the repository). If the file is found, the database is populated with the application name, its product and release versions, its path, etc.

Changing the Package Status in the Repository

The procedure of changing the status of an installation package is very simple: as soon as a request for this procedure is passed via Plesk Control Panel, SiteAppPackageManager just changes the relevant 'status' flag in Plesk database.

Deleting an Installation Package from AV Repository

Having received a command to delete an installation package from the repository, SiteAppPackageManager looks for the **uninstall** script in the `/uninstall` folder and executes it if any available. Nevertheless, the **uninstall** script is optional and, if present, is not obliged to delete an installation package from AV Repository as using the RPM/SH/DEB package implies that the package contents will be deleted by system facilities.

Domain-Side Operations

SiteAppManager is the logical unit responsible for the deployment and reconfiguring of web applications on the domain side (deleting applications from a domain refers to the operations directly managed by Plesk).

The applications stored in AV Repository may rather differ in the way they are deployed on a domain. If all these particular cases were the care of *SiteAppManager*, it would present an inflated and dummy tool begging for more and more updates. On the contrary, *SiteAppManager* presents a very simple automaton responsible only for the most trivial functions. The deployment and reconfiguring logic lays within the installation package in the form of PHP forms, handlers, and scripts. The unified folder structure of installation packages allows *SiteAppManager* to access this logic and to execute it.

Installation routine

This topic describes how SiteAppManager installs packages on the domain. First of all, Plesk Control Panel displays a package stored in AV Repository only if the `info.xml` file (the application's description) is found in the `/info` folder of the installation package.
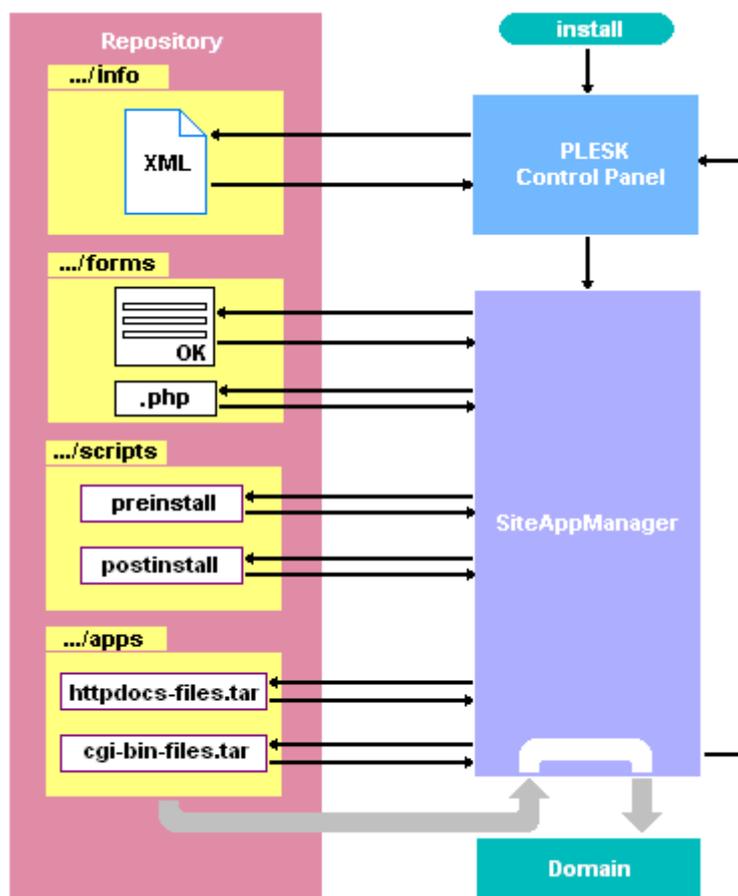
Figure 2: Installing a web application on the domain

Later on, Plesk Control Panel receives the user's command to install the web application to a certain domain and passes control to SiteAppManager. The task of SiteAppManager is to take preparatory steps for deployment as well as to deploy the application to the domain.

During the installation routine, SiteAppManager interacts with users via forms, and event handling is arranged as a two-level process. It is proposed that the handler (a PHP file) reads data from the fields of a form and verifies it as well, but low-level operations (i.e. creating folders on the domain, adding records to the database, setting user permissions, and so on) are the care of scripts executed by the operating system directly.

1. First SiteAppManager searches the `/scripts` folder in the installation package, and if it finds the **preinstall** script in it, then the script executes and the result is returned to SiteAppManager . E.g. the **preinstall** script may be handy if one needs to prepare the hierarchy of folders on a domain right before installing an application.

2. Then SiteAppManager searches for the `/forms` folder across the installation package, chooses the appropriate input forms in it, displays them to the user one after another, and waits for the user's reaction (for the pressure of certain buttons).

3. As soon as the expected event (the pressure of the **OK** button) occurs, SiteAppManager searches for a matching handler in the same `/forms` folder and passes control to it if succeeds.

4. Then the application's archive files are unpacked to the domain.

5. As soon as the unpack routine finishes, SiteAppManager checks the `/scripts` folder again. If it discovers the **postinstall** script in it, then the configuration parameters read from the forms are passed in to this script, after which the script executes (some records are added to the database, user permissions are set, the application's configuration files are modified) and SiteAppManager gets the result of its execution.

Finally, the web application is registered on the domain (SiteAppManager adds proper records to Plesk database) and appears in the AV List that displays all web applications deployed on the domain.

Reconfiguration routine

The reconfiguration routine applied to deployed web application requires the following actions from SiteAppManager:

1. SiteAppManager searches for the `/forms` folder across the installation package, picks out the parameter input form in it, displays it to the user, and waits for the reaction (for **OK** button pressure).

Note: If the requires form is missing, the configuration procedure fails.

2. Once the expected event has occurred, SiteAppManager searches for a handler of the same name across the `/forms` folder and passes control to it. The handler reads new configuration parameters from the form and passes them to SiteAppManager.

3. SiteAppManager checks the `/scripts` folder for the **reconfigure** script, passes new parameters to it, after which the script executes (some records are added to the database, the application's configuration files are modified) and returns the execution result to SiteAppManager.

Deletion routine

The deletion procedure is not managed by SiteAppManager. If Domain User decides to delete an application from the domain, then Plesk takes control of this procedure. Nevertheless, while the deletion performs, it is possible to perform some extra operations not proposed by Plesk (e.g. logging). For this to happen, it is enough to put the **preuninstall** or **postuninstall** script to the installation package. These scripts will be called by Plesk before and after the physical deletion of application files respectively.

CHAPTER 3

Using Application Vault

In this section:

