Application Packaging

Developer's Guide

Revision 1.0.26

Contents

CHAPTER 1

Introduction

This guide is aimed to help you implement Software-as-a-Service (SaaS) business model for your applications. Here in this manual you will find scenarios showcasing structure and methods of handling typical applications according to Application Packaging Standard.

About Application Packaging Standard

With the increased in development and the need to comply with integration requirements, hosting providers tend to offer a limited variety of hosted applications. As a result, independent software vendors have little incentive to create hosted applications that they cannot sell, and customers suffer from limited application choices.

In response to this, *Application Packaging Standard* (APS) was developed. It is an open format of packaging and managing web applications that will make it easier for the whole hosting industry to take advantage of the expanding *Software-as-a-Service* (SaaS) market.

By defining such open format, APS increases business opportunities for the entire hosting ecosystem bringing together application vendors and hosting service providers. By implementing APS, application vendors take advantage of sales and marketing channels available through APS-enabled hosting providers. Hosting service providers, in turn, gain access to a great variety of APS applications and render added value for their subscribers.

CHAPTER 2

About This Guide

The purpose of this guide is to help software vendors deliver an application to Customers in a structured format compliant to *APS: Package Format Specification*. This specification defines application services structure, settings, delivery method and other aspects required for automated provisioning of the application. Based on the two simple scenarios described here, you will learn how to package a SaaS application at a local and external environment. This way the application, delivered to end users via an APS-compliant system will generate revenue based on your licensing terms.

Useful Links

Official APS site. Latest news and relevant information on Application Packaging Standard development.

- APS Packaging Standard - Package Format Specification 1.2
- APS Integrating Application with Parallels Automation http://www.parallels.com/r/docs/poa/Integrating_Application_with_Parallels_Automation_by_APS/index.htm

XML Technologies:

- *XPath Homepage*
- *RELAX NG Compact Syntax Tutorial*

Typographical 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.
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 commands, files, and directories.	The license file is located in the `http://docs/common/licenses` 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`**
CAPITALS	Names of keys on the keyboard.	SHIFT, CTRL, ALT
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another.	CTRL+P, ALT+F4

CHAPTER 3

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.

CHAPTER 4

Packaging Instruction

Application Packaging Standard defines the packages structure and rules of packages processing. Package is a file that contains application files and metadata required to create and manage instances of the application.

Package contains essential information that uniquely identifies each package among others of the same application. Namely, it is application version, package release and packager's signature. For example, if packager changes something in application metadata or in application files, it must be reflected in metadata and signing data. This ensures package consistency and integrity.

Packaging Outline

Packaging an application includes three general steps:

1 Creating an application XML manifest.

2 Packaging an application with the manifest, scripts and a signed list of included files.

3 Installing the application on the customer end.

The first step may be itemized subject to application preferences and requirements. Such steps are described below. You are strongly recommended to familiarize with them while envisioning your project. Steps 2 and 3 are described in details in APS Getting Started Guide. It is also assumed here that you already have a working application.

Note: It is not recommended to make changes to APS packages created by other vendors. If however you need to alter an APS Package originally created by a different vendor, add alpha suffix to the version number. For example, if the original version is **1.2-3**, change it to **1.2-3a** for your custom version. This is necessary to retain the upgrade path: a newer APS package released by the vendor (e.g. 1.2-4 will upgrade 1.2-3), while the custom version will only be upgraded by version 1.2-4a or later.

---

1. Define type of an environment where the application is to be provisioned.

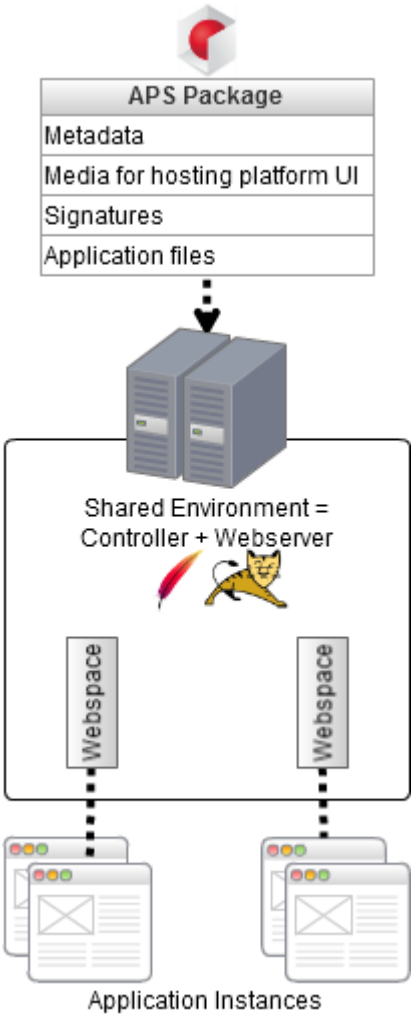
All applications may be classified by the type of environment where they are provisioned. Based on the required environment, the Controller performs the related provisioning procedures.

Web Applications Requiring Shared Environment (Wordpress, Joomla, Sugar CRM, etc.)

A web application may usually be downloaded as a distributable and deployed at the hosting environment. Such applications do not support services running in the background. To provision a Web application, a Shared Environment is required.

Besides the manifest part, related media and signatures, the APS package also includes all application files to be deployed at a web server.

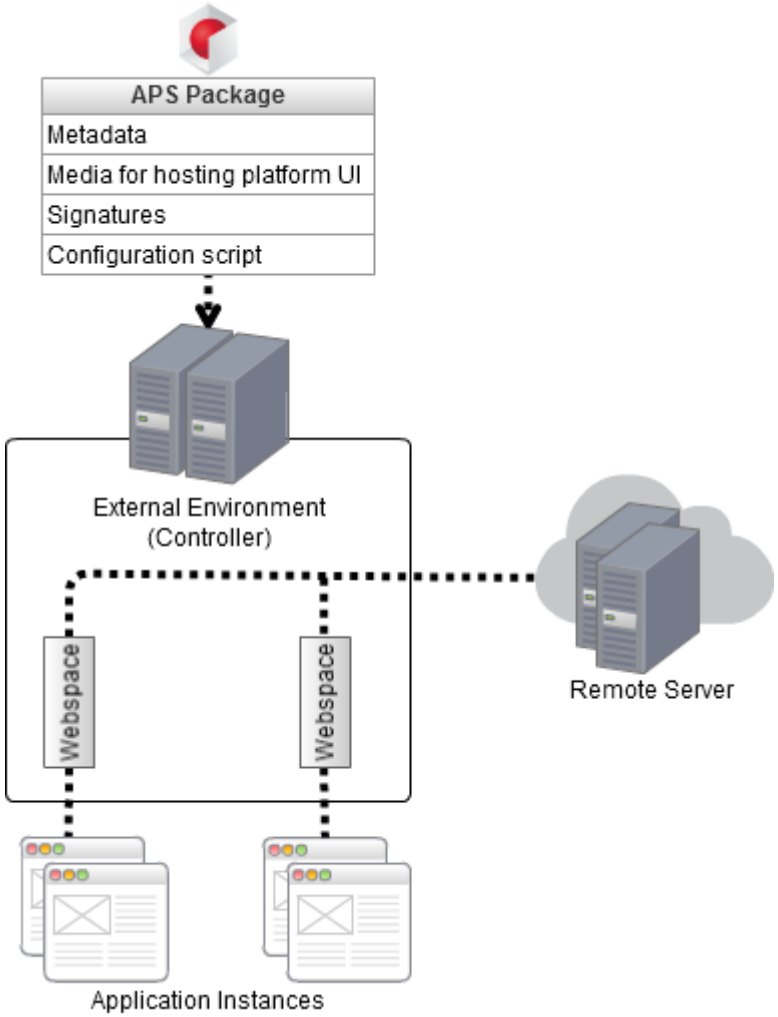
A shared environment includes a running web server. Each application is provisioned inside a partition (webspace) which is unique for each customer.


APS Package
Metadata
Media for hosting platform UI
Signatures
Application files
Shared Environment =
Controller + Webserver
Webspace
Webspace
Application Instances

Applications Requiring External Applications (Office365, Open-Xchage, LiveOffice, etc.)

External applications may run at the same server, where the APS-compliant hosting system is running or, alternatively, at a remote server. The Controller has no control over the external environment.

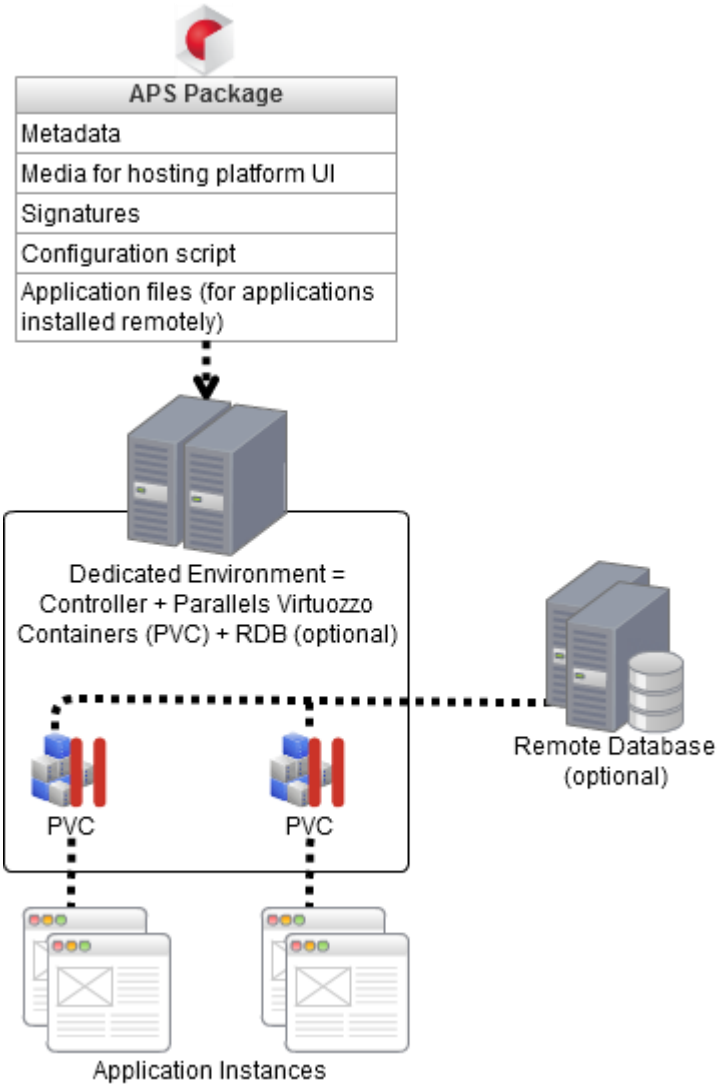
The APS package will contain only media, displayed in the Control Panel (icons, screenshots), signatures of included files and also scripts required to provision the application at an external service. In other words, the APS package comprises a connector to an application running at an external server.



Applications Requiring Dedicated Environment (VDI Desktop, 1C, Plesk, etc.)

As the term suggests, this type of applications require a dedicated OS instance. This is made possible through Parallels Virtuozzo Containers (PVC) technology (see PVC aspect for more details on PVC technology and PVC templates). Typically, applications of this type are not originally designed as web applications, i.e. such applications may be installed and operated in a standalone workstation/server. Also, such applications support services running in background mode.

The APS package includes manifest, media displayed in the Control Panel (icons, screenshots), signatures of included files. Application files may be included in the APS package or otherwise, a reference may be declare in the manifest to install application files from a specified path.



2. Define a model of service provisioning.

Selecting the provisioning model application vendor should consider the level of application management by Controller and application services hierarchy. Namely, one should correctly map application services on application. One may single out the following models of service provisioning:

- *Multi-tenant* - this model implies that a single instance of the application runs on hosting vendor servers, serving multiple client organizations (tenants). Tenants data and application configuration are stored separately. When a tenant logs in, an application instance respective configuration (customization) is applied and each tenant sees and manages only his data as if he is an application single user. So, in this model, application context is already exists in hosting environment and customers' data is stored within it separately.
- *Single-tenant* - this model implies that an application instance run on hosting vendor servers, serving only one tenant. In other words, a separate application instance is provisioned for each customer. Application files are simply copied to location dedicated to customer and data of all customer applications is stored within customer.

Note: both models can be implemented for shared environment and for dedicated one as well.

3. Define application structure.

An application may comprise main module and additional ones. The main module represents core functions, whereas additional modules provide extra services. For example, a content management application may come as a main module, and the skins or localizations may be offered as add-ons.

If your application has such additional modules, consider packaging them individually like a general package with dependency to their main module. In other words, metadata of packaged add-ons must reference to their master package, so the Controller could check whether the package is really an additional module to an application installed in hosting environment and its version meets condition (if any) in add-on's metadata file.

For details on packaging add-ons, refer to the **5.1.5. Master package reference** section of the Specification.

4. Define application services hierarchy.

In general, an application purpose is to provide services to end-users. Application services hierarchy should reflect order of application services provisioning and level of application services management by Controller.

For example, let's consider multi-tenant application. The top-level application service represents a tenant. Under top-level service an application context is declared (second-level service) that is provisioned inside of the tenant. Then one may declare a number of application features (third-level services). Another way is to declare users and specify features for a user.

To make it possible for a tenant to provide end-users with application accounts, the related application service of class 'account' should be declared in the APS package. Such services may be, for example, an email account, an editor's account in a blogging platform. In other words, a service of 'account' class binds an end-user to the application account. For details of implementation, see Packaging Open-Xchange Application (on page 35).

5. Define resources for application services

Application services may expose resources. Usage statistics for such resources are collected by the Controller and may be displayed in the Control Panel. For more information on how to declare resources, see Resources http://www.apsstandard.org/r/doc/package-format-specification-1.2/index.html#s.metadata.service.resources in APS Format Specification.