Web Application Flow

A web application selected for use in Plesk passes through several transformation stages during its lifetime in Plesk. These stages are as follows.
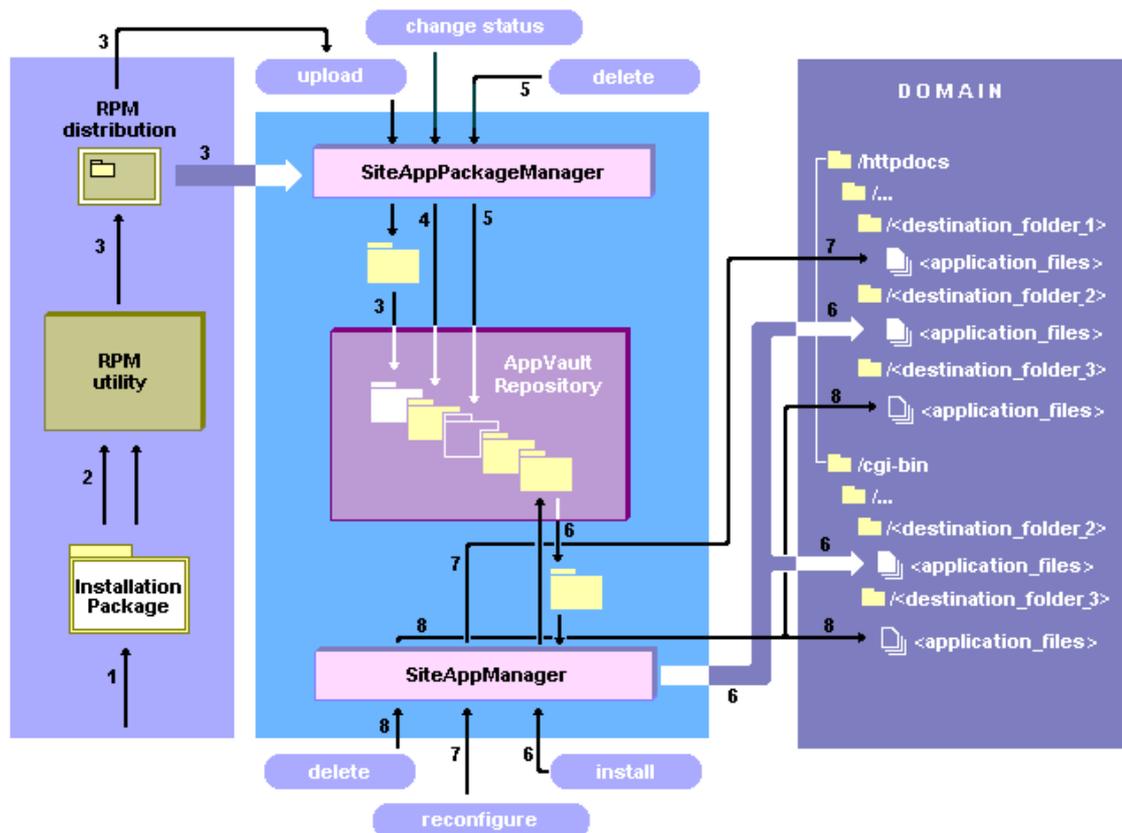
Figure 3: The main stages of the web application's life

Stage 1. The first stage lasts from the moment a web application is selected for Application Vault till it has got to AV Repository. This stage includes *creating an installation package* (1), *wrapping it into a distribution package* (2), and *uploading this distribution* to AV Repository (3).

At this stage, a web application turns into an *installation package* which serves as a minimal indivisible unit Application Vault can operate. To look at the detailed description of the installation package structure, open the *Installation Package* (on page 27) section of *Reference*. Later on, the installation package transforms into a *distribution package. For instance,* this can be done using the RPM utility. But this 'wrapper' package is only necessary to push the installation package into Application Vault. Once the distribution package is got to AV Repository, it is unwrapped, and the repository gets the installation package for storage.

Stage 2. The second stage refers to the time the web application is stored in AV Repository in the form of an installation package.

This stage comes to end when the installation package is *deleted from AV Repository* (5). Until this has happened, the installation package can experience as many *installations to domains* (6) as necessary. A web application can be installed on a domain on a commercial basis or free of charge. To switch between these states, one can apply the *'change status' operation* to the installation package (4). Installing a web application on a domain does not affect the installation package as transferred to the selected domain is just a copy, not the package itself. Once a web application is installed on a domain, it begins to live its own life, and here the next stage begins.

Stage 3. This stage of the web application's life goes on the domain the application is deployed on. Such an application presents a copy of the installation package stored in AV Repository. Thus, all operations performed against this copy do not affect the original. E.g. one can install a certain application on the same domain as many times as wanted, and every time the new installation will be entirely identical to the previous one, provided the source installation package remains unchanged. During its lifetime on the domain, an installation can experience an unlimited number of *reconfigurations* (7). The application 'dies' at the moment it is *deleted from the domain* (8).

How to Upload an Installation Package to AV Repository

Adding an installation package to AV Repository implies uploading the files and folders of this package to the `/cgitory` folder and registering the new installation package in AV Repository. This can be done via Plesk Control Panel or by other means not related to Plesk, e.g. via CLI provided by Unix-specific utilities. Uploading an installation package to AV Repository is fully automated if Plesk Administrator decides in favor of Plesk Control Panel, so this approach is recommended.

No matter which way is chosen to upload a web application's installation package, the first thing to do is to create an RPM/SH/DEB distribution package and to locate it anywhere, either locally or in the network, so that is can be accessed during the upload procedure. The detailed description of how to create an installation package and wrap it into a distribution package can be seen in the **How to Build an Installation Package** (on page 33) tutorial of this documentation.

Uploading an installation package via Plesk Control Panel

1   In **Plesk Control Panel**, click on the **Server** tab in the navigation pane and get to the **Server Administration** page.

2   In the **Services** section of this page, open the **Application Vault** tool by clicking on the relevant icon. You will get to the **Application Vault** page.

3   In the **Tools** section, click on the **Add New Application Package** button. You will get to the **Add new site application package** page.

4   Click the **Browse** button and select an RPM/SH/DEB package you wish to upload to Plesk.

Once this is done, the RPM/SH/DEB package is uploaded to the web server, and the selected RPM/SH/DEB package extracts itself to the application folder specially created within the AV Repository (`<plesk_root_dir>/var/cgitory`).

Note: If an installation package wrapped by the selected RPM/SH/DEB package has been formed incorrectly (the rules of building its file and folder structure have been violated, or the `info.xml` file contains some wrong information, etc.), then Plesk will fail to register the unpacked installation package in Plesk database. It will display an error message instead, and the upload will be rolled back.

If the upload has passed through all its steps successfully, **Plesk Control Panel** will update the **Application Vault** page and the new installation package will be displayed in the list of installation packages at the bottom of the page.

Uploading an installation package via Command Line Interface

The other way to upload an installation package to Plesk is to use the SSH utility that ships with UNIX. This feature is supported to allow the upload of installation packages by programmatic means.

1   Copy the source RPM/SH/DEB package to any directory located on the web server running Plesk.

2   From **SSH** console, set 'read/write/execute' permissions for this RPM/SH/DEB file using the following command (the example shows how to do this for the SH file named `package.sh`):

3   `Chmod 755 package.sh`

4   From **SSH** console, trigger the package to start (the example demonstrates this command for files `package.sh`, `package.rpm`, and `package.deb`, respectively):

```
./package.sh
rpm  -Uhv  package.rpm
dpkg  -i  package.deb
```

After the package has extracted its contents to AV Repository, the package is still unregistered in Plesk database. To register it, click on the **Refresh** button located on the **Application Vault** page (**Server**->**Application Vault**). Once this is done, the **Application Vault** page is updated and the new installation package is displayed in the list of installation packages at the bottom of the page.

How to Change the Status of an Installation Package

An installation package is assigned one of two possible 'access level' values: *free* or *commercial*. By default, all installation packages uploaded to AV Repository have the *free* status, which means that they can be installed to a domain *free of charge*.

To change the status of the application, Plesk Administrator should proceed through the following steps:

1   In **Plesk Control Panel**, click on the **Server** tab in the navigation pane and get to the **Server Administration** page.

2   In the **Services** section of this page, open the **Application Vault** tool by clicking on the relevant icon. You will get to the **Application Vault** page that will display the list of installation packages (AV Repository) at the bottom.

There are two ways how to change the status of an application. You can just switch between two states by clicking on the relevant icon against the target installation package in the list. Or you can do the same from the **Site Application Package Information** page related to the package:

3   On the **Application Vault** page, click on the application whose status you wish to change.

4   On the **Site Application Package Information** page, choose between two states in the **Access Level** drop-down list and click **OK**.

Once this is done, **Plesk Control Panel** will apply the new setting and display the updated **Application Vault** page.

How to Delete an Installation Package from AV Repository

Deleting an installation package from AV Repository implies removing all records from Plesk database that are associated with this package, and then removing all files and folders of the package physically from the server-side folder of Application Vault (`/cgitory`).

Note: If the package being deleted has installations on domains, the deployed web applications will go on function smoothly after the package is deleted from AV Repository. The deletion of such applications from the domain won't encounter any problems, but Plesk Administrator should take into account that reconfiguring these applications after the deletion of the 'source' package will be locked.

To delete an installation package from AV Repository, Plesk Administrator should do the following:

1   In **Plesk Control Panel**, click on the **Server** tab in the navigation pane and get to the **Server Administration** page.

2   In the **Services** section of this page, open the **Application Vault** tool by clicking on the relevant icon.

3   In the list of installation packages at the bottom of the **Application Vault** page, check an application (or several ones) to delete.

4   Click on the **Remove Selected** button located in the right upper corner of the installation package list.

5   After you get to the **Removal Confirmation** page, check the **Confirm Removal** checkbox and click **OK**.

Once the delete operation has finished, you will get back to the **Application Vault** page displaying the updated installation package list.

How to Deploy an Application to a Domain

The contents of AV Repository is visible both on the server-side level (to Plesk Administrator) and on the level of a domain (to Domain User). Both these users are allowed to install applications from AV Repository on the domain side: Plesk Administrator can do it on any domain located on the server, while Domain User can do it on his own domain only.

If you are a Domain User, proceed through the following steps:

1   On the navigation pane located at the left-hand area of **Plesk Control Panel**, choose the **Home** link.

2   In the **Hosting** section of the **Domain** page, click on the **Application Vault** tool. You will get to the **Installed applications** page. At the bottom, this page will display the list of web applications installed on the domain (if there is not an application installed, the list is empty).

3   In the **Tools** section of this page, click on the **Add New Application** button. You will get to AV Repository (the **Installation: select an application to be installed** page).

> **Note:** The same page is accessible directly from **Desktop (Plesk Control Panel -> Desktop->Tools** section**->Install a site application** button).

4   If AV Repository is not empty, the **Installation: ...** page will display the list of installation packages that can be deployed on the domain side. Check the application you wish to install on your domain and click the install button located in the right upper corner of the list. You will get to the **Site application installation** page where you need to specify your preferences for this application. Of special importance are two of these settings.

5   In the **Installation preferences** section of this page, choose **Install application to "/https" virtual host** if you wish this application to run in the SSL-protected manner, otherwise choose "/http".

6   If you install the first application on your domain, you will be able to choose between two destination directories in the **Installation preferences** section of the same page, that is, between **Document Root** and the folder with the application's name.

> **Note:** If you choose the **Document Root** destination folder, the application will be deployed to the root directory of the domain, which will make it impossible to install any other applications on this domain until this one is deleted. The deployment policy used in Application Vault is as follows: you can deploy as many applications on a domain as you need, each application isolated in its own application folder, or you can install one and the only application directly to the root folder of a domain.

7   In sections Database preferences, Administrator's preferences, etc. of the Site application installation page, enter passwords as required. A password should be 5 to 16 characters long, and it should not contain the login name used in this "login - password" pair.

Note: When installing an application on the domain, Plesk automatically generates the default name for the application's folder. If the domain has a similar application installed with the same preferences (namely, to the same /httpdocs or /httpsdocs root folder), then Plesk asks to give a different name to the application folder as the folder with the default name already exists. *Important*: the same is true if an application has been installed, then deleted, and now is being installed again. In this case, the files created by the previous application instance may still remain in the file system, and so does the application folder named by default if it contains such files.

Once all the steps are passed through, Plesk starts the installation routine during which it verifies the installation package structure and virtual host settings (whether the required technologies like PHP, CGI, Apache ASP, etc. are supported), checks the disk space and databases available, then deploys and configures the application, and registers it in Plesk database.

Finally, Plesk updates the **Installed Applications** page and displays it to the user and shows the newly installed web application in the list of installed applications at the bottom of the page.

If you are a Plesk Administrator, you can deploy an application on the selected domain as follows:

1 On the navigation pane located to the left, choose the **Domains** link.

2 Once you have got to the **Domains** page, check the domain you wish to deploy on in the list of domains located at the bottom of the page.

3   Follow the instructions given for Domain User, beginning from step 2.

How to Reconfigure an Application on the Domain Side

Reconfiguring an application deployed on the domain side is accessible to Domain User of a given domain, and to Plesk Administrator as well.

If you are a Plesk Administrator, pass through the following steps:

1   In the navigation pane located at the left-hand area in **Plesk Control Panel**, select the **Domains** link in the **General** section.

2   On the **Domains** page, there will be a list of domains at the bottom of the page that are associated with Plesk. Click on the name of the domain that contains an application you wish to reconfigure.

3   On the **Domain** page opened for the selected domain, click on the **Application Vault** button in the **Hosting** section.

4   On the **Installed Applications** page referring to the selected domain, you will see the list of installed applications at the bottom of the page. To reconfigure a certain application, click on the relevant icon against the required application.

5   On the **Site application reconfiguring** page associated with the selected application, edit your preferences and settings as planned and click **OK**.

Plesk will apply the modifications made to the selected application and reload the **Installed Applications** page anew.

If you are a Domain User, follow the instructions stated below:

1   In the **Plesk Control Panel** navigation pane, select the **Home** link.

2   On the **Domain** page, click on the **Application Vault** tool located in the **Hosting** section.

3   Proceed to step 4 of the instructions described for Plesk Administrator (see above).

How to Delete an Application from the Domain

An application deployed on a domain can be deleted by Domain User of a given domain and by Plesk Administrator.

If you are a Plesk Administrator, proceed through the following steps:

1   On the navigation pane located at the left-hand area in **Plesk Control Panel**, select the **Domains** link in the **General** section. You will get to the **Domains** page containing the list of domains at the bottom of the page that are associated with Plesk.

   Note: This page can be accessed via Plesk Desktop. On the navigation pane, click on the **Desktop** link, then click on the **Remove** link below the **Install a site application** button in the **Tools** section, and get to the **Site application installation** page where you can choose the required domain in the list.

2   On the **Domains** page, there will be a list of domains associated with Plesk. Click on the domain that contains an application you wish to delete.

3   On the **Domain** page opened for the selected domain, click on the **Application Vault** button in the **Hosting** section.

4   On the **Installed Applications** page referring to the selected domain, you will see the list of installed applications at the bottom. To delete a particular application, check it and click the **Remove Selected** button located in the right upper corner of the application list.

5   On the **Removal** confirmation page, check the **Confirm removal** checkbox and click **OK**.

Plesk will delete the selected application both from Plesk database and from related folders, after which the **Installed Applications** page will reload and display the list of installed applications without the deleted one.

Note: If the folders associated with the application to be deleted contain files that have been created by the application during its execution, such files and folders cannot be deleted from the domain. The problem is that the delete operation performs with permissions of Domain User, which is not enough to manage the files created with permissions of Apache User. E.g. if an application has been deployed in the application folder within the /httpdocs root folder and there have been any files created by that application during its execution, then such files and the application folder will not be deleted from /httpdocs if one deletes that application from the domain. The side effect is as follows: when one tries to reinstall the same application to /httpdocs again, he will be asked to rename the application folder as the application folder with the default name is present in /httpdocs already.