6. Define an application licensing procedure.

APS allows for packaging of licensed applications. A license may be bound to a single service or an application as a whole.

It is important to understand the difference between a licence as a document and a licence key (which is also referred here as a license). When being installed, an application displays to a customer a license text referred from the APS package (see 5.2.2 License Agreement in the Application Packaging Standard). Subject to a specific service platform, the license text may display automatically or otherwise, like in PBA Online Store, it has to be configured manually.

Traditionally, a license describes terms of use, payment terms, copyright, etc. But an application will not be activated until a valid license key is delivered to it. This happens only when the application is paid and installed, but here we assume that a payment terms are satisfied by the billing system (here - in PBA).

For details on license requirement declaration, refer to the **APS Licensing Aspect http://www.apsstandard.org/r/doc/aspect-licensing-1.2/** document.

7. Define technologies used by your application.

Your application may utilize various scripting languages, specific server modules, databases, mailboxes, DNS records in application domain and the like. Application technologies form the list of requirements to be satisfied to provision application. In general, requirements usually request some resource to be allocated or specific configuration to be performed.

A Controller can handle these requirements as qualified technologies – software, resources (e.g. databases, mailboxes or DNS records) or executable code described in the aspect included in the Specification. For details on aspects that describe qualified technologies, refer to the **8. Common aspects** section of the Specification.

Note: If the application uses non-qualified technologies, write an aspect for each such technology and make it public. You may then submit a request to extend the Specification (recommended scenario), or otherwise include the information into the package release notes. This will help control panel developers to configure Controllers to provision the package properly. For details on what data should be included in an aspect, refer to the **7. Points of Extensibility** section of the Specification.

Tip: For common understanding, please, use the terms given in the Specification.

8. Define presentation details and settings for application and its services.

Your package metadata may include details and settings of the application interface. If an application is multi-tenant, for example, it should meet requirements of a number of tenants that use this application for absolutely different purposes. In other words, the more settings you declare for an application, the more flexible it becomes. This way you may set up automated generation of setting values ensuring their uniqueness within the required scope. Some settings may be prefilled automatically with a Customer's login, password, email or domain of the application without requesting the data from end-user. Depending on the value length defined for a setting in the APS Package, it is displayed as a data field for settings with `min-length` attribute under 200 characters, or as a text area for settings with `min-length` attribute equal or over 200 characters. For details, refer to the **5.1 Common Application Properties, 5.2 Services** and **5.2.5 Service Settings** sections of the Specification.

Some settings may contain limiting values for resources declared in the Application. When referred to a limiting setting, the resource usage cannot be overused by the Customer. To define such a limit, a resource `limiting-setting` attribute is used in Resource declarations http://www.apsstandard.org/r/doc/package-format-specification-1.2/index.html#s.metadata.service.resources. This setting may be initialized with a default value in the APS package, and later it may be changed as required in the control panel.

Enum or boolean service settings may determine visibility of associated settings. For example, the values of an enum-type setting may include **Other** item, which will display an additional field where a Provider would enter a custom value. To make this possible, the associated setting should have visibility attributes.

To enable quick access to application services for a Customer, the APS package should include entry points (hyperlinks) referring to those services or parts of the application, e.g. login screen, control panel, etc. For more details, see **5.2.4 Entry Points**. For details of implementation, see Open-Xchange Sample Metadata File.

An APS package may include settings and non-APS classes which are supported by specific Control Panels. For information on settings and classes supported by Parallels Automation, see **PA-specific Classes** in **Integrating Application with Parallels Automation**.

Tip: For common understanding, please, use the terms given in the Specification.

---

9. Define application content delivery method.

Content delivery method defines the way application files are deployed to hosting environment. For now APS allows using the following methods:

- *Inside package* - the method implies that package content does not require any special processing prior to its deployment. The Controller simply copying the very application files and directories from the package to a hosting environment specific location, according to URL mapping rules (if any specified).
- *Inside PVC template* - the method implies that application files are packed in PVC template that regulates application files deployment. For details on PVC templates, refer to the Parallels Virtuozzo Containers: Templates Management guide.
- *Delivery is not required* - the method implies that hosting environment already contains all necessary content and do not require any configuration. This method suits for provisioning external service.

10. Create a draft of metadata file.

Each package must contain a well-formed XML file (called `APP-META.xml`) that includes all the metadata required to manage the application. Metadata file includes application common properties like name, version, description as well as list of application services and resources required. The XML presentation of the metadata is described by the RELAX NG schema. Some schema elements are defined by aspects. It means that you must use aspects when adding the elements. For details on metadata file and its elements, refer to the **4.3. Metadata File** and **5. Metadata Descriptor** sections of the Specification.

11. Prepare scripts.

The Specification defines the following provisioning methods: URL mapping and configuration script. These methods are not mutually exclusive and are not interdependent. Thus, both of them can be declared for a service provisioning. URL mapping configures access to service through web, and configuration script performs service installation. The selection of method is determined by the hosting environment which is required for the Application. APS Package may contain multiple sets of requirements to meet the environment in case either set of requirements may not be satisfied. For details on provisioning methods, refer to the **5.3. Service Provision Methods** section of the Specification.

You may develop a number of scripts that serve different purposes:

- *Configuration script.* It serves as method of service provisioning. Also, it is invoked by Controller on application instances configuration, removal, upgrade and enabling or disabling service. If an application has more then one service, separate configuration scripts may be implemented for each service. All resources required by a service must be allocated, instance files unpacked and placed to the file system before configuration script invocation. For details on configuration script invocation, its arguments and environment variables, refer to the **5.3.2. Configuration Script** section of the Specification.
- *Verification script*. It serves to verify service settings for consistency, but it should never change state of application or its instances. Verification script has the same calling conventions as a configuration script, but does not support `enable` and `disable` arguments. Verification script is invoked twice: during installation and reconfiguration of an application. Initially, it is executed prior to prompting a user for setting values. At this stage the application settings receive auto-generated and default values. Second time it is invoked to validate the submitted settings values. If the return value is not zero, the script is invoked again. For details on verification script invocation, its arguments and environment variables, refer to the **5.3.3. Verification Script** section of the Specification.
- *Resource script*. It serves to report application current resources usage to a Controller. The script utilizes the same calling conventions as configuration one, but its output stream should match XML scheme defined on the Specification. For details on resource script invocation, its arguments, output and environment variables, refer to the **5.3.4. Resource Script** section of the Specification.
- *Backup script*. It serves to preserve and restore users' data. It is invoked by Controller on application instance backing up and restoration. Also, an operation of migration is performed with this script as sequential execution of backup and restore operations. For details on backup script invocation, its arguments and environment variables, refer to the **5.3.5. Backup/Restore Script** section of the Specification.

  The script should be able to back up all sensitive application data to file specified and all involved resources, if required; restore backed up data to location specified.

- *License script*. It serves to install application license, if any. It is invoked when new license is issued for the application, existing license is updated or removed. The script should be able to get instance-specific license information, install issued license and remove installed one. For details on license script invocation, its arguments and environment variables, refer to the **APS Licensing Aspect** document.

Script-writing language must be a qualified technology defined in the Specification and must be declared in the metadata file. Application data is passed to scripts by means of environment variables. The variables contain all information about resources and settings of application instance you defined on the **7th** step of the instruction.

Scripts execution is not always going without a hitch. Consider making script output structure matching XML schema defined in the Specification. A Controller catches such structured output, reads instructions and quickly reacts, in case some errors occur. These instructions may include identifiers of settings with erroneous values accompanied by localized error messages or problems detailed descriptions as well as warnings and errors related to the requirements. For details on forming structure of script output, refer to the **5.3.2.3 Configuration script output** section of the Specification.

After necessary scripts are implemented, update metadata file with scripts declarations, according to the Specification. For details, refer to the **5.3. Service Provision Methods** and **8. Common Aspects** sections of the Specification.

12. Prepare package contents listing.

Each package must contain a well-formed XML file named APP-LIST.xml in the package root directory. The file must contain the list of all files in the package, but itself. Size in bytes and SHA256 digest value must be specified for each file in the list. This file may be digitally signed by a packager or certification authority.

Presence of packager signature assures package integrity and helps to prevent counterfeit. If a package is signed by certification authority, for example APS organization, the package contains a certificate as well.

For details on listing structure, refer to the **4.4 Package contents listing** and **6. Package contents listing** sections of the Specification

13. Create an application package structure.

Package structure must comply with the content organization requirements declared in the Specification. Check your package structure before archiving and update the application content if needed. Make sure, that *APP-META.xml* and *APP-LIST.xml* files are well-formed and contain all necessary information about an application.

For details on the requirements, refer to the **1. Introduction**, **4. Basic Package Format** and **6.5. Additional scripts** sections of the Specification.

14. Archive the application package files.

The package should be a ZIP archive with the `.app.zip` extension, according to the **4.1. File format** section of the Specification.

15. Validate the application package.

The package validation may comprise two steps: checking whether package structure conforms to APS and then checking whether package is properly managed by Controllers. Package structure can be validated using *APSLint* command-line utility. If the application package is APS-compatible, then you can check how it is managed by Controllers. To do this, examine the Controller operations that require application package data. For details, refer to the **Validating Application Package chapter** ("**Validating Application Package**" on page 78) further in this guide.

CHAPTER 5

Packaging Scenarios

Subject to sales flow and design of an application, you will need to follow a similar scenario. This chapter contains a several application packaging scenarios, grouped by provisioning environment (see 1. Define type of an environment where the application is to be provisioned ("1. Define type of an environment where the application is to be provisioned." on page 9)) plus describes integration of some additional components.

- Internally hosted applications (shared and dedicated environment):
  - Packaging Wordpress Application (on page 22). This is the simplest scenario showing how to define basic settings for a blogging application.
  - Packaging SugarCRM Application (on page 28). This scenario shows how to package an application in dedicated Parallels Virtuozzo Container (PVC or VPS).
  - Packaging a Wordpress Translation Add-On ("Packaging German Translation Add-on for WordPress" on page 32). This scenario extends the first one by showing how to package application add-ons: translations, themes and other enhancements adding extra value and enabling users to customize the core packaged application.
- Externally hosted applications (external environment):
  - Packaging Open-Xchange Application. This scenario shows how to package an external application bound to a mail service.
  - Packaging SpamExperts Incoming Email Security Firewall ("Packaging SpamExperts Incoming Email Security Firewall Application" on page 41). This chapter explains how to package a spam filtering application and point it to domains which it will service.
- Additional scenarios
  - Integrating Applications with Upsell Services (on page 44). This scenario shows how to define applications with different editions. If you want to package, for example, an email application offering collaboration options at an extra fee, this is your case. This scenario relates to provisioning of application services, rather than to provisioning of the application itself, and therefore it cannot be classified as either external or shared.
  - Integration with Domains (on page 45). This scenario describes how to integrate application with single or multiple domains and how to create DNS records on them.
  - Integration between Services (on page 47). This chapter explains how to integrate services of application with each other.

Shared and Dedicated Environments

Packaging WordPress Application

This chapter outlines the process of packaging WordPress - one of the most popular blogging platforms. WordPress may be installed on a shared web server or otherwise a user may sign up and create his own blog at an external domain (www.your-blog.wordpress.com). The latter scenario is covered in the APS Getting Started Guide. The more advanced scenario is packaging WordPress for shared web server environment.

The application requires a PHP interpreter, a MySQL database and takes up approximately 7Mb of diskspace. WordPress UI may be localized and delivered in a language of customer's choice.

1. Edit APP-META.xml

Based on the design and requirements of WordPress, edit APP-META.xml as follows.

1.1 General Details

An APS Package must declare at minimum the following general details:

- root node with reference to the APS Specification version that this APS package complies with
- application ID, name, version and release
- application category
- service(s) provided by the application.

Note: If you wish to package WordPress for both external and shared web server environments make sure the application ID's (see above) don't match. Otherwise application provisioning may fail.

To declare these details for our Wordpress application, add the following part to APP-META.xml:

```
<application xmlns="http://apstandard.com/ns/1" version="1.2">
<id>http://wordpress.org/</id>
<name>WordPress</name>
<version>3.0</version>
<release>5</release>
<homepage>http://wordpress.org/</homepage>
```

To add more details on the application and allow users locate it by packager or vendor name, add the following part:

```
<vendor>
      <name>WordPress.org</name>
      <homepage>http://wordpress.org/</homepage>
</vendor>
<packager>
      <name>Parallels</name>
      <homepage>http://parallels.com/</homepage>
```

```
      <uri>uuid:714f0a7b-85d6-4eb8-b68e-40f9acbb3103</uri>
</packager>
```

Once the APS Package is imported, the related details will display in the Control Panel. This information is defined on the next step.

1.2 Application Presentation

To define the way the application will appear in the Control Panel, both on the provider and on the customer end, the APS package should include a brief description, icons and screenshots, the application and the language used in the application UI. These details are defined in the following listing:

```
<presentation>
<summary>WordPress is a state-of-the-art semantic personal publishing platform with a focus
on aesthetics, web standards, and usability.</summary>
<description>
WordPress is a state-of-the-art semantic personal publishing platform with a focus on
aesthetics, web standards, and usability. What a mouthful.
WordPress is both free and priceless at the same time. More simply, WordPress is what you
use when you want to work with your blogging software, not fight it.
</description>
<icon path="images/icon.png"/>
<screenshot path="images/admin_page.jpg">
<description>
      Admin page.
</description>
</screenshot>
<categories>
      <category>Web/Blog</category>
</categories>
<languages>
      <language>en</language>
</languages>
</presentation>
```

1.3 Services

WordPress is a blogging platform and thus provides only one service. It may be extended by various plug-ins, themes and translations, but those are packaged separately as add-ons and each of such add-ons comprises an individual service (see Packaging a WordPress Translation Add-on ("Packaging German Translation Add-on for WordPress" on page 32)). To define the core application service for WordPress, add the following part:

```
<service id="wordpress"></service>
```

To define service details and settings, follow steps ...

1.3.1 License

WordPress is distributed under GPL licence. To point the application to the supplied license file, add the following part:

```
<license must-accept="true">
      <text>
      <name>GPLv2</name>
      <file>htdocs/license.txt</file>
      </text>
</license>
```

1.3.2 Service Presentation

To provide access to most frequently used application components, include entry points to the front-end, back end and other pages.

```
<name>Wordpress Instance</name>
	<summary>Wordpress blog engine</summary>
	<entry-points>
		<entry dst="/">
			<label>Blog</label>
		</entry>
		<entry dst="/wp-login.php" method="POST">
			<label>Administrative interface</label>
			<variable name="log" value-of-setting="admin_name"/>
			<variable name="pwd" value-of-setting="admin_password"/>
		</entry>
		<entry dst="/wp-login.php?redirect_to=wp-admin/theme-uploader.php"
method="POST">
			<label>Upload theme</label>
			<variable name="log" value-of-setting="admin_name"/>
			<variable name="pwd" value-of-setting="admin_password"/>
		</entry>
		<entry dst="/wp-login.php?redirect_to=wp-admin/plugin-uploader.php"
method="POST">
			<label>Upload plugin</label>
			<variable name="log" value-of-setting="admin_name"/>
			<variable name="pwd" value-of-setting="admin_password"/>
		</entry>
	</entry-points>
```

The application prompts for administrator's credentials which are automatically supplied as variable values, defined below.

1.3.3 Service Settings

WordPress requires a number of settings for successful installation and configuration. These are: administrator's credentials, blog title and locale settings. All settings are grouped by types.

```
	<settings>
		<group>
			<name>Administrator's preferences</name>
			<setting id="admin_name" type="string" default-value="admin"
min-length="1" max-length="32" regex="^[a-zA-Z][0-9a-zA-Z_]*">
				<name>Administrator's login</name>
				<error-message>Please make sure the text you entered starts with
a letter and continues with either numbers, letters, underscores or hyphens.</error-message>
			</setting>
			<setting id="admin_password" type="password" min-length="1" >
				<name>Password</name>
			</setting>
			<setting id="admin_email" type="email">
				<name>Administrator's email</name>
			</setting>
		</group>
		<group>
			<name>Weblog's preferences</name>
			<setting id="title" type="string" min-length="1" default-value="My
Blog">
				<name>Weblog title</name>
			</setting>
```

```
            </group>
            <setting id="locale" class="locale" type="enum" default-value="en-US">
                  <name>Interface language</name>
                  <choice id="en-US" >
                    <name>English</name>
                  </choice>
                        <choice id="ru-RU" >
                    <name>Russian</name>
                  </choice>
                  <choice id="de-DE" >
                    <name>German</name>
                  </choice>
                  <choice id="fr-FR" >
                        <name>French</name>
                  </choice>
                  <choice id="es-ES" >
                        <name>Spanish</name>
                  </choice>
                  <choice id="nl-NL" >
                        <name>Dutch</name>
                  </choice>
            </setting>
      </settings>
```

With these settings defined, the application will prompt the user to complete the necessary data upon installation.

1.3.4 Service Requirements

As mentioned above, WordPress requires PHP version 4.3 or greater and MySQL version 4.0 or greater. To define these requirements, add the following part.

```
      <requirements xmlns:php="http://apstandard.com/ns/1/php"
xmlns:db="http://apstandard.com/ns/1/db">
            <php:version min="4.2.0"/>
            <php:extension>mysql</php:extension>
            <php:safe-mode>false</php:safe-mode>
            <db:db>
                  <db:id>main</db:id>
                  <db:default-name>wordpress</db:default-name>
                  <db:can-use-tables-prefix>true</db:can-use-tables-prefix>
                  <db:server-type>mysql</db:server-type>
                  <db:server-min-version>4.0.0</db:server-min-version>
            </db:db>
      </requirements>
```

1.3.5 Delivery Method

WordPress is installed on a customer's or vendor's (third-level) domain, and therefore all application files should be publicly accessible from the web. To make this possible, url-mapping rules and required disk space should be defined as follows.

```
<provision>
      <configuration-script name="configure">
      <script-language>php</script-language>
      </configuration-script>
      <url-mapping>
            <default-prefix>wordpress</default-prefix>
      <installed-size>6696960</installed-size>
      <mapping url="/" path="htdocs" xmlns:php="http://apstandard.com/ns/1/php">
```

```
            <php:permissions writable="true"/>
            <php:handler>
                  <php:extension>php</php:extension>
            </php:handler>
            <mapping url="wp-config.php"><php:permissions writable="true"/></mapping>
            <mapping url="blogs/media"><php:permissions writable="true"/></mapping>
            <mapping url="wp-content"><php:permissions writable="true"/></mapping>
            <mapping url="tmp"><php:permissions writable="true"/></mapping>
      </mapping>
      </url-mapping>
</provision>
```

If you followed all above steps carefully, you should have a file similar to the example ("WordPress Sample Metadata File" on page 49).

2. Create an archived APS package

Create a zipped APS package with the following file structure:

APP-META.xml	# Metadata container. XML file.
scripts/	
configure	# This script will be invoked when application instance is to be setup.
images/	
icon.png admin_page.jpg	# Icon and screenshots of the application
htdocs/	
blogs/ ... index.php ...	# WordPress files

3. Validate the APS Package

Prior to importing the package, double check that it is valid and thus complies with APS. For instructions on APS package validation see Validating Application Package (on page 78).

4. Test the Result

If your package proved valid on the previous step, the last thing to do is actually run it on the end Customer's side. Make sure your destination environment meets the requirements defined in **1.3.4 Service Requirements** above and install the package as described in **APS Getting Started Guide** (see **Hello World for Shared Environment**).

Packaging SugarCRM Application

Sugar CRM Community Edition enables organizations to efficiently organize, populate, and maintain information on all aspects of their customer relationships. It is more complex application then WordPress and requires more CPU power and memory. The application is managed by administrator, who has power to regulate access to application for multiple users and customize the look and feel of the application across an organization.

SugarCRM requires the following components to be installed on server: PHP, Apache and MySQL. Because of necessity in more powerful hardware, shared web server environment is not enough for application and it has to be installed in a dedicated virtual machine or virtual container (VPS). The last one can be done via APS.

Listed below are the sections of an APP-META.xml file supplied with your package:

1. Type of environment

The application vendor states that Sugar CRM requires own CPU and memory. So, we can draw a conclusion that the application required dedicated environment type. It means application files have to be delivered via PVC templates, see PVC 4.6 for Linux Templates Management Guide on how to package files in a PVC template. PHP and MySQL client are not included in VPS template by default, thus they have to be installed additionally. According to PVC aspect, the following section describes the template in metadata, where :

```
<content xmlns:pvc="http://apstandard.com/ns/1/pvc">
             <pvc:templates class="lin">
                    <pvc:archive-root path="/var/www/html/SugarCE-Full"/>
                    <pvc:template
path="templates/vzpem-sugarcrm-as4-template-20100410-1.0-1.i386.rpm"/>
                    <pvc:template filename="vzpem-php5-cgi-as4"/>
                    <pvc:template filename="vzpem-php-as4"/>
                    <pvc:template filename="vzpem-mysql-client-as4"/>
             </pvc:templates>
</content>
```

In example above:

- vzpem-sugarcrm-as4-template-20100410-1.0-1.i386.rpm is the PVC template with SugarCRM application
- vzpem-php5-cgi-as4, vzpem-php-as4, vzpem-mysql-client-as4 are names of PVC templates for PHP and MySQL. They have pre-created PVC templates and, thus, should not to be included in APS package.

2. Model of service provisioning

The application is managed by administrator (top-level user) with access to each application module. It is possible to create more than one such user within one application instance. Administrator enables users and regulates their access rights. Still, all data is shared between users and stored within the application instance. So, Sugar Pro is a single-tenant or simple application and application files are simply copied to location dedicated to customer.

3. Services hierarchy

Sugar CRM allows creating multiple users within single application instance. Let's write down it in the Specification terms:

```
<service id="vhost">
      <service id="account" class="account">
      </service>
</service>
```

4. Used technologies

Sugar CRM requires PHP 5 or higher (PHP is pre-installed as PVC template), MySQL 5 or higher and PVC environment on Linux. Let's write down the last 2 requirements in the Specification terms:

```
      <requirements xmlns:env="http://apstandard.com/ns/1/environment"
                    xmlns:php="http://apstandard.com/ns/1/php"
                    xmlns:db="http://apstandard.com/ns/1/db">
             <db:db>
                   <db:id>main</db:id>
                   <db:default-name>sugarce</db:default-name>
                   <db:can-use-tables-prefix>false</db:can-use-tables-prefix>
                   <db:server-type>mysql</db:server-type>
                   <db:server-min-version>5.0</db:server-min-version>
             </db:db>
            <env:environment xmlns:env="http://apstandard.com/ns/1/environment">
                   <env:linux>redhat-as4</env:linux>
            </env:environment>
      </requirements>
```

5. Presentation details and settings

We defined earlier that SugarCRM provides two services - instance and account. Application requires the following settings on installation: 'admin' user login and password, system name to be displayed in the browser title bar, whether application sends usage statistics and how checks for upgrade are performed. The second-level service - account - requires user login and password and profile information. APS allows assigning the `class` attribute to setting or setting group. This attribute inform Controller about the setting meaning. The Controller may decide whether to prompt customer or use some pre-defined values, basing on attribute meaning. Let's group these settings for better presentation, and write down it in the Specification terms. Here, we demonstrate only instance settings, to view the whole set, refer to the **Appendix B. Sugar CRM Sample Metadata** section ("SugarCRM Sample Metadata File" on page 53).

```
<service id="vhost">
             <settings>
                <group class="authn">
                    <name>Administrator's preferences</name>
                    <setting id="admin_name" class="login" type="string"
default-value="admin" min-length="1" max-length="32" regex="^[a-zA-Z][0-9a-zA-Z_\
                        <name>Administrator's Login</name>
                    </setting>
                    <setting id="admin_password" class="password" type="password">
                        <name>Administrator's Password</name>
                    </setting>
                </group>
                <group class="web">
                    <setting id="title" class="title" type="string"
default-value="SugarCRM" min-length="1">
                        <name>System Name</name>
                    </setting>
```

```
                </group>
                    <setting id="send_usage_statistics" type="enum" default-value="true"
installation-only="true">
                                    <name>Send Anonymous Usage Statistics</name>
                             <choice id="true">
                                  <name>Yes</name>
                             </choice>
                                    <choice id="false">
                                 <name>No</name>
                             </choice>
                     </setting>
            </settings>
</service>
```

6. Content delivery method

Sugar CRM is a simple application and should be accessible via Internet. So, delivery method is *Inside package* and URL mapping rules should be declared. URL mapping rules declaration is needed only for "vhost" service. Let's write down URL mapping rules and specify required amount of disk space in the Specification terms:

```
      <url-mapping>
            <default-prefix>/</default-prefix>
         <mapping url="/" path="." xmlns:php="http://apstandard.com/ns/1/php">
                   <php:permissions writable="true"/>
               <php:handler>
                  <php:extension>php</php:extension>
               </php:handler>
            </mapping>
      </url-mapping>
```

7. Draft of metadata file

Now, it is time to make a draft of the application metadata file. Describe application general information in the Specification terms. For details, refer to the **5.1 Common Application Properties** section of the Specification. Add information written down on previous steps and replenish it with details.

8. Configuration script

Sugar CRM provides two services, consider possibility to write separate configuration scripts for each service as in our example. Configuration script should perform all actions required for application configuration using environment variables defined on previous steps. For the list of environment variables, refer to the **Sample Environment Variables** section ("Sample Environment Variables" on page 59) further in this guide.

The application vendor states that any user could not be deleted, the only way to restrict access at all is to disable user. Considering this information, we added `<status-control/>` element to the script of "account" service.

Add information about scripts to the respective `provision` sections of the metadata file. For example,

```
 <service id="account" class="account">
        <provision>
            <configuration-script name="usermanager">
                <script-language>php</script-language>
```

```
                <status-control/>
            </configuration-script>
        </provision>
</service>
```

9. Create archived APS package

Form application package according URL mapping rules defined and requirements in the **4. Basic Package Format** section of the Specification and archive it. In our case, the structure is the following:

APP-META.xml	# Metadata container. XML file.
scripts/	
configure usermanager	# This script will be invoked when application instance is to be setup.
images/	
icon.png ...	# Icon and screenshots of the application
templates/	
vzpem-sugarcrm-as4-template- 20100410-1.0-1.i386.rpm	# SugarCRM files packaged in PVC template

Packaging German Translation Add-on for WordPress

This chapter describes creation of an APS package containing a German translation for Wordpress. Here we will refer to a translation as an add-on – an APS package adding extra functionality to another (master) APS package. You may also use this brief guide to create add-ons for any other applications. Before you create your first APS package, you are recommended to familiarize with the packaging procedure for a master package (see Packaging WordPress Application (on page 22)):

Listed below are the sections of APP-META.xml file supplied with the add-on package:

1. **Type of Environment.** A translation is attached to all instances of Wordpress, therefore it belongs to shared type.
2. **Model of Service Provisioning.** A translation should be provisioned as a single-tenant application, because Wordpress add-ons are configured for each Customer according to his specific demands.
3. **Services hierarchy.** A translation is represented as a single service and appears in the following way:

```
<service id="de-lang">
```

4. **Used technologies.** The add-on does not impose any requirements for any specific technologies, rather relying on the master application (Wordpress).
5. **Presentation details and settings**. The representation details include the supported add-on version, changelog, application category ("Removing Application Instance" on page 82) and the language.

```
<presentation>
	<summary>German translation for WordPress</summary>
	<changelog>
		<version version="2.9.1" release="1">
			<entry>German translation for WordPress is packaged.</entry>
		</version>
	</changelog>
	<categories>
		<category>Theme</category>
	</categories>
	<languages>
		<language>de</language>
	</languages>
</presentation>
```

To switch Wordpress to German, add the following setting:

```
<setting id="replace_current_lang" type="boolean" default-value="true">
	<name>Activate language</name>
</setting>
```

6 **Content delivery.** German translation comprises a number of files which are placed to the same path where WordPress resides. This means that the WordPress add-on belongs to the Inside Package content delivery type.

```
<provision>
	<url-mapping>
		<installed-size>1794048</installed-size>
		<mapping url="/" path="htdocs"></mapping>
	</mapping>
	</url-mapping>
</provision>
```

7 **Metadata file.** Now it is time to make a draft of the add-on application metadata file. Describe application general information in the Specification terms, and also the id of the master package ("http://www.wordpress.org") to which this translation should be applied.