If you are a Domain User, follow the instructions stated below:

1   In the **Plesk Control Panel** navigation pane, select the **Home** link.

2   On the **Domain** page, click on the **Application Vault** tool located in the **Hosting** section.

3   Proceed to step 4 of the instructions described for Plesk Administrator (see above).

C H A P T E R   4

Package Physical Structure

A fully functional web application's installation package should meet the requirements laid upon its file and folder arrangement. The application's folder nested within the Application Vault (AV) root directory (…/`<plesk_root_dir>`/`var/cgitory/`) should look as follows:
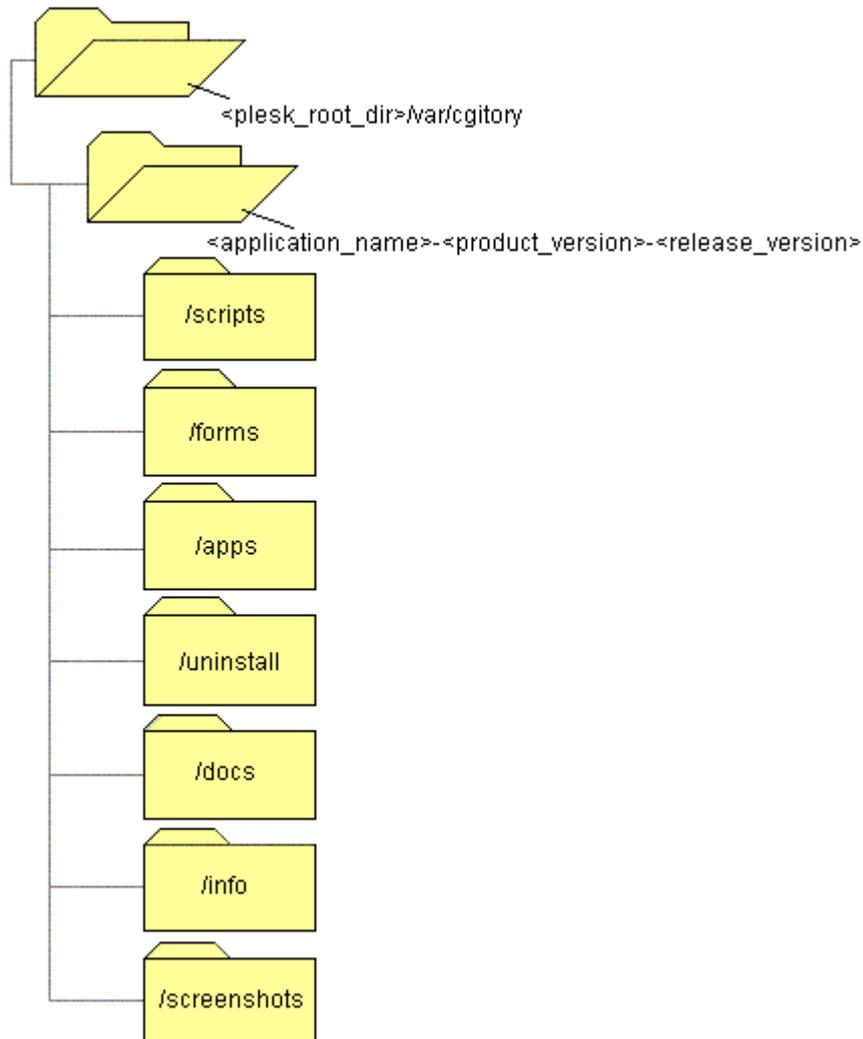
Figure 4: The structure of the installation package

The /apps folder is a real repository of application files, and the remaining folders are service ones used by AV mechanisms.

Folder name	Description
scripts	Contains **shell** and **perl** script files used when installing/reconfiguring/uninstalling applications on the domain side.
forms	Contains PHP files of two kinds – data input forms necessary to install/reconfigure/uninstall applications on a domain, and files with event handlers matching these forms.
apps	Contains TAR archive files that store files of the application itself and extracted on a domain.
uninstall	Contains the **uninstall** script used while removing the installation package from AV Repository. This script is optional.
docs	Contains a document displayed by Plesk Context Help system to describe a given application.
info	Contains the info.xml file that describes the application's properties. The file is necessary to display the application in Plesk Control Panel. Also, the /info folder contains an image used by default if a custom button is created.
screenshots	Contains screenshots of the application's GUI.

Scripts folder

In this folder Application Vault searches scripts participating in the deployment of applications from AV Repository on a domain, in uninstalling them from the domain, and in reconfiguring them as well. The names of the scripts and their quantity are fixed in the Application Vault constants file. It is proposed that the scripts are called within the listed procedures in the following order.

Script	Use	Description
preinstall	optional	Is called from within the installation procedure after AV has obtained the application parameters from the user, but before the TAR archive file is unpacked to a domain.
postinstall	optional	Is called from within the installation procedure after the TAR archive file is unpacked on a domain. This script is used to make modifications to the configuration file of the application deployed on the domain. If executed correctly, the script returns '0', otherwise it returns an error (any non-zero value), though the application is registered on the domain OK anyway.
reconfigure	optional	Is called from within the reconfiguration procedure after AV has obtained the new application parameters from the user. This script modifies the configuration file of the application deployed on the domain.
preuninstall	optional	Is called from within the uninstall procedure before the application files are deleted from the domain.
postuninstall	optional	Is called from within the uninstall procedure after the application files are deleted from the domain.
preupgrade	not used	The script is reserved for future use.
postupgrade	not used	The script is reserved for future use.

Forms folder

This folder should keep the dialog forms where the user will enter parameters necessary to install/reconfigure a given application on the domain side. Also, this folder should contain event handler files referring to these forms. If the folder is empty, the installation package is invalid.

When the install/reconfiguration procedure is triggered for a web application, first the AV mechanism searches for a respective parameter input form across this folder. The parameter input form is implemented as a PHP file with its name formatted as follows:

```
installer-form-<step number>.php
reconfigure-form-<step number>.php
```

The PHP file name specifies the type of operation, indicates that the file is an input form, and specifies the step number (which is reserved for multi-step wizards).

After the user has entered the required information in the parameter input form and pressed the **OK** or **NEXT** button, Application Vault looks through this folder again, this time searching for a matching handler. A handler is a PHP file with the following name format:

```
installer-handler-<step number>.php
reconfigure-handler-<step number>.php
```

This name format is similar to the one used for forms, except it describes the file as a handler.

The following table summarizes the above information and specifies what objects should be present in the `/forms` folder of a valid installation package:

Procedure	Form		Handler	
	Name format	**If missing**	**Name format**	**If missing**
Install	`installer-form-<step_number>.php`	error	`installer-handler-<step_number>.php`	error
Reconfigure	`reconfigure-form-<step_number>.php`	error	`reconfigure-handler-<step_number>.php`	error
Deletion	Not used	ok	Not used	ok

The table presents three important facts, that is:

- It is important that the developers always follow the naming convention when creating form and handler files, as the AV mechanism fully relies on it and just skips files with incorrect names from processing.

- A called procedure cannot be started if a respective form is missing in the `/forms` folder of the installation package.
- If Application Vault has received the 'button pressed' event from the form, but failed to find a matching handler, the button pressure remains unhandled, i.e. the installation/reconfiguration process cannot be completed.

Apps folder

This folder should store the files of the application itself, all of them packed into one or two TAR archives files, `httpdocs-files.tar` and `cgi-bin-files.tar`, depending on what type the application's files are.

The files of the web application can be organized into a folder structure. They should be packed into the `httpdocs-files.tar` archive file. When being unpacked, these files and their folder structure (if any) will be copied to the folder specially created for this application in the `/httpdocs` host directory, or in the `/httpsdocs` one if it is planned to run the application using the SSL-protected connection.