Create an APP-META.xml file and begin with adding general details as follows:

```
<id>http://de.wordpress.org/</id>
       <name>German Translation</name>
       <version>3.0.5</version>
       <release>1</release>
       <homepage>http://de.wordpress.org/</homepage>
       <master-package>
              <package id="http://wordpress.org/">
       </master-package>
       <vendor>
              <name>wordpress</name>
              <homepage>http://de.wordpress.org/</homepage>
       </vendor>
       <packager>
              <name>Parallels</name>
              <homepage>http://parallels.com/</homepage>
              <icon path="images/icon.png" />
              <uri>uuid:714f0a7b-85d6-4eb8-b68e-40f9acbb3103</uri>
       </packager>
```

8 **Editing configuration script.** Create a configuration script that performs configuration and removal of the add-on. Configuration script should perform registration of the translation file in Wordpress. For list of environment variables, refer to the **Sample Environment Variables** (on page 51) section further in this guide.

9 **Archiving application.** Form application package pursuant to the requirements stated in the **4. Basic Package Format** section of the Specification and archive it.

APP-META.xml	# Metadata container. XML file.
scripts/	
configure	# This script will be invoked when the add-on is instantiated.
images/	
icon.png	# Add-on Icon
htdocs/	# Translation files

External Environments

Packaging Open-Xchange Application

The Open-Xchange Hosting Edition is built in a completely modular way that allows service providers to offer a variety of Smart Collaboration solutions - from business e-mail over personal information management (PIM) to a comprehensive collaboration solution. The architecture of the Open-Xchange Hosting Edition enables seamless integration into existing infrastructures with existing tools for authentication, user setup, system administration, accounting and e-mail storage. The Open-Xchange Hosting Edition offers a complete client capability that makes it possible to operate many thousands of customers simultaneously in a virtual server environment. This guarantees customers consistently good efficiency and performance and utilizes hardware resources in the best possible way.

Note: above information is taken from application vendor home page.

Listed below are the sections of an APP-META.xml file supplied with your package:

1 **Type of environment.** The application vendor states that Open-Xchange can seamlessly integrate to any environment. So, we can draw a conclusion that the application is a multi-instance one and may be provisioned to external environment that is not managed by a Controller.

2 **Model of service provisioning.** The application may serve to some number of customers simultaneously in a virtual server environment. Each customer can work in his own environment within this instance – completely separated from each other customer.

3 **Services hierarchy.** Open-Xchange supports multi-tenancy and this means that customers (tenants) are enabled within one application instance. Within each tenant (organization) a number of end-users (staff members) is created. In order to inform the Controller about service destination we assign `class` attribute to each service. Let's write down it in the Specification terms:

```
<service id="context" class="service">
  <service id="account" class="account">
  </service>
</service>
```

4 **Used technologies.** There is no need to provision application itself, so the only requirements we should declare are script language support and mail box for end-users. Let's write down it in the Specification terms:

```
<service id="context" class="service">
 <requirements xmlns:php="http://apstandard.com/ns/1/php">
  <php:version min="5.0" />
  <php:extension>soap</php:extension>
 </requirements>
 <service id="account" class="account">
  <requirements xmlns:mail="http://apstandard.com/ns/1/mail">
   <mail:mailbox>
    <mail:id>account</mail:id>
    <mail:access>
```

```
    <mail:imap/>
    </mail:access>
    <mail:outgoing>
```

<mail:smtp/>

```
    </mail:outgoing>
   </mail:mailbox>
  </requirements>
 </service>
</service>
```

5 **Define resources for application services.** Open-Xchange offers a limited amount of diskspace to store attached documents. Firstly, you will have to add a limiting setting where the resource will refer. After that, you need to create the resource self. And then `filestore` service will look as follows:

```
<service id="filestore">
       <settings>
              <setting type="integer" id="filestore_quota" default-value="300">
              <name>filestore_quota</name>
              <description>Maximum dispkspace that may be used to store user's
media</description>
       </setting>
       </settings>
       <resources>
              <resource class="mb" id="filestore_disk_usage"
limiting-setting="filestore_quota">
                     <name>filestore_disk_usage</name>
                     <description>Diskspace occupied by the contents of a
filestore.</description>
              </resource>
       </resources>
</service>
```

6 **Presentation details and settings.** We defined earlier that there are two services - tenant environment and end-user account. Tenant environment requires the following settings on installation: 'administrator login and password, and profile information. The second-level service - account - requires user login, password and profile information. When end-user logs in application it is necessary to provide tenant credentials along with user ones. Therefore, we have to hide tenant's credentials from the user. APS allows assigning the `class` attribute to a setting or a setting group. This attribute informs Controller about the setting purpose. The Controller may decide whether to prompt customer or use some pre-defined values, basing on attribute meaning. Let's group these settings for better presentation, and write down it in the Specification terms. Here, we demonstrate only tenant settings. to view the whole set, refer to the **Appendix C. Open-Xchange Sample Metadata** section ("Open-Xchange Sample Metadata File" on page 65).

```
<service id="context" class="service">
 <settings>
  <group class="authn">
   <setting id="admin_login" class="login"
            type="email" installation-only="true"
            default-value="admin">
    <name>Administrator login</name>
   </setting>
   <setting id="admin_password" class="password"
            type="password" track-old-value="true"
            min-length="1">
    <name>Administrator password</name>
   </setting>
  </group>
  <group class="vcard">
   <group class="email">
    <setting id="admin_email" class="value" type="email">
     <name>Administrator primary email address</name>
    </setting>
```

```
   </group>
   <group class="fn n">
    <setting id="admin_given_name" class="given-name" type="string"
             min-length="1">
     <name>Administrator given name</name>
    </setting>
    <setting id="admin_surname"
             class="family-name" type="string"
             min-length="1">
     <name>Administrator surname</name>
    </setting>
   </group>
    <setting id="organization_name" class="organization-name"
            default-value="" type="hidden">
     <name>Organization</name>
    </setting>
  </group>
  <setting id="filestore_quota" type="hidden"
          default-value="1024">
   <name>Open-Xchange context wide filestore quota (in MB)
   </name>
  </setting>
 </settings>
</service>
```

Open-Xchange comes in three editions - each with a specific set of features. Having a provisioned application, a tenant may select either edition for any specific user, whereas the user must not have such option. To make this possible, the service of `account` class must have a special hidden setting. The setting value will determine the edition of Open-Xchange enabled for a specific user.

```
<service id="account" class="account">
<setting id="ox_module_access" type="string" visibility="hidden"
default-value="groupware">
       <name>Open-Xchange administrator access level</name>
              <description>
              This is access level of Open-Xchange context administrator account.
              Valid values are 'webmail_plus', 'pim_plus', 'groupware',
              or any other defined in ModuleAccessDefinitions.properties file.
              </description>
</setting>
</service>
```

7 **Content delivery method.** Open-Xchange integrates into existing infrastructures by itself. So, it is provisioned like external service. When Controller provisions external services, it need to know service installation host, DNS name or IP address and login credentials. These settings affects for all instances of services and should be set prior to any service provisioning. Such settings are application global settings. Let's write down it in the Specification terms:

```
<global-settings>
 <setting id="ox_host" class="title" type="string"
          default-value="" min-length="1">
  <name>Open-Xchange installation host</name>
  <description>This is DNS name or IP address of
               Open-Xchange installation, used for
               provisioning access.
  </description>
 </setting>
 <setting id="ox_site" class="title" type="string"
          default-value="" min-length="1">
  <name>Open-Xchange public site address</name>
  <description>This is DNS name or IP address of
               Open-Xchange public site address.
  </description>
 </setting>
 <setting id="ox_master_admin" class="title" type="string" installation-only="true"
uniq="global">
  <name>Master Administrator Login</name>
  <error-message>Please make sure the text you entered
                 starts with a letter and continues
                 with either numbers, letters,
                 underscores or hyphens.
  </error-message>
 </setting>
 <setting id="ox_master_password" class="title" type="password">
  <name>Master Administrator Password</name>
</setting>
</global-settings>
```

Note the ox_master_admin has a uniq attribute which indicates that the unicity of this value is ensured by the Controller. This attribute may have the following values defining the scope of unicity:

- `global` - the setting value should be unique for all application instances provisioned in all global contexts.
- application - the setting value should be unique for all application instances provisioned in the global context.
- owner - the setting value should be unique for all application instances of the customer.
- service - the setting value should be unique for all service instances provisioned in the application instance.

8 **Draft of metadata file.** Now, it is time to make a draft of the application metadata file. Describe application general information in the Specification terms. For details, refer to the **5.1 Common Application Properties** section of the Specification. Add information written down on previous steps and replenish it with details.

9 **Configuration script.** Consider possibility to write separate configuration scripts for each service as in our example. Configuration script should perform all actions required for application configuration using environment variables defined on previous steps. Script for second-level service should configure user mailbox additionally. For the list of environment variables, refer to the **Sample Environment Variables** section (**"Sample Environment Variables" on page 70**) further in this guide.

The application vendor states that any tenant/user could not be deleted, the only way to restrict access at all is to disable tenant/user. Considering this information, we added `<status-control/>` element to both scripts.

Add information about scripts to the respective `provision` sections of the metadata file. For example,

```
<service id="context" class="service">
 <provision>
  <configuration-script name="configure.php">
   <script-language>php</script-language>
   <status-control/>
  </configuration-script>
 </provision>
 <service id="account" class="account">
  <provision>
   <configuration-script name="configure-mbox.php">
  <script-language>php</script-language>
    <status-control/>
   </configuration-script>
  </provision>
 </service>
</service>
```

10 **Archiving application.** Form application package requirements in the **4. Basic Package Format** section of the Specification and archive it. The application is provisioned as external service, so application files themselves are not required. In our case, the structure is the following:

`APP-META.xml`	# Metadata container XML-file.
`sctipts/`	

`configure`
`configure-mbox.php`
`elogin.php`
	# The scripts will be invoked when upon service setup.

`images/`	
`ox_calendar.jpg`	# Screenshots of the application

Packaging SpamExperts Incoming Email Security Firewall Application

SpamExperts Incoming Email Security Firewall is an ISP level malware filtering solution used to protect provide a clean pipe for end Customers. SpamExperts Incoming Email Security Firewall comes as an integration module which redirects all incoming mail traffic to an external server. This chapter will guide you through the process of packaging SpamExperts Incoming Email Security Firewall.

Packaging procedure:

1 **Type of environment.** SpamExperts Incoming Email Security Firewall is not managed by the Controller and is provisioned by the configuration script. Therefore, SpamExperts Incoming Email Security Firewall belongs to the External Environment provisioning type.

2 **Service provisioning.** SpamExperts Incoming Email Security Firewall operates under multiple tenants, and therefore it is a multi-tenant application.

3 **Services hierarchy.** All functions of SpamExperts integration module are represented as a single service:

```
<service id="main" class="service">
```

4 **Used technologies.** SpamExperts Incoming Email Security Firewall redirects all incoming mail traffic to an intermediate mail service for examination and then returns it back to the source domain. To make it possible, a DNS substitute has to be specified for the current DNS record. The latter one will obtain the value of the global setting thus making it possible for the hosting provider to define host of the external mail service.

```
<global-settings>
<group>
	<name>Virtual MX record</name>
	<setting id="mx1" class="title" type="string">
		<name>Primary MX</name>
		</setting>
	</group>
</global-settings>
<requirements xmlns:php="http://apstandard.com/ns/1/php"
xmlns:dns="http://apstandard.com/ns/1/dns">
<php:version min="5.2" />
<php:extension>openssl</php:extension>
<php:extension>curl</php:extension>
<dns:record>
	<dns:id>MX1</dns:id>
		<dns:mx>
			<dns:src>
				<dns:external value-of-setting="domains"/>
			</dns:src>
			<dns:priority>10</dns:priority>
			<dns:dst>
				<dns:external value-of-setting="mx1"/>
			</dns:dst>
			<dns:substitute/>
		</dns:mx>
</dns:record>
</requirements>
```

Presentation details and settings. Among common presentation settings SpamExperts Incoming Email Security Firewall should prompt for domains where mail traffic will be filtered by the application. To make this possible, you will have to define a setting with a domain-name class and a list type. The following is an example of a setting which holds multiple domain names:

```
      <setting id="domains" class="domain-name" visibility="hidden" protected="true"
type="list"
      element-type="string" track-old-value="true">
                  <name>List of domains</name>
                        <description>
                                    Domain names to provision.
                        </description>
            </setting>
```

Note that the setting is protected and invisible, which means that it is not displayed in the Control Panel, cannot be edited and its value is initialized by the configuration script.

1 **Content Delivery Method.** SpamExperts Incoming Email Security Firewall belongs to 'Inside Package' type and requires no processing prior to application provisioning. To provision an external service, the Controller requires an installation host, a DNS name (or IP address) and login credentials. These settings are used across all instances of the application and should be defined prior to any service provisioning. The following shows how to define such global settings.

```
<global-settings>
        <group>
             <name>API Settings</name>
             <setting id="apihost" class="title" type="string">
                 <name>API hostname</name>
                 <description>
                    This is DNS name or IP address of SpamExperts product installation, used
for API access.
                 </description>
             </setting>
             <setting id="apiuser" class="title" type="string" default-value="admin">
                 <name>API username</name>
                 <error-message>
                    Please make sure the text you entered starts with a letter and continues
with either numbers, letters, underscores or hyphens.
                 </error-message>
             </setting>
             <setting id="apipass" class="title" type="password">
                 <name>API password</name>
             </setting>
        </group>
    </global-settings>
```

2 **Draft of metadata file**. Now, it is time to make a draft of the application metadata file. Describe application general information in the Specification terms. For details, refer to the **5.1 Common Application** Properties section of the Specification. Add information written down on previous steps and replenish it with details.

3 **Preparing scripts**. To pass the values to the SpamExperts Incoming Email Security Firewall, a configuration script is used which, among other things, must start and stop the service and ensure that all filtered mail traffic is returned back to the initial mail service. The hostname of the initial mail service may be taken from DNS_<id>_SUBSTITUTE environment variable, where <id> is the id of the original DNS record.

4 **Archiving application.** Form application package according to the requirements stated in the **4. Basic Package Format** section of the Specification and archive it.

APP-META.xml	# Metadata container. XML file.
scripts/	
configure.php	# This script will be invoked when Application is instantiated or reconfigured.
images/	
prospamfilter2.png	# Application Icon
screenshot.png	# Application screenshot

Additional Scenarios

Integrating Applications with Upsell Services

Many applications have different editions where additional features may be provided as an up-sell for the features provisioned initially. Likewise, a hosting provider may offer upgrades of, for example, “Standard” edition on “Premium” one for existing application instance. This chapter will outline a way for packaging an application with the feature up-sell capability. It is assumed here that you already have a valid APS package to add upsell features to.

Changing of application feature set is done by performing reconfiguration procedure of application service. Controller delivers the new feature set in value of special setting which cannot be modified by application user directly. Hosting provider defines editions (feature sets) of application service with different values of the setting and a workflow, following which, a user is able to purchase a configuration and to assign it to a service instance.

Packaging procedure:

1 **Add Upsell setting in Metadata.** Declare a singular service with hidden setting for feature set. Note that this service should be included in a parent application service. Here's how it's done.

```
<service>
      <!-- parent application service -->
      <service id="edition_type" singular="true" class="service">
            <presentation>
            <name>Service Package</name>
            <summary>Types of available services</summary>
      </presentation>
            <settings>
      <setting type="enum" id="feature-set" visibility="hidden" default-value="basic">
                  <name>Feature set</name>
            <choice id="Basic">
                  <name>Basic Edition</name>
            </choice>
            <choice id="Gold">
                  <name>Gold Edition</name>
            </choice>
            <choice id="Premium">
           <name>Premium Edition</name>
               </choice>
      </setting>
      </settings>
      </service>
      ..
</service>
```

2 **Modify configuration script.** Edit the configuration script so that it would switch the application edition according to the value of SETTING_EDITION environment value.

Integration with Domains

Some applications need one or multiple domains for various purpose, for example, to create a virtual host or to provide email or spam filer service for these domains.

Domains are received by application via setting with class="domain-name". There are two cases:

1. Application needs single domain.

To receive selected domain name in application, add setting with class="domain-name" and type="domain-name". Setting must be 'hidden' and 'protected'. When being installed, application gets one domain and the setting is filled with the domain name.

Example:

```
<setting id="shop_domain" class="domain-name" visibility="hidden" protected="true"
type="domain-name">
       <name>Shop domain</name>
</setting>
```

2. Application can use multiple domains

Add 'hidden' setting of class="domain-name" and type="list". When being installed, application gets list of domains selected by user and the setting is filled with domain names. If user adds new domain to application or remove an existing one, application is reconfigured and the setting is updated with new list of domains. It's recommended to use "track-old-value" attribute to get old list of domains in provisioning scripts.

Example:

```
<setting id="domains" class="domain-name" visibility="hidden" protected="true"
type="list" element-type="string" track-old-value="true">
       <name>List of domains</name>
</setting>
```

Application receives domains list in a set of environment variables SETTINGS_<setting_id>_1, SETTINGS_<setting_id>_2, etc and old domain list OLDSETTINGS_domains_1, OLDSETTINGS_domains_2, OLDSETTINGS_domains_3.

Example:

```
SETTINGS_domains_1=domain1.com
SETTINGS_domains_2=domain2.com
OLDSETTINGS_domains_1=domain1.com
OLDSETTINGS_domains_2=domain2.com
OLDSETTINGS_domains_3=olddomain3.com
```

Creation of DNS records

Controller can create or replace DNS records on domains which are integrated with application. Let's consider an example where existing DNS record of MX type is replaced with application's one and old MX record is communicated to application. Given that application has integration with multiple domains ('list' domain setting) and needs that MX records to be placed in each of domain from the list.

To achieve it application should declare requirement for DNS record and refer to domain setting in it using `<dns:external />` element. To replace existing record with a new one, `<dns:substitute/>` element must be used.

Example:

```
Domain setting:
<setting type="list" id="domains" protected="true" visibility="hidden"
element-type="domain-name" class="domain-name" track-old-value="true" >
       <name>Domains</name>
</setting>
DNS requirement:
<dns:record>
       <dns:id>MX1</dns:id>
       <dns:mx>
              <dns:src>
                     <dns:external value-of-setting="domains"/>
              </dns:src>
              <dns:priority>10</dns:priority>
              <dns:dst>
                     <dns:external value-of-setting="mx1"/>
              </dns:dst>
              <dns:substitute/>
       </dns:mx>
</dns:record>
```

Old value of DNS record is passed to application via `DNS_<id>_SUBSTITUTE` environment variable, see APS DNS Aspect http://www.apsstandard.org/r/doc/package-format-specification-1.2/index.html#s.aspects.dns for details. In case of 'list' domain setting, this variable is extended to a set of `DNS_<id>_SUBSTITUTE_<domain name>` variables. See below an example of configuration script's environment variables where MX records 'old.mx.com' and 'old.mx2.com' are replaced for 'domain1.com' and 'domain2.com' correspondently.

Example:

```
'DNS_MX1_SUBSTITUTE_domain1.com=old.mx.com.'
'DNS_MX1_SUBSTITUTE_domain2.com=old.mx2.com.'
```

If there are multiple MX records on a domain, then all existing ones are communicated to application in form `DNS_<id>_SUBSTITUTE_<domain name>_<num>` environment variables, where `<num>` is number of record. This list is ordered by MX record priority (more prioritized go first). Record with the highest priority also gets into `DNS_MX1_SUBSTITUTE_<domain name>` and `DNS_MX1_SUBSTITUTE` variables.

Example:

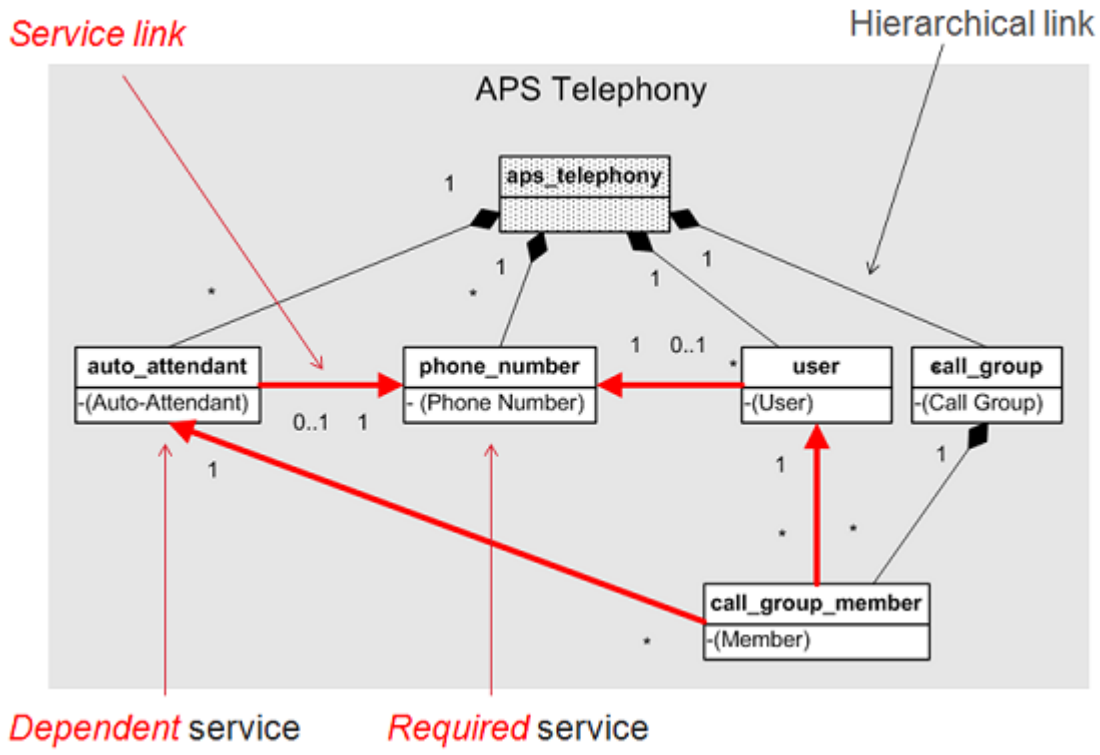
```
DNS_MX1_SUBSTITUTE=old.mx.com
```

```
DNS_MX1_SUBSTITUTE_domain1.com=old.mx.com
DNS_MX1_SUBSTITUTE_domain1.com_1=old.mx.com
DNS_MX1_SUBSTITUTE_domain1.com_2=old.mx2.com
DNS_MX1_SUBSTITUTE_domain1.com_3=old.mx3.com
```

Integration between Services

Application may allow different services to be be linked with each other. The link between services means that one service can use another one. For example a hosted PBX application may link phone numbers with auto-attendants, users with call groups and events with schedules. Another example is a Mail application where mailboxes can be linked with distribution lists and anti-spam or anti-virus service. Which types of services can be linked together is defined in APS package metadata, via **APS Service Aspect**.

Let's consider and example of "APS Telephony" hosted PBX application where each user needs a phone number.



It's described in APP-META.xml as **user** service which declares requirement of **phone number** one using service **class**.

```
<service id="phone_number" class="phone_number">
</service>

<service id="user" class="account">

	<requirements xmlns:svc="http://apstandard.com/ns/1/service">
		<svc:service>
			<svc:id>phone_number</svc:id>
			<svc:class>phone_number</svc:class>
			<svc:name>Phone Number</svc:name>
			<svc:unique uuid="abc"/>
		</svc:service>
	</requirements>
</service>
```

When the service requirement is satisfied, configuration script of "user" services gets settings of certain instance of "phone_number" one via environment variables in form

```
SERVICE_<requirement-id>_SETTINGS_<setting-id1>,
SERVICE_<requirement-id>_SETTINGS_<setting-id2>,
...
SERVICE_<requirement-id>_SETTINGS_<setting-idN>,
```

where **<requirement_id>** id the **svc:id** from requirement metadata and **<setting-id>** the ID of required service setting. See **APS Service Aspect** for more details.