If the application contains CGI script files, it is recommended that these files are packed into a separate `cgi-bin-files.tar` archive. In this case, when the application is being installed on a domain, these files and the whole folder structure (if any) will be copied to the application folder created in the `/cgi-bin` host directory.

So, if the application contains both CGI files and forms, the installation package must contain two archive files in its `/apps` folder: `httpdocs-files.tar` and `cgi-bin-files.tar`. When installing this application, Plesk will create two application folders, one in the `/httpdocs` or `/httpsdocs` host directory and another one in the `/cgi-bin` host directory. The `httpdocs-files.tar` and `cgi-bin-files.tar` archives will be unpacked to these folders respectively, their inner file and folder structure kept unchanged.

Both archive files are optional in the sense that one of them may be missing in the `/apps` folder. In case both are missing, the installation package is not considered invalid as the application files can be generated by the **preinstall** or **postinstall** script.

Info folder

This folder contains the `info.xml` file that describes the properties of an application, its system requirements, and its configuration parameters. This files serves as a beacon for Plesk Control Panel, informing it about a separate application stored in AV Repository. In other words, if either the `info.xml` file or the `/info` folder is missing in the application package, the application package will not be visible to Plesk.

For details on `info.xml` refer to the **info.xml File** section of the Reference.

Docs folder

This folder contains the `index.<locale_name>.html` file that contains a description of a given web application. This description will be displayed to the user in a separate help window after the user clicks on the **"?"** sign against the application listed in AV Repository (select **Application Vault** on the **Server Administration** page of Plesk to display the contents of AV Repository).

When creating the description file, one should follow the naming convention defined for such files. It requires that the file name specifies the locale settings, namely, the language and the dialect in which the document is written, in place of `<locale_name>`, e.g.:

```
index.en-US.html
index.en-UK.html
index.de-DE.html
```

Uninstall folder

It is expected that this folder contains the **uninstall** script meant to for low-level operations on AV Repository when an application package is being deleted from it.

The **uninstall** script and the `/uninstall` folder are optional.

Screenshots folder

This folder contains screenshot files `app_screenshot_<number>.png` and `app_screenshot_thumb.png`. The first file is a full-sized screenshot of an application's GUI. There can be as many full-sized images as necessary, and their names will differ in the <number> section only. The second image is a thumb version of any full-sized image. The `app_screenshot_thumb.png` image is shown on the **Site Application Package Information** form of AV Repository (accessible if you select **Application Vault** on the **Server Administration** page of Plesk and click on any application package listed below). A mouseclick on a thumb image will load all full-sized images in a separate window.

C H A P T E R   5

Building Installation Package

This chapter will guide the developer through the process of creating an installation package. This includes nine steps as follows:

1 First the hierarchy of folders of the installation package is formed.

2 The `/apps` folder of the installation package is filled with the application's tar archive files.

3 The description file is added to the `/docs` folder of the installation package.

4 The `/screenshots` folder is filled with screenshots of the application's GUI. These screenshots are displayed on the application's information page in Application Vault.

5 Forms and handlers are created and added to the `/forms` folder of the installation package.

6 Application scripts are created and added to the `/scripts` folder of the installation package.

7 The **uninstall** script is created (if necessary) and added to the `/uninstall` folder of the installation package.

8 The `info.xml` file is written and added to the `/info` folder of the installation package.

9 The installation package is wrapped into an RPM/SH/DEB distribution package and passed to Plesk Server Administrator for allocation in AV Repository.

In this section:

Step 1. Making up the hierarchy of folders

If you are going to build an installation package for a web application, making up the hierarchy of folders is a good point to start. This hierarchy should begin with the system root and look as follows: `/<plesk_root_dir>/var/cgitory/`….

The `/cgitory` folder is the root of Application Vault. The application's main folder nested within should have a name formatted as follows:

```
<application_name>-<product_version>-<release_version>
```

- The `<application_name>` section stands for the name of the application. This section should fully match with the name of the application specified in the `info.xml` file. Allowed are all literals and digits, spaces are inadmissible. The length of the section is not restricted.

- The `<product_version>` section indicates the current version of the application. It can contain literals, digits, and 'dot' delimiters. The length of the section is not restricted.

- The `<release_version>` section is used to indicate the version of the current release. Allowed are literals, digits, and dots. The length of the section is not restricted.

- These sections should be delimited with the '-' character. There should not be any spaces between the sections and their delimiters. The length of the application folder name is unlimited.

Here is the example of a valid application folder:

```
phpAds-2.0.5-7512
phpBB-2.0.17-7118
phpBook-1.50-8011
```

The application folder can contain up to seven subfolders as follows:

📂	`/<application_name>-<product_version>-<release_version>`	
	📂	`/apps`
	📂	`/docs`
	📂	`/screenshots`
	📂	`/forms`
	📂	`/scripts`
	📂	`/uninstall`
	📂	`/info`

Please note that all folder names are fixed and given in the lower case.

It does not matter in what order these folders follow one another. Moreover, if some files of the web application are missing, the relevant folder can be missing in the installation package too. If the application's folder is added some extra subfolders, the installation package is considered valid anyway. Application Vault will copy these extra folders to AV Repository, but the further use of resources stored in such extra folders is the sole responsibility of the developer.

Thus, this step should result in the hierarchical structure of empty folders arranged as described above and having fixed names.

Step 2. Adding the application files to the /apps folder

At this step the files of your web application need to be packed into TAR archives and added to the `/apps` folder. Plesk 'recognizes' two TAR archives: `httpdocs-files.tar` and `cgi-bin-files.tar`. In other words, it is expected that the developer will isolate all application files in two groups (TAR archive files) according to the following principle:

- CGI script files (if any) should be put into the `cgi-bin-files.tar` archive. When installing the application on the domain, Plesk will copy these files to the application folder created in the `cgi-bin` host directory.

- The remaining files of the web application should be packed into the `httpdocs-files.tar` archive file. When installing the application on the domain, Plesk will copy these files to the folder specially created for this application in the `/httpdocs` host directory or in the `/httpsdocs` one if it is planned to run the application using the SSL-protected connection.

Each group of files can be organized into a convenient folder structure within the TAR archive if necessary. In this case this folder structure will be also copied to a proper folder during the install procedure.

Note: The above instructions refer to the most common situation when the application presents a complete executable unit composed of a definite set of files. But sometimes the application files do not exist at the stage of forming an installation package yet, e.g. they would be generated by installation scripts during the install routine. In this case, leave the `/apps` folder empty - Plesk considers such installation packages valid anyway.

Thus, this step should result in one or two TAR archive files, `httpdocs-files.tar` and `cgi-bin-files.tar`, created and added to the `/apps` folder, or this folder can be left empty, provided that a special *installation script file* added to the `/scripts` folder would generate the application files during the installation routine.

Step 3. Adding the description file to the /docs folder

This step is optional and only necessary if the application requires an associated description that would be displayed to the user on demand. The description itself can present a small HTML-formatted text or even a user manual. The only requirement is the name of its index file formatted as follows:

```
index.<locale_name>.html
```

The `<locale_name>` section should specify the *language* and the *dialect* in which the document is written, e.g.:

```
index.en-US.html
index.en-UK.html
index.de-DE.html
```

To associate the description file with the application it describes, put the file to the /docs folder of the installation package. Once this is done, the description can be displayed to the user in a separate help window at a mouseclick on the "**?**" sign against the application in the list (**Plesk Contorl Panel**->**Server**->**Application Vault** or **Plesk Control Panel**->**Domains**->click on a certain domain in the list->**Application Vault**).

Step 4. Adding GUI images to the /screenshots folder

This step is optional. Do not skip it if you want to demonstrate the images of the application's GUI on the information page.