APPENDIX A

WordPress Sample Metadata File

This appendix demonstrates the metadata file of the *WordPress* application package. The following metadata file is extended with delimiters that separate file fragments added on different steps of the metadata file creation described in the **Packaging WordPress Application** section ("Packaging WordPress Application" on page 22) earlier in this guide.

```
<!-- Application namespaces and APS version (step 9) -->
<application xmlns="http://apstandard.com/ns/1" version="1.2">
<!-- Application common properties (step 9) -->
 <id>http://wordpress.org/</id>
 <name>WordPress</name>
 <version>2.7.1</version>
 <release>1</release>
 <homepage>http://wordpress.org/</homepage>
 <vendor>
  <name>WordPress.org</name>
  <homepage>http://wordpress.org/</homepage>
  <icon path="images/icon.png"/>
 </vendor>
 <packager>
  <name>Parallels</name>
  <homepage>http://parallels.com</homepage>
  <uri>uuid:714f0a7b-85d6-4eb8-b68e-40f9acbb3103</uri>
 </packager>
 <presentation>
  <summary>WordPress is a state-of-the-art semantic personal
           publishing platform with a focus on aesthetics, web
           standards, and usability.
  </summary>
  <description>
      WordPress is a state-of-the-art semantic personal
      publishing platform with a focus on aesthetics,
      web standards, and usability. What a mouthful.
      WordPress is both free and priceless at the same time.
      More simply, WordPress is what you use when you want
      to work with your blogging software, not fight it.
  </description>
  <icon path="images/icon.png"/>
  <screenshot path="images/admin_page.jpg">
   <description>Admin page.</description>
  </screenshot>
  <changelog>
   <version version="2.7.0" release="5">
    <entry>'PHP safe mode off' requirement is added.</entry>
   </version>
  </changelog>
  <categories>
   <category>Web/Blog</category>
   <category>Personal/Blog</category>
  </categories>
  <languages>
   <language>en</language>
  </languages>
```

```
 </presentation>
 <patch match="/application/version = '2.5.1'
        and /application/release = '4'"/>
 <upgrade match="/application/version = '2.0'"/>
<!-- Application service (step 4) -->
 <service id="blog">
<!-- Service presentation properties (step 7) -->
  <license must-accept="true">
   <text>
    <name>GPLv2</name>
    <file>htdocs/license.txt</file>
   </text>
  </license>
  <presentation>
   <name>Blog</name>
   <entry-points>
    <entry class="control-panel" dst="/wp-admin/" method="POST">
      Administrative Interface
    </entry>
    <entry class="frontpage" dst="/">Application entry point</entry>
   </entry-points>
  </presentation>
<!-- Service settings (step 7) -->
  <settings>
   <group>
    <name>Administrator's preferences</name>
    <setting id="admin_name" type="string"
             default-value="admin" min-length="1"
             max-length="32" regex="^[a-zA-Z][0-9a-zA-Z_]*">
     <name>Administrator's login</name>
     <error-message>Please make sure the text you entered
           starts with a letter and continues with either numbers,
           letters, underscores or hyphens.
     </error-message>
    </setting>
    <setting id="admin_password"
             type="password" min-length="1" >
     <name>Password</name>
    </setting>
    <setting id="admin_email" type="email">
     <name>Administrator's email</name>
    </setting>
   </group>
   <group>
    <name>Weblog's preferences</name>
     <setting id="title" type="string" min-length="1">
      <name>Weblog title</name>
     </setting>
   </group>
   <group>
    <name>Other preferences</name>
     <setting id="locale" type="enum" default-value="en-US">
      <name>Interface language</name>
      <choice id="en-US" >
       <name>English</name>
      </choice>
      <choice id="de-DE" >
       <name>German</name>
      </choice>
     </setting>
   </group>
  </settings>
<!-- Service used technologies (step 6) -->
```

```
  <requirements xmlns:php="http://apstandard.com/ns/1/php"
                xmlns:db="http://apstandard.com/ns/1/db">
   <php:version min="4.2.0"/>
   <php:extension>mysql</php:extension>
   <php:safe-mode>false</php:safe-mode>
   <db:db>
    <db:id>main</db:id>
    <db:default-name>wordpress</db:default-name>
    <db:can-use-tables-prefix>false
    </db:can-use-tables-prefix>
    <db:server-type>mysql</db:server-type>
    <db:server-min-version>4.0.0</db:server-min-version>
   </db:db>
  </requirements>
 <!-- Content delivery settings (step 8) -->
  <provision>
   <url-mapping>
    <default-prefix>wordpress</default-prefix>
    <installed-size>6696960</installed-size>
    <mapping url="/" path="htdocs"
             xmlns:php="http://apstandard.com/ns/1/php">
     <php:handler>
      <php:extension>php</php:extension>
     </php:handler>
     <mapping url="blogs/media">
      <php:permissions writable="true"/>
     </mapping>
     <mapping url="wp-content">
      <php:permissions writable="true"/>
     </mapping>
     <mapping url="tmp">
      <php:permissions writable="true"/>
     </mapping>
    </mapping>
   </url-mapping>
  <!-- Service configuration script declaration (step 10) -->
   <configuration-script name="configure">
    <script-language>php</script-language>
   </configuration-script>
  </provision>
 </service>
</application>
```

Sample Environment Variables

The types of environment variables that are passed to the *WordPress* configuration script by a Controller are as follows:

- Variables defined by URL-mappings (see **5.3.2.2.1. URL Mapping Variables** section of the Specification).

  BASE_URL_SCHEME

  BASE_URL_HOST

  BASE_URL_PORT

  BASE_URL_PATH

  WEB___DIR

WEB__blogs_media_DIR

WEB__wp-content_DIR

WEB__tmp_DIR

- Variables defined by application settings (see the **5.3.2.2.2. Settings** section of the Specification).

  SETTINGS_admin_name

  SETTINGS_admin_password

  SETTINGS_admin_email

  SETTINGS_title

  SETTINGS_locale

- Aspects-defined (see the **7.4. Environment Variables** section of the Specification).
  - **PHP aspect**

    PHP_VERSION

  - **Database aspect**

    DB_main_TYPE

    DB_main_NAME

    DB_main_LOGIN

    DB_main_PASSWORD

    DB_main_HOST

    DB_main_PORT

    DB_main_VERSION

    DB_main_PREFIX

APPENDIX B

SugarCRM Sample Metadata File

This appendix demonstrates the metadata file of the *SugarCRM* application package. The following metadata file is extended with delimiters that separate file fragments added on different steps of the metadata file creation described in the **Packaging SugarCRM Application** section ("Packaging SugarCRM Application" on page 28) earlier in this guide.

```
<application xmlns="http://apstandard.com/ns/1" version="1.2">
       <!-- Application namespaces and APS version (step 7) -->
       <id>http://apsstandard.org/app/SugarCRM-pvc</id>
    <name>SugarCRM-pvc</name>
    <version>5.5.1</version>
    <release>5</release>
       <homepage>http://www.sugarcrm.com</homepage>
    <vendor>
        <name>SugarCRM Inc.</name>

<homepage>http://www.sugarcrm.com/crm/about/about-sugarcrm.html</homepage>
        <icon path="images/icon.png"/>
    </vendor>
    <packager>
        <name>Parallels</name>
        <homepage>http://parallels.com</homepage>
        <uri>uuid:714f0a7b-85d6-4eb8-b68e-40f9acbb3103</uri>
    </packager>
    <presentation>
        <summary>Sugar CRM Community Edition</summary>
        <description>
Sugar CRM Community Edition enables organizations to efficiently organize, populate,
and maintain information on all aspects of their customer relationships. It provides
integrated management of corporate information on customer accounts and contacts,
sales leads and opportunities, plus activities such as calls, meetings, and assigned
tasks. The system seamlessly blends all of the functionality required to manage
information on many aspects of your business into an intuitive and user-friendly
graphical interface.
        </description>
        <icon path="images/icon.png"/>
        <screenshot path="images/screen_home.png"><description>Home
Screen</description></screenshot>
        <screenshot path="images/screen_admin.png"><description>Admin
Screen</description></screenshot>
        <screenshot path="images/screen_dashboard.png"><description>Dashboard
View</description></screenshot>
        <screenshot path="images/screen_accounts.png"><description>Account
Details</description></screenshot>
        <screenshot path="images/screen_campaigns.png"><description>Marketing
Campaigns</description></screenshot>
        <screenshot path="images/screen_bug_tracker.png"><description>Bug
Tracker</description></screenshot>
        <screenshot
path="images/screen_calendar.png"><description>Calendar</description></screenshot
>
```

```
        <screenshot
path="images/screen_documents.png"><description>Documents</description></screensh
ot>
       <changelog>
                     <version version="5.5.1" release="5">
                            <entry>php dependency corrected</entry>
                     </version>
              <version version="5.5.1" release="4">
                            <entry>The Password Management feature enables system
administrators to create and manage system-generated passwords</entry>
                            <entry>Emails module has been redesigned</entry>
                     <entry>Packaged as APS 1.2</entry>
              </version>
              <version version="5.2.0j" release="4">
                     <entry>Packaged as APS 1.1</entry>
              </version>
       </changelog>
        <categories>
            <category>Back office/Customer Relationship Management</category>
        </categories>
        <languages>
            <language>en</language>
        </languages>
    </presentation>


       <!-- Type of environment (step 1) -->
       <content xmlns:pvc="http://apstandard.com/ns/1/pvc">
              <pvc:templates class="lin">
                     <pvc:archive-root path="/var/www/html/SugarCE-Full"/>

                     <pvc:template
path="templates/vzpem-sugarcrm-as4-template-20100410-1.0-1.i386.rpm"/>
                     <pvc:template filename="vzpem-php5-cgi-as4"/>
                     <pvc:template filename="vzpem-php-as4"/>
                     <pvc:template filename="vzpem-mysql-client-as4"/>
                     <pvc:template filename="vzpem-pgsql-client-as4"/>
              </pvc:templates>
       </content>
         <!-- Application services (step 3) -->
         <service id="vhost">
                     <license must-accept="true">
                            <free/>
                            <text>
                                   <name>GPLv3</name>
                                   <file>LICENSE.txt</file>
                            </text>
                     </license>
               <!-- Service presentation and settings(step 5) -->
             <presentation>
                 <name>SugarCRM Instance</name>
                 <infolinks>
                     <link class="official"
href="http://www.sugarcrm.com/">Official site</link>
                     <link class="support"
href="http://www.sugarcrm.com/crm/support">Support and training</link>
                     <link class="demo"
href="http://www.sugarcrm.com/crm/demo/daily-demo.html">Demo</link>
                 </infolinks>
                 <entry-points>
                     <entry class="control-panel" dst="/index.php" method="post">
                         <label>Application entry</label>
                         <variable name="module">Users</variable>
                         <variable name="action">Authenticate</variable>
```

```
                        <variable name="return_module">Users</variable>
                        <variable name="return_action">Login</variable>
                        <variable name="cant_login"></variable>
                        <variable name="login_module"></variable>
                        <variable name="login_action"></variable>
                        <variable name="login_record"></variable>
                        <variable name="user_name" class="login"
value-of-setting="admin_name"/>
                        <variable name="user_password" class="password"
value-of-setting="admin_password"/>
                        <variable name="login_theme">Sugar</variable>
                        <variable name="login_language">en_us</variable>
                        <variable name="Login">++Login++</variable>
                    </entry>
                </entry-points>
            </presentation>

            <settings>
                <group class="authn">
                    <name>Administrator's preferences</name>
                    <setting id="admin_name" class="login" type="string"
default-value="admin" min-length="1" max-length="32" regex="^[a-zA-Z][0-9a-zA-Z_\>
                        <name>Administrator's Login</name>
                        <error-message>Please make sure the text you entered starts
with a letter and continues with either numbers, letters, underscores or
hyphens.</error-message>
                    </setting>
                    <setting id="admin_password" class="password" type="password">
                        <name>Administrator's Password</name>
                    </setting>
                </group>
                <group class="web">
                    <setting id="title" class="title" type="string"
default-value="SugarCRM" min-length="1">
                        <name>System Name</name>
                        <description>This name will be displayed in the browser title
bar when users visit the Sugar application.</description>
                    </setting>
                </group>
                            <setting id="send_usage_statistics" type="enum"
default-value="true" installation-only="true">
                    <name>Send Anonymous Usage Statistics</name>
                    <description>If selected 'Yes', Sugar will send anonymous
statistics about your installation to SugarCRM Inc. every time your system checks for
new versions.</description>
                    <choice id="true">
                        <name>Yes</name>
                    </choice>
                    <choice id="false">
                        <name>No</name>
                    </choice>
                </setting>

                <!-- setting id="check_for_updates" type="hidden"
default-value="manual">
                    <name>Check For Updates</name>
                    <description>How the system will check for updated versions of
the application.</description>
                   <choice id="manual">
                        <name>Manual</name>
                    </choice>
                    <choice id="automatic">
                        <name>Automatic</name>
```

```
                    </choice>
                </setting -->
            </settings>

             <!-- Used technologies (step 4) -->
            <requirements xmlns:env="http://apstandard.com/ns/1/environment"
xmlns:php="http://apstandard.com/ns/1/php"
xmlns:db="http://apstandard.com/ns/1/db">
                        <env:environment
xmlns:env="http://apstandard.com/ns/1/environment">
                                <env:linux>redhat-as4</env:linux>
                        </env:environment>
                <db:db>
                    <db:id>main</db:id>
                    <db:default-name>sugarce</db:default-name>
                    <db:can-use-tables-prefix>false</db:can-use-tables-prefix>
                    <db:server-type>mysql</db:server-type>
                    <db:server-min-version>5.0</db:server-min-version>
                </db:db>
            </requirements>

            <provision>
                    <!-- Content delivery method (step 6) -->
                <url-mapping>
                    <default-prefix>/</default-prefix>
                    <mapping url="/" path="."
xmlns:php="http://apstandard.com/ns/1/php">
                                        <php:permissions writable="true"/>
                        <php:handler>
                            <php:extension>php</php:extension>
                        </php:handler>

                        <php:permissions writable="true"/>
                        <mapping url="tmp"><php:permissions
writable="true"/></mapping>
                        <mapping url="cache"><php:permissions
writable="true"/></mapping>
                        <mapping url="custom"><php:permissions
writable="true"/></mapping>
                        <mapping url="data"><php:permissions
writable="true"/></mapping>
                        <mapping url="modules"><php:permissions
writable="true"/></mapping>
                    </mapping>
                </url-mapping>

                    <!-- Configuration script (step 8) -->
                <configuration-script name="configure">
                    <script-language>php</script-language>
                </configuration-script>
            </provision>

            <service id="account" class="account">
                <presentation>
                    <name>SugarCRM Account</name>
                    <entry-points>
                        <entry class="control-panel" dst="/index.php"
method="post">
                            <label>Application entry</label>
                            <variable name="module">Users</variable>
                            <variable name="action">Authenticate</variable>
                            <variable name="return_module">Users</variable>
                            <variable name="return_action">Login</variable>
```

```
                            <variable name="cant_login"></variable>
                            <variable name="login_module"></variable>
                            <variable name="login_action"></variable>
                            <variable name="login_record"></variable>
                            <variable name="user_name" class="login"
value-of-setting="user_login"/>
                            <variable name="user_password" class="password"
value-of-setting="user_password"/>
                            <variable name="login_theme">Sugar</variable>
                            <variable name="login_language">en_us</variable>
                            <variable name="Login">++Login++</variable>
                        </entry>
                    </entry-points>
                </presentation>

                <settings>
                    <group class="authn">
                        <name>Account preferences</name>
                        <setting id="user_login" class="login"
track-old-value="true" type="string" min-length="3" max-length="60"
regex="^[a-zA-Z][0-9a-zA-Z_@\\.]*">
                            <name>Account's Login</name>
                        </setting>
                        <setting id="user_password" class="password"
type="password" min-length="4">
                            <name>Account's Password</name>
                        </setting>
                    </group>
                    <group class="vcard">
                        <group class="fn n">
                            <setting id="first_name" class="given-name"
type="string" max-length="30">
                                <name>First Name</name>
                            </setting>
                            <setting id="last_name" class="family-name"
type="string" max-length="30">
                                <name>Last Name</name>
                            </setting>
                        </group>
                        <group class="email">
                            <setting id="user_email" class="value" type="email"
optional="true">
                                <name>Email</name>
                            </setting>
                        </group>
                        <setting id="title" class="title" type="string">
                            <name>Title</name>
                        </setting>
                        <setting id="department" class="organization-unit"
type="string">
                            <name>Department</name>
                        </setting>
                        <group class="tel">
                            <name>work</name>
                            <setting id="phone_work" class="value" type="string">
                                <name>Work Phone Number</name>
                            </setting>
                        </group>
                        <group class="tel">
                            <name>cell</name>
                            <setting id="phone_mobile" class="value" type="string">
                                <name>Mobile Phone Number</name>
                            </setting>
```

```
                        </group>
                        <group class="tel">
                            <name>fax</name>
                            <setting id="phone_fax" class="value" type="string">
                                <name>Fax Number</name>
                            </setting>
                        </group>
                        <group class="tel">
                            <name>home</name>
                            <setting id="phone_home" class="value" type="string">
                                <name>Home Phone Number</name>
                            </setting>
                        </group>
                        <setting id="address_street" class="street-address"
type="string">
                            <name>Street</name>
                        </setting>
                        <setting id="address_city" class="locality" type="string">
                            <name>City</name>
                        </setting>
                        <setting id="address_state" class="region" type="string">
                            <name>Region</name>
                        </setting>
                        <setting id="address_country" class="country-name"
type="string">
                            <name>Country</name>
                        </setting>
                        <setting id="address_postalcode" class="postal-code"
type="string">
                            <name>Postal Code</name>
                        </setting>
                        <setting id="description" class="note" type="string">
                            <name>Description</name>
                        </setting>
                    </group>
                </settings>

                    <!-- Configuration script (step 8) -->
                <provision>
                    <configuration-script name="usermanager">
                        <script-language>php</script-language>
                        <status-control/>
                    </configuration-script>
                </provision>
            </service>
        </service>
</application>
```