The information page of your web application is formed by Plesk (Plesk Control Panel -> **Server** -> **Application Vault** -> click on any installation package in the list). It can display the *thumb* image of GUI only. If the *full-sized* versions of GUI are provided, they will be displayed in a separate window (all in one) once the *thumb* image is clicked.

If you are going to use this feature, proceed through the following steps:

- First of all, you need at least one image file - the *thumb* image named `app_screenshot_thumb.png`. The file name and the PNG graphics format are fixed for any web application.
- Add the thumb image to the /screenshots folder of the installation package. There must be *one and the only* thumb image in this folder.
- If you need full-sized images of the application's GUI, create them as PNG files with the names formatted as follows:

```
app_screenshot_<screenshot_number>.png
```

The names of the full-sized image files are fixed for any web application. The `<screenshot_number>` section indicates the ordinal number of the image, e.g.:

```
app_screenshot_1.png
app_screenshot_2.png
app_screenshot_3.png
```

▪ Add the full-sized images to the `/screenshots` folder of the installation package. The number of such image files is unlimited.

Note: The thumb GUI image serves as the button on the information page of a web application. If one clicks on it, a separate window with full-sized images is opened. If the `app_screenshot_thumb.png` file is missing in the `/screenshots` folder, the thumb image will not appear on the information page, so the full-sized images put to the same `/screenshots` folder would be inaccessible .

Thus, if not skipped, this step should result in a set of GUI images added to the `/screenshots` folder of the installation package.

Step 5. Adding forms and handlers to the /forms folder

Within the application management cycle, there are 2 procedures that require input parameters from the user. Input parameters are required when deploying a web application on the domain as well as when reconfiguring the deployed application. To let the user enter parameters via GUI, it is necessary to create parameter input forms and put them to the `/forms` folder of the installation package.

The parameter input procedure can require a single form or a multipage GUI. At a transfer from one form to another, the old form 'dies' and the values entered by the user need to be saved first. This is done by means of a special 'close form' *event handler* that reads all parameter values from the fields of the form and saves them in memory. Thus, each form requires a matching *handler* file. The matching handler files are added to the `/forms` folder beside their forms.

Later on, this topic considers the following aspects:

▪ it states the naming convention for form and handler PHP files (on page 38);
▪ it describes the creation of PHP forms (on page 39);
▪ it describes the creation of PHP handlers (on page 42).

Naming convention and common rules

The files of the /forms folder should be created according to the following rules.

1. The file name format is as follows:

```
<ACTION>-form-<STEP>.php
<ACTION>-handler-<STEP>.php
```

Here <ACTION> is 'installer' for the install procedure and 'reconfigure' for the repeated configuration procedure; <STEP> stands for the ordinal number of a parameter input form in a multistep procedure. E.g., the contents of the /forms folder can look as follows:

```
installer-form-1.php
installer-handler-1.php
installer-form-2.php
installer-handler-2.php
reconfigure-form-1.php
reconfigure-handler-1.php
```

2. Only PHP files are allowed.

3. All name sections are case-sensitive and should not contain capital letters. There should not be spaces between sections and '-' delimiters.

4. Every *form* file should have a matching *handler* file such that its name differs from the name of the form file in the second section only.

5. In multistep form sets, there should not be 'holes' in numbering. E.g. if the /forms folder contains form files installer-form-1.php and installer-form-3.php only, then Plesk may fail to find the form next to the first one.

Note: If the /forms folder is empty, the installation package is considered invalid.

Creating forms

A standard parameter input form looks as follows:

Figure 5: Installing a web application on the domain

No matter what web application is being installed on the domain, the layout of this form has much in common for all of them: there is the **Site Application installation** caption at the top, the **Info** section with attributes **Name**, **Version**, **Release**, and **Description**, the **Installation preferences** section, and the navigation buttons at the bottom of the page. These items are the *minimal* set of information required for the successful installation. Most of them (except the **Installation preferences** section and its contents) are generated by Plesk on basis of its own internal schemas and the application's info.xml file. The developer never takes care of this contents.

In contrast, there are elements that are under the sole responsibility of the developer. These elements come in two flavors: the **Installation preferences** section and its contents are *required*, other sections (e.g. **Database preferences**, **Administrator's preferences**, etc.) are *custom*. Plesk knows nothing about these elements until it is informed about them directly. This is the point where the PHP *form* file comes to help: this file (or a set of files) provides Plesk with the information what parameter input fields to display on the form, and what data types and formats are required in them as well.

From the programming aspect, a parameter input form presents a typical HTML page generated by Plesk. It is structured as follows:

```
<HTML>
<BODY>
      <!-- Styles and service functions -->
      <…>
      <FORM …>
            <!--  1. Here comes the block of tags and scripts
generated by Plesk to display the required elements on the parameter
input page -->
            <…>
            …
            <!--  2. Here Plesk inserts the text from installer-form-
1.php that refers to the Installation Preferences section -->
            <…>
            <!--  3. Here Plesk inserts the text from installer-form-
1.php that refers to other custom sections -->
            <…>
            …
            <!--  4. Here Plesk generates the block that displays the
navigation buttons on the parameter input page -->
            <…>
      </FORM>
      …
</BODY>
</HTML>
```

Note: The source code generated by Plesk for a parameter input page is accessible in the TXT file format if one right-clicks on the page and selects the **View Source** menu item in it.

Thus, Plesk automates the task of forming a parameter input page practically in full: it creates a frame of HTML code, finds a PHP form file in the /forms folder of the installation package, and inserts the code from this file into the frame. The only thing that remains to the developer is to write this PHP form file (or several files if a multistep wizard is desired).

It's up to the developer to decide how this will be done - Plesk does not provide direct recommendations on the programming techniques to use. For example, writing a form file could include defining the CSS style, creating some classes for elements of the page, and describing the page contents using these resources. The description itself comes to making up a pair of elements like <LABEL TEXT>-<INPUT ELEMENT> for each parameter.

For instance, the **Administrator's preferences** section requires three pairs of this kind: "Administrator's login"-"admin_login", "Password"-"admin_passwd", and "Confirm password"-"admin_confirm".

Figure 6: The Administrator's preferences section of the parameter input form

The PHP form file could describe these parameters as follows:

```
<legend><?=msg('administrator_preferences')?></legend>
<table>
  <tr>
      <td class="name"><label for="id_admin_login/></td>
      <td>
      <input type="text" name="admin_login" id="id-admin_login"
value="">
      </td>
  </tr>
  <tr>
      <td class="name"><label for="id-admin_passwd"/></td>
      <td>
      <input type="password" name="admin_passwd" id="id-admin_passwd"
value="">
      </td>
  </tr>
  <tr>
      <td class="name"><label for="id-admin_confirm"/></td>
      <td>
      <input type="password" name="admin_confirm" id="id-
admin_confirm" value="">
      </td>
  </tr>
</table>
```

However, there is a strong recommendation to follow when it comes to giving a *name* for a custom input element. That is, *the name given to an input element and the parameter name associated with it should match.* In reference to the above example this means that the developer intends to pass three parameters named 'admin_login', 'admin_passwd', and 'admin_confirm', for which three input elements of the same names are created on the form.

The reason for this requirement is as follows. When Plesk starts the parameter handling routine, it reads values from the input elements of the page, after which it takes the *names* of the input elements one after another and searches PROPERTY elements *with the same names* in the `info.xml` file. E.g. the PROPERTIES section for the above example can look as follows:

```
<PROPERTIES>
  <PROPERTY name="admin_login" default="" type="string"
valtype="login" />
  <PROPERTY name="admin_passwd" default="" type="string" valtype="
password " />
  <PROPERTY name="admin_confirm" default="" type="string" valtype="
password " />
</PROPERTIES>
```

Once, and only if, a matching PROPERTY element is discovered, Plesk verifies the entered parameter value using the attributes of the PROPERTY element. If Plesk decides that the value is valid, it is passed in as a *parameter* to a handler file associated with the parameter input form. Thus, Plesk uses the input element name, the PROPERTY element, and the parameter as a bundle, and binding is made by their names.

A sample `installer-form-1.php` file (see **Code Samples** section of **Rererence**) demonstrates how to write a PHP form file for the install procedure. The `reconfigure-form-1.php` file serves as an example of a valid PHP form file.

Thus, this sub-step must result in a PHP form file (or a set of files) created according to the above description and put to the `/forms` folder of the installation package.

Creating handlers

A handler file related to a parameter input form should solve the following tasks:

- it gets the values submitted by the input form;
- it verifies the passed in parameters;
- it sets the verified parameters to global variables of Plesk.

To perform these tasks, the functions of *Application Vault (AV) API* come to help. They can be grouped as follows:

Getting/Setting/Verifying Parameters
sapp_check_install_prefix
sapp_get_domain_name
sapp_get_image
sapp_get_install_prefix
sapp_get_locale
sapp_get_param
sapp_get_ssl

sapp_get_submit_value
sapp_include_path
sapp_is_ssl_available
sapp_is_wrong
sapp_open_application_url
sapp_set_install_prefix
sapp_set_param
sapp_set_ssl
Operations on Databases
sapp_create_database
sapp_create_database_user
sapp_get_new_db_name
sapp_get_new_db_user
sapp_is_database_exists
sapp_is_database_user_exists
Error Handling
sapp_set_error
sapp_set_errormsg
sapp_set_warning
sapp_get_wrong
sapp_set_wrong
Format checkup
check_dbUserName
check_dns_dom
check_domain
check_email
check_filename

check_idn_domain
check_int
check_ip
check_mail_passwd
check_mailname
check_mask
check_pg_login
check_pg_passwd
check_phone
check_shortUrl
check_sys_login
check_sys_passwd
check_url
PHP Interpreter Directives
sapp_include
sapp_include_once
sapp_require
sapp_require_once

The examples of `installer-handler-1.php` and `reconfigure-handler-1.php` files demonstrate the use of these functions.

A valid handler file should return an integer that specifies the ordinal number of the next parameter input form to call. E.g. to execute the `installer-form-2.php` file after `installer-form-1.php`, the `installer-handler-1.php` handler should return '2'. If the current form is the last (or the only) one in the sequence of parameter input forms, then the matching handler should return '0' or a negative value.

Thus, this sub-step should result in a PHP handler file (or a set of files) created using AV API functions and put to the `/forms` folder of the installation package.

Step 6. Adding scripts to the /scripts folder

This step is optional. Notice it if you need to extend parameter input handling (that performs with permissions of Domain User) with actions that require permissions of System Administrator. E.g. being installed, web applications often require creating folders on the domain first, or they may require adding/changing records in a database, setting user permissions, and so on, during the install procedure or reconfiguring. All these actions require permissions of System Administrator.

To help overcome the limitation with Domain User permissions, Plesk provides support for several script files, each having a fixed name and reserved for a certain task. The names of the script files supported by Plesk are defined in its `sapp_constants.php` file as follows:

```
define('SITEAPP_SCRIPT_PREINSTALL',      'preinstall');
define('SITEAPP_SCRIPT_POSTINSTALL',     'postinstall');
define('SITEAPP_SCRIPT_PREUNINSTALL',    'preuninstall');
define('SITEAPP_SCRIPT_POSTUNINSTALL',   'postuninstall');
define('SITEAPP_SCRIPT_RECONFIGURE',     'reconfigure');
```

These script files can be created as UNIX *shell* scenarios, or they can be written in *any* scripting language (e.g. Perl) on condition that a relevant interpreter is installed in the system and allows file execution from CLI.

To use a reserved script when handling parameters, the developer should create it, give it a predefined name, and add it to the `/scripts` folder of the installation package. This guarantees that the script will be called by Plesk at a proper moment during the relevant procedure. To learn more about the *tasks* assigned to each reserved script as well as about the *moment* these scripts are executed, pass to the **Scripts folder (on page 29)** topic of the **Installation package** section in **Reference**.

A typical script file includes four kinds of operations, that is:

- first it retrieves parameters from the standard input stream;
- then it validates the accepted parameters;
- it sets a series of low-level operations using the values passed in via parameters;
- it returns the exit status.

The passed in parameters enter a script as a string formatted as follows:

```
"param1=value1;param2=value2;param3=value3..."
```

First this string needs to be parsed and defragmented into pairs like:

```
<parameter_name>=<parameter_value>
```

Here is the example of a valid Perl code extracting parameters from a standard input stream:

```
#!/usr/bin/perl -w
…
my @imp_params = qw( vhost_path domain_name install_prefix
ssl_target_directory admin_login admin_passwd );
```

Later on, these parameters are checked. Here is the code snippet that demonstrates this checkup:

```
#!/usr/bin/perl -w
…
my %params;
…

sub check_parameter
{
     my ($param) = @_;
     unless (defined $params{$param}){
          return 0;
     } else {
          return 1;
     }
}
```

Next comes the block of code responsible for various low-level (system) operations. E.g., the following code snippet specifies the installation path by which the web application will be installed on the domain:

```
#!/usr/bin/perl -w
…    …    …
my $proto;
my $documents_directory;
if ($params{'ssl_target_directory'} eq 'true'){
     $documents_directory = 'httpsdocs';
     $proto = 'https://';
} else {
     $documents_directory = 'httpdocs';
     $proto = 'http://';
}
```

After all low-level operations are described, the script must return its exit status. A script returns a zero value if its execution has been successful. A non-zero exit status indicates an error condition of some sort.

Please find the examples of valid script files preinstall, postinstall, reconfigure, and preuninstall in the **Code samples** section of this documentation.

Thus, this step should result in creating some reserved scripts (if needed) and in adding them to the /scripts folder of the installation package.

Step 7. Adding the uninstall script to the /uninstall folder

This step is optional. The deletion of the installation package is fully defined by the uninstall scripts of the 'wrapper' RPM (or DEB, or SH) package. A special **uninstall** script is created and added to the `/uninstall` folder of the installation package in order to solve some non-trivial tasks. E.g., the RPM/SH/DEB package just deletes files and folders of the installation package from the system, while the **uninstall** script can be used for cleaning out some other traces of the application's presence in the system.

When Plesk receives a command to delete an installation package from AV Repository, it searches for the **uninstall** script in the `/uninstall` folder of the installation package and executes it. If the **uninstall** script is not found, the uninstall procedure will not fail, but continue as defined by the RPM/SH/DEB package.

Thus, this step should result in the **uninstall** script added to the `/uninstall` folder of the installation package if any specific 'delete' actions are necessary.

Step 8. Adding the info.xml file to the /info folder

This step is important. It consists in adding a specially created `info.xml` file to the `/info` folder of the installation package. This file describes the properties of a web application, its system requirements, and its configuration parameters. It serves as a beacon signaling to Plesk Control Panel about a separate application available in AV Repository. If the `info.xml` file is missing in the `/info` folder, the application package will not be visible to Plesk. The detailed description of the structure of the `info.xml` file can be found in the **Reference** section of **Application Vault** SDK documentation. Its **Info.xml File** subsection can serve as a guide to creating a valid `info.xml` file.

Thus, this step should result in the `info.xml` file structured properly and added to the `/info` folder of the installation package.