Sample Environment Variables

The types of environment variables that are passed to the *SugarCRM* configuration scripts by a Controller are as follows:

- Variables are defined by application settings (see the **5.3.2.2.2. Settings** section of the Specification).
- Aspects-defined (see the **6.4. Environment Variables** section of the Specification).

The full list of *SugarCRM* environment variables list is as follows:

'Root' service variables:

- Pre-defined variables representing the instance: URL:

  BASE_URL_SCHEME

  BASE_URL_HOST

  BASE_URL_PORT

  BASE_URL_PATH

- Variables defined by service settings declarations:

  SETTINGS_admin_name

  SETTINGS_admin_password

  SETTINGS_title

  SETTINGS_send_usage_statistics

  SETTINGS_check_for_updates

- Variables defined by service requirements declarations:

  PHP_VERSION

  DB_main_NAME

  DB_main_LOGIN

  DB_main_PASSWORD

  DB_main_HOST

  DB_main_PORT

  DB_main_VERSION

  DB_main_PREFIX

- Variables defined by service URL mapping declarations:

  WEB___DIR

  WEB__cache_DIR

  WEB__custom_DIR

  WEB__data_DIR

  WEB__modules_DIR

  WEB__tmp_DIR

  WEB__config.php_DIR

Child service variables:

- Pre-defined variables representing the instance URL:

  BASE_URL_SCHEME

  BASE_URL_HOST

  BASE_URL_PORT

  BASE_URL_PATH

- Variables defined by service settings declarations:

  SETTINGS_user_login

  OLDSETTINGS_user_login

  SETTINGS_user_password

  SETTINGS_first_name

  SETTINGS_last_name

  SETTINGS_user_email

  SETTINGS_title

  SETTINGS_department

  SETTINGS_phone_work

  SETTINGS_phone_mobile

  SETTINGS_phone_fax

  SETTINGS_phone_home

  SETTINGS_address_street

  SETTINGS_address_city

  SETTINGS_address_state

  SETTINGS_address_country

  SETTINGS_address_postalcode

  SETTINGS_description

- Variables defined by parent service requirements declarations:

  PHP_VERSION

  DB_main_NAME

  DB_main_LOGIN

  DB_main_PASSWORD

  DB_main_HOST

  DB_main_PORT

  DB_main_VERSION

  DB_main_PREFIX

APPENDIX C

WordPress German Translation Add-on Sample Metadata File

This appendix demonstrates the metadata file of the *WordPress* translation package. The following metadata file is extended with delimiters that separate file fragments added on different steps of the metadata file creation described in the **Packaging Add-ons** earlier in this guide.

```
<!-- Application namespaces and APS version (step 1) -->
<application xmlns="http://apstandard.com/ns/1" version="1.2">
<!-- Application common properties (step 1) -->
        <id>http://de.wordpress.org/</id>
        <name>German Translation</name>
        <version>3.0.5</version>
        <release>1</release>
        <homepage>http://de.wordpress.org/</homepage>
        <master-package>
                <package id="http://wordpress.org/"/>
        </master-package>
        <vendor>
                <name>wordpress</name>
                <homepage>http://de.wordpress.org/</homepage>
        </vendor>
        <packager>
                <name>Parallels</name>
                <homepage>http://parallels.com/</homepage>
                <uri>uuid:714f0a7b-85d6-4eb8-b68e-40f9acbb3103</uri>
        </packager>
<!-- Presentation details (step 2) -->
        <presentation>
                <summary>German translation for WordPress</summary>
                <icon path="images/icon.png"/>
                <changelog>
                        <version version="3.0.5" release="1">
                                <entry>German translation for WordPress is
packaged.</entry>
                        </version>
                </changelog>
                <categories>
                        <category>Web/Blog</category>
                </categories>
                <languages>
                        <language>de</language>
                </languages>
        </presentation>
<!-- Application services (step 3) -->
        <service id="ru-lang">
                <presentation>
                        <name>German translation</name>
                        <summary>German localization package for Wordpress</summary>
                </presentation>
                <settings>
```

```
            <setting id="replace_current_lang" type="boolean" default-value="true">
                <name>Activate language</name>
            </setting>
             </settings>
             <provision>
                    <configuration-script name="configure.php">
                           <script-language>php</script-language>
                    </configuration-script>
                    <url-mapping>
                           <installed-size>1794048</installed-size>
                           <mapping url="/" path="htdocs"
xmlns:php="http://apstandard.com/ns/1/php">
                           </mapping>
                    </url-mapping>
             </provision>
      </service>
</application>
```

Sample Environment Variables

The types of environment variables that are passed to a WordPress add-on configuration script by a Controller are as follows:

- Variables defined by URL-mappings (see **5.3.2.2.1. URL Mapping Variables** section of the Specification).
  - `WEB___DIR`
- Variables defined by application settings (see **5.3.2.2.2. Settings** section of the Specification).
  - `SETTINGS_replace_current_lang`

APPENDIX D

Open-Xchange Sample Metadata File

This appendix demonstrates the metadata file of the *Open-Xchange* application package. The following metadata file is extended with delimiters that separate file fragments added on different steps of the metadata file creation described in the **Packaging Open-Xchange Application** section earlier in this guide.

```
<!-- Application namespaces and APS version (step 9) -->
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://apstandard.com/ns/1" version="1.2">
<!-- Application common properties (step 9) -->
 <id>http://www.open-xchange.com/</id>
 <name>Open-Xchange</name>
 <version>6.7</version>
 <release>17</release>
 <homepage>http://www.open-xchange.com/en/products/open-xchange-
           hosting-edition-en
 </homepage>
 <vendor>
  <name>Open-Xchange Inc.</name>
  <homepage>http://www.open-xchange.com</homepage>
  <icon path="images/ox_logo.jpg" />
 </vendor>
 <packager>
  <name>Parallels</name>
  <homepage>http://parallels.com</homepage>
  <uri>uuid:c9fa49b2-ba47-11dd-9fed-00155882361a</uri>
 </packager>
 <presentation>
  <summary>
    Open-Xchange Hosting Edition is a highly advanced
    business-class email and collaboration solution
    for competitive success.
  </summary>
  <description>
     Open-Xchange Hosting Edition is a highly advanced
     business-class email and collaboration solution
     for competitive success. It provides a highly
     advanced, improved and intuitive browser based
     interface that meets all their email, calendaring,
     tasking, contacts and document sharing requirements.
     Each of the functions simplifies the daily work and
     it is the unique linking and integration of the
     different modules that makes organizing and managing
     within the company even smoother, faster, and more
     transparent.
  </description>
  <icon path="images/ox_logo.jpg" />
  <screenshot path="images/ox_portal.jpg">
   <description>Open-Xchange portal page</description>
  </screenshot>
  <screenshot path="images/ox_email.jpg">
   <description>Open-Xchange E-Mail page</description>
  </screenshot>
```

```
  <screenshot path="images/ox_calendar.jpg">
   <description>Open-Xchange calendar page</description>
  </screenshot>
  <screenshot path="images/ox_contact.jpg">
   <description>Open-Xchange contact page</description>
  </screenshot>
  <screenshot path="images/ox_task.jpg">
   <description>Open-Xchange task page</description>
  </screenshot>
  <screenshot path="images/ox_infostore.jpg">
   <description>Open-Xchange infostore page</description>
  </screenshot>
  <changelog>
   <version version="6.7" release="1">
    <entry>Initial package version</entry>
   </version>
  </changelog>
  <categories>
   <category>Collaboration/Email</category>
  </categories>
  <languages>
   <language>en</language>
   <language>de</language>
  </languages>
 </presentation>
<!-- Content delivery settings (step 8) -->
 <global-settings>
  <setting id="ox_host" class="title" type="string"
           default-value="" min-length="1">
   <name>Open-Xchange installation host</name>
   <description>This is DNS name or IP address of
                Open-Xchange installation, used for
                provisioning access.
   </description>
  </setting>
  <setting id="ox_site" class="title" type="string"
           default-value="" min-length="1">
   <name>Open-Xchange public site address</name>
   <description>This is DNS name or IP address of
                Open-Xchange public site address.
   </description>
  </setting>
  <setting id="ox_master_admin" class="title" type="string">
   <name>Master Administrator Login</name>
   <error-message>Please make sure the text you entered
                  starts with a letter and continues
                  with either numbers, letters,
                  underscores or hyphens.
   </error-message>
  </setting>
  <setting id="ox_master_password" class="title" type="password">
   <name>Master Administrator Password</name>
  </setting>
 </global-settings>
<!-- Application services (step 4) -->
 <service id="context" class="service">
<!-- Service presentation properties (step 7) -->
  <license must-accept="true">
   <text>
    <name>End User License Agreement</name>
    <url>http://software.open-xchange.com/
         OX6/doc/OXHE-Provisioning/ar02.html
    </url>
```

```
  </text>
 </license>
 <presentation>
  <entry-points>
   <entry dst="http://{ox_site}/mail/elogin.php" method="POST">
    <label>Open-Xchange context administration</label>
    <variable name="ox_site" value-of-setting="ox_site" />
    <variable name="username" value-of-setting="admin_login" />
    <variable name="password" value-of-setting="admin_password" />
   </entry>
  </entry-points>
 </presentation>
<!-- Service settings (step 7) -->
 <settings>
  <group class="authn">
   <setting id="admin_login" class="login"
            type="email" installation-only="true"
            default-value="admin">
    <name>Administrator login</name>
   </setting>
   <setting id="admin_password" class="password"
            type="password" track-old-value="true"
            min-length="1">
    <name>Administrator password</name>
   </setting>
  </group>
  <group class="vcard">
   <group class="email">
    <setting id="admin_email" class="value" type="email">
     <name>Administrator primary email address</name>
    </setting>
   </group>
   <group class="fn n">
    <setting id="admin_given_name"
             class="given-name" type="string" min-length="1">
     <name>Administrator given name</name>
    </setting>
    <setting id="admin_surname" class="family-name" type="string"
             min-length="1">
     <name>Administrator surname</name>
    </setting>
   </group>
   <setting id="organization_name"
            class="organization-name"
            default-value="" type="hidden">
    <name>Organization</name>
   </setting>
  </group>
  <setting id="filestore_quota" type="hidden"
           default-value="1024">
   <name>
     Open-Xchange context wide filestore quota (in MB)
   </name>
  </setting>
 </settings>
<!-- Service used technologies (step 6) -->
 <requirements xmlns:php="http://apstandard.com/ns/1/php">
  <php:version min="5.0" />
  <php:extension>soap</php:extension>
 </requirements>
<!-- Service configuration script declaration (step 10) -->
 <provision>
  <configuration-script name="configure.php">
```

```
   <script-language>php</script-language>
   <status-control/>
  </configuration-script>
 </provision>
 <!-- Application services (step 4) -->
 <service id="account" class="account">
<!-- Service presentation properties (step 7) -->
  <presentation>
   <entry-points>
    <entry dst="http://{ox_site}/mail/elogin.php" method="POST">
     <label>Open-Xchange account</label>
     <variable name="ox_site" value-of-setting="ox_site"/>
     <variable name="username" value-of-setting="user_login"/>
     <variable name="password" value-of-setting="user_password"/>
    </entry>
   </entry-points>
  </presentation>
<!-- Service settings (step 7) -->
  <settings>
   <group class="authn">
    <setting id="user_login" class="login"
             type="string" installation-only="true"
             min-length="1">
     <name>Login</name>
    </setting>
    <setting id="user_password" class="password"
             type="password" min-length="1">
     <name>Password</name>
    </setting>
   </group>
   <group class="vcard">
    <group class="email">
     <setting id="user_email" class="value" type="email">
      <name>Primary email address</name>
     </setting>
    </group>
    <group class="fn n">
     <setting id="user_given_name" class="given-name" type="string"
              min-length="1">
      <name>Given name</name>
     </setting>
     <setting id="user_surname" class="family-name" type="string"
              min-length="1">
      <name>Surname</name>
     </setting>
    </group>
   </group>
   <setting id="admin_login" type="hidden"
            value-of-setting="admin_login"/>
   <setting id="admin_password" type="hidden"
            value-of-setting="admin_password"/>
  </settings>
<!-- Service used technologies (step 6) -->
  <requirements xmlns:mail="http://apstandard.com/ns/1/mail">
   <mail:mailbox>
    <mail:id>account</mail:id>
    <mail:access>
     <mail:imap/>
    </mail:access>
    <mail:outgoing>
     <mail:smtp/>
    </mail:outgoing>
   </mail:mailbox>
```

```
    </requirements>
<!-- Service configuration script declaration (step 10) -->
    <provision>
     <configuration-script name="configure-mbox.php">
      <script-language>php</script-language>
      <status-control/>
     </configuration-script>
    </provision>
   </service>
 </service>
</application>
```

Sample Environment Variables

The types of environment variables that are passed to the *Open-Xchange* configuration scripts by a Controller are as follows:

- Variables are defined by application settings (see the **5.3.2.2.2. Settings** section of the Specification).