Step 9. Creating an RPM/SH/DEB distribution package

Once all above steps of the installation package building procedure are passed through, the resulting package should be packed once again, this time to create a distribution package. Plesk supports three package formats of this kind:

- the RPM package (on page 48) format is 'understood' by Unix operating system;
- the SH package (on page 49) format is a standard shell script;
- the DEB package (on page 51) format targets Debian Linux.

Later on, these approaches are considered in detail.

Creating an RPM package

Wrapping the installation package into an RPM package implies two steps:

- first the SPEC file is created for the RPM package,
- then the RPM package is assembled.

A standard SPEC file should have a name formatted as follows:

```
<package name>-<application version>-<release number>.spec
```

A SPEC file contains the information necessary to build an RPM package. The header of this file contains the application's description, its name, version number, release number, etc. Also, this file contains instructions on the building process, lists of application files and lists of third-party applications necessary for the install procedure.

The following SPEC file created for the webExample v.2.0.2-3 test application can serve as an example:

```
Summary: A simply test program. //summary info
Name: webExample //name of application
Version: 2.0.2
Release: 3
Copyright: GPL //license type
Group: Applications/WWW //group of application
BuildRoot: %prodbuildroot //patch to build dir
Prefix: %product_root_d/var/cgitory //patch to RPM make dir
Requires: php # set any //needed software
Provides: Plesk-application-vault
%description
Simple description
%files
%defattr(-,root,root)
/usr/local/psa/var/cgitory/%{namesrc}-%{version}-%{release}/* //path
to application files
```

Once the SPEC file is ready, it's time to build an RPM package. This can be done using the RPM (Redhat Package Manager) utility by running the following command from CLI:

```
# rpmbuild --bb --target=noarch /usr/src/redhat/SPECS/<spec_file_name
>.spec
```

Once this command is executed, a message is displayed that informs the user where the resulting RPM package is located (normally, in /usr/src/redhat/RPMS/). E.g. the above example will put the RPM package to the /usr/src/redhat/RPMS/noarch folder.

Visit http://www.rpm.org/RPM-HOWTO/build.html for more details about the process of building RPM packages.

Creating an SH package

To wrap an installation package into a SH package, proceed through the following steps:

- create an empty SH file;
- create the contents of this file;
- encode the source installation package using UUENCODE;
- append the encoded installation package to the SH package.

A SH file should have a name as follows:

```
<application name>-<version number>-<release number>.sh
```

First you need to create an empty SH file, e.g. `exampleapps-2.0-1.sh`, and make it executable using the following commands:

```
> exampleapps-2.0-1.sh
chmod 777 exampleapps-2.0-1.sh
```

The contents of this file should contain a header that indicates that the package targets Plesk Application Vault:

```
#!/bin/sh
#Provides: plesk-application-vault
```

Next comes the section that describes how the source installation package will be decoded from UUENCODE. Then follows the code that copies the application's TAR archive files to a specified location, unpacks them, and removes the temporary installation files and folders.

The following snippet presents the structure of a SH file:

```
uudecode $0
app_name=`basename $0|sed -e "s/.sh//"`
path=/usr/local/psa/var/cgitory
echo "==============Installing package ${app_name}"
tar -C ${path} -xf ${app_name}.tar
if [ $? -ne 0 ]; then
echo "!Error: cannot untar ${app_name} in ${path}"
exit 1
fi
rm -f ${app_name}.tar
echo "==============+OK"
exit 0
```

Next the source installation package is encoded using the UUENCODE utility and added to the SH file using the following command:

```
uuencode exampleapps-2.0-1.tar exampleapps-2.0-1.tar >> exampleapps-
2.0-1.sh
```

The resulting SH package will be structured as follows:

```
#!/bin/sh
#Provides: plesk-application-vault
uudecode $0
```

```
app_name=`basename $0|sed -e "s/.sh//"`
path=/usr/local/psa/var/cgitory
echo "==============Installing package ${app_name}"
tar -C ${path} -xf ${app_name}.tar
if [ $? -ne 0 ]; then
echo "!Error: cannot untar ${app_name} in ${path}"
exit 1
fi
rm -f ${app_name}.tar
echo "==============+OK"
exit 0
begin 644 exampleapps-2.0-1.tar
M061V86YC9610;VQL+3(N,BTT+P``````````````````````````<B<
code>r>M`````````````````````````````````````````````````
M``````````````#`P-P-#`W-`,`,`````````````````````````````
M```````````````````````````````````````````````````````
M```````````````````````````````````````````````````````
end
```

Creating a DEB package

The procedure of packing an installation package into the DEB package resembles creating RPM packages. But there are two differences, that is:

- a CONTROL file is used instead of SPEC;
- assembling the package is done using the dpkg utility.

The CONTROL file contains various values necessary to the package management tool for the management purposes. Though there are not any format restrictions on the CONTROL file name, it is recommended that the developers follow a standard naming convention for such files:

```
<package name>-<application version>-<release number>.control
```

A CONTROL file should contain the following control information necessary for the source package:

- **Source**: the name of the source package. Optional.
- **Section**: the section of the distribution the source package goes into (in Debian: main, non-free, or contrib., plus logical subsections, e.g. 'admins', 'doc', 'libs').
- **Priority**: describes how important it is that the user installs this package. This item can be left as 'optional'.
- **Maintainer**: the name and email address of the maintainer.
- **Build-Depends**: the list of packages required to build your package. This item is optional.
- **Standards-Version**: the version of the Debian Policy standards this package follows, the versions of the Policy manual you read while making your package. This item is optional.

The information that describes the source package is as follows:

- **Package**: the name of the binary package. This is usually the same as the name of the source package.
- **Version**: the version information related to the application itself and to Plesk.
- **Installed-size**: the amount of disk space (in bytes) required for a given application.
- **Architecture**: the CPU architecture the binary package can be compiled for.
- **Depends**: the packages on which the given one depends. The package will not be installed unless these packages are installed. This item should be used only if the application absolutely will not run (or will cause severe breakage) unless a particular package is present.
- **Provides**: this field specifies that the package targets Plesk Application Vault.
- **Description**: a short description of the package.

<Here is the place where the long description goes>

The Package field is specified using the following format:

```
Package: psa-appvault-<APP_NAME_IN_LOW_CASE>
```

The **Version** field is formatted as follows:

```
Version: <APP_VER>-<PLESK_VERSION><APP_RELEASE>
```

Here <APP_VER> stands for the version of the packed application, <PLESK_VERSION> means the version of Plesk (e.g. 80 for Plesk 8.0), and <APP_RELEASE> means the number of the application's build.

The **Depends** field has the following format:

```
Depends: psa (>= 8.0)<APP_REQUIRES>
```

I.e. the required packages should follow one another without spaces, their versions specified in brackets as shown in the format string.

The field coming after the **Description** field should meet the requirements to follow: this should be a paragraph which gives more details about the package. There must be no blank lines, but you can put a single . (dot) in a column to simulate that. Also, there must be no more than one blank line after the long description.

Here is a sample CONTROL file:

```
Section: non-free/net
Priority: extra
Maintainer: SW Soft Inc. <info@swsoft.com>
Package: psa-appvault-advancedpoll
Version: 2.03-8014)
Installed-size: 12345678
Architecture: all
Depends: psa (>= 8.0)
Provides: plesk-application-vault
Description: Poll management system
AdvancedPoll is a poll management system.
```

To build a DEB package, you can use the **dpkg** utility and issue the following command:

```
dpkg -b ${build_dir} ${package_file}
```

Here is the example of such command:

```
dpkg -b /usr/local/appsbuilder/build/Mambo-4.5.2.3-4_deb31_80
/usr/local/appsbuilder/build/psa-appvault-mambo_4.5.2.3-8004_all.deb
```

Visit http://www.debian.org/doc/maint-guide/index.en.html for more details on the process of building a DEB package.