  SETTINGS_ox_host

  SETTINGS_ox_site

  SETTINGS_ox_master_admin

  SETTINGS_ox_master_password

  SETTINGS_admin_login

  SETTINGS_admin_password

  SETTINGS_admin_email

  SETTINGS_admin_given_name

  SETTINGS_admin_surname

  SETTINGS_organization_name

  SETTINGS_filestore_quota

  OLDSETTINGS_admin_password

  SETTINGS_user_login

  SETTINGS_user_password

  SETTINGS_user_email

  SETTINGS_user_given_name

  SETTINGS_user_surname

- Aspects-defined (see the **7.4. Environment Variables** section of the Specification).
  - **PHP aspect**

    PHP_VERSION

  - **Mail aspect**

    MAIL_account_EMAIL

    MAIL_account_IMAP_HOST

    MAIL_account_IMAP_MAILBOX

    MAIL_account_IMAP_PORT

    MAIL_account_IMAP_PORT_SSL

    MAIL_account_PASSWORD

    MAIL_account_POP3_HOST

```
MAIL_account_POP3_PORT
MAIL_account_POP3_PORT_SSL
MAIL_account_SMTP_HOST
MAIL_account_SMTP_PORT
MAIL_account_SMTP_PORT_SSL
MAIL_account_USER
```

APPENDIX E

SpamExperts Incoming Email Security Firewall Sample Metadata File

This appendix includes metadata file of the SpamExperts Incoming Email Security Firewall application package. The following metadata contains comments with references to specific steps in Packaging SpamExperts Incoming Email Security Firewall Application (on page 41).

```
<!-- Application namespaces and APS version (step 9) -->
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://apstandard.com/ns/1" version="1.2">
<!-- Application common properties (step 1) -->
   <id>spamexperts-psf</id>
    <name>Professional Spam Filter</name>
    <version>2</version>
    <release>186</release>
    <homepage>http://www.spamexperts.com/</homepage>

    <vendor>
        <name>SpamExperts B.V.</name>
        <homepage>http://www.spamexperts.com/</homepage>
        <icon path="images/prospamfilter2.png"/>
    </vendor>

    <packager>
        <name>SpamExperts B.V.</name>
        <homepage>http://www.spamexperts.com/</homepage>
        <uri>uuid:7d652dbc-f2e9-463a-a658-b8a8369112ad</uri>
    </packager>


<!-- Presentation details and settings (step 6) -->
    <presentation>
        <summary>The addon automatically adds new domains to the Incoming Email
Security Firewall from SpamExperts, changes the MX records and offers the enduser a
one-click login option. Usage of this addon requires the SpamExperts Incoming Email
Security Firewall.</summary>
        <description>Server-side anti-spam filter for all users of your domain;
Proven, trusted, and secure; Best price/ quality anti-spam gateway on the market; Over
99.99% filtering accuracy; Easy to implement; Fully managed service; Friendly
multilevel support with 24/7/365 monitoring; No email ever gets lost</description>
        <icon path="images/prospamfilter2.png"/>
        <screenshot path="images/screenshot.png">
            <description>Main application view</description>
        </screenshot>

        <changelog>
            <version version="1.0" release="174">
                <entry>Package revision 1.0, release 174, built at 05-01-2011
09:08</entry>
            </version>
            <!-- Yes, we should add a "real" changelog here -->
        </changelog>
```

```
        <categories>
            <category>Collaboration/Email</category>
        </categories>
        <languages>
            <language>en</language>
        </languages>
    </presentation>

    <global-settings>
       <group class="authn">
             <name>SpamPanel Settings</name>
             <setting id="spampanel_url" class="access_point" type="string"
default-value="" regex="^https?://.*$">
                 <name>Spampanel URL</name>
                 <description>
                    This is an URL in format &amp;apos;http://www.spampanel.com/&amp;apos;
showing where spampanel is installed. Unless you use a custom CNAME, this is the primary
hostname of your first server.
                 </description>
                 <error-message>Please make sure the text you entered starts with
either http:// or https://.</error-message>
             </setting>
        </group>

        <group>
             <name>API Settings</name>
             <setting id="apihost" class="title" type="string" default-value=""
min-length="1">
                 <name>API hostname</name>
                 <description>
                    This is DNS name or IP address of SpamExperts product
installation, used for API access.
                 </description>
             </setting>
             <setting id="apiuser" class="title" type="string"
default-value="admin" min-length="1">
                 <name>API username</name>
                 <error-message>
                    Please make sure the text you entered starts with a letter and
continues with either numbers, letters, underscores or hyphens.
                 </error-message>
             </setting>
             <setting id="apipass" class="title" type="password" default-value=""
min-length="1">
                 <name>API password</name>
             </setting>
             <setting id="ssl_enabled" class="title" type="boolean"
default-value="true">
                 <name>Enable SSL</name>
                 <description>
                    If you want to communicate with the spamcluster over SSL, tick
this box
                 </description>
             </setting>
        </group>

       <group>
             <name>Virtual MX record</name>
             <setting id="mx1" class="title" type="string" default-value=""
min-length="1">
                 <name>Primary MX</name>
             </setting>
       </group>
```

```
    </global-settings>

    <upgrade match="/application/version = '1.0' and /application/release
&gt;='161'" />
<!-- Defining services (step 3) -->
    <service id="main" class="service">
        <license must-accept="true">
            <text>
                <name>General Terms &amp; Conditions and License Terms</name>

<url>http://www.spamexperts.com/fileadmin/documentation/EN/SpamExperts_General_Te
rms_and_Conditions_and_License_Terms.pdf</url>
            </text>
        </license>

        <presentation>
                    <name>Domain Specific Settings</name>
                    <summary>This part contains all the domain specific settings.
</summary>
                    <infolinks>
                           <link class="official"
href="http://www.spamexperts.com/">Official site</link>
                           <link class="support"
href="https://secure.spamexperts.com/whmcs/supporttickets.php">Support</link>
                    </infolinks>
                    <entry-points>
                           <entry dst="{spampanel_url}/index.php" method="POST">
                                  <label>Antispam Controlpanel</label>
                                  <variable name="spampanel_url"
value-of-setting="spampanel_url" />
                                  <!--<variable name="username"
value-of-setting="domain_username" />-->
                                  <variable name="password"
value-of-setting="domain_password" />
                                  <variable name="redirect">_BASE_</variable>
                           </entry>
                    </entry-points>
             </presentation>

      <settings>
             <group class="authn">
                    <setting id="domains" class="domain-name" visibility="hidden"
protected="true" type="list" element-type="string" track-old-value="true">
                    <name>List of domains</name>
                           <description>
                                        Domain names to provision.
                           </description>
                    </setting>
                    </setting>
                    <setting id="domain_password" class="password" type="password"
protected="true" track-old-value="true" min-length="1">
                           <name>Password</name>
                           <description>
                                  This password is used to login to the antispam
control panel.
                           </description>
                    </setting>
             </group>
             <group class="vcard">
                    <group class="email">
                           <setting id="email" class="email-on-application-domain"
type="email" optional="true">
                               <name>Your e-mail address</name>
```

```
                                    <description>This is the login part of your email
which is automatically completed once a new account is created.</description>
                                <error-message>Please provide a valid email
address</error-message>
                            </setting>
                        </group>
                </group>
        </settings>
<!-- Defining used technologies (step 4) -->
        <requirements xmlns:php="http://apstandard.com/ns/1/php"
xmlns:dns="http://apstandard.com/ns/1/dns">
                <php:version min="5.2" />
                <php:extension>openssl</php:extension>
                <php:extension>curl</php:extension>
<!-- Defining content delivery method (step 5) -->
                <!-- Primary Record (Required setting)-->
                <dns:record>
                        <dns:id>MX1</dns:id>
                        <dns:mx>
                                <dns:src>
                                        <dns:external
value-of-setting="domains"></dns:external>
                                </dns:src>
                                <dns:priority>10</dns:priority>
                                <dns:dst>
                                        <dns:external value-of-setting="mx1"/>
                                </dns:dst>
                                <dns:substitute/>
                        </dns:mx>
                </dns:record>

        </requirements>
<!-- Adding references to scripts (step 7) -->
        <provision>
                <configuration-script name="configure.php">
                        <script-language>php</script-language>
                        <structured-output/>
                </configuration-script>

                <!--
                <resource-script name="report-resources.php">
                        <script-language>php</script-language>
                </resource-script>

                <backup-script name="backup.php">
                        <script-language>php</script-language>
                </backup-script>
                -->
        </provision>
    </service>
</application>
```

Sample Environment Variables

The types of environment variables that are passed to the *Open-Xchange* configuration scripts by a Controller are as follows:

- Variables defined by application settings (see **5.3.2.2.2. Settings** section of the Specification):

  `SETTINGS_apihost`

  `SETTINGS_apiuser`

  `SETTINGS_apipass`

  `SETTINGS_ssl_enabled`

  `SETTINGS_email`

  `SETTINGS_domain_password`

  `SETTINGS_domains_`

  `OLDSETTINGS_domains_`

- Variables defined by the DNS zone:

  `DNS_MX1_SUBSTITUTE`

CHAPTER 6

Validating Application Package

This chapter provides instructions on validating application package. The package validation may comprise two steps:

- Checking package physical structure using APSLint utility.
- Checking whether package is properly managed by Controllers.

APSLint Utility

After you created an application package, you can determine whether the package conforms to the APS using a command-line utility called `APSlint`. The utility also enables you to check whether the application package can attain a certain APS certification level. For information on certification levels and requirements a package must satisfy to attain a specific level, refer to the *APS: Application Certification Criteria* document located on the official APS site.

`APSlint` utility validates the following:

- Application package file format;
- Application package physical structure;
- Application package metadata file.

As a result, the utility outputs the following data:

- Errors indicating why a package fails to conform to the APS;
- Errors indicating requirements that a package must satisfy to attain a certain certification level;
- Total number of errors. If 0 errors occur, the package is successfully validated.

The `APSlint` utility can be downloaded from the APS Standard web site (Windows and Linux/Unix versions are available).

Managing APSLint for Windows

To install the APSlint utility:

1. Download the utility using the following URL: http://www.apsstandard.org/r/doc/apslint.zip.
2. Install Microsoft .NET Framework version 2.0/3.0/3.5. In case you already have it, just skip this step.
3. Extract the utility to your preferred location. We recommend it to be a folder where your application package file is located.

To validate an application package:

1. Run `APSlint` from the directory where the application package is located using the following command:

```
> apslint.exe <application-package-name>.app.zip
```

Note: if you run `APSlint` from another directory, you must provide the absolute or relative path to the application package that you want to validate.

Managing APSLint for Linux/Unix

APSLint requires *mono 2.0* emulator and *libgdiplus* package to be installed on server before running APSLint.

To install the APSlint utility:

1. Download the utility using the following URL: http://www.apsstandard.org/r/doc/apslint.tar.gz.
2. Install Mono version 1.2.x (2.0 runtime) and libgdiplus. In case you already have them, just skip this step.
3. Extract the utility to your preferred location.

To validate an application package:

1. Run `APSlint` from the directory where the application package is located using the following command:
   - If the `binfmt` module is installed in your system:

```
# ./apslint.exe <application-package-name>.app.zip
```

   - If the `binfmt` module is not installed in your system:

```
# mono ./apslint.exe <application-package-name>.app.zip
```

Note: if you run `APSlint` from another directory, you must provide the absolute or relative path to the application package that you want to validate.

Controller Operations on Packages

This section explains how Controller performs management operations that require application package data. Examine your package, considering Controller operations execution. These operations are as follows:

- Creating application instance from application package;
- Updating application instances using application package;
- Removing application instance created from application package.

Creating Application Instance

Controller performs the following actions when it is instantiating an application package:

1 Prepares all the data and conditions necessary for instantiating:

- Presents an application-usage license text (if any is provided in the package) to a user and provides tools necessary for accepting the license if it is required. For details, refer to the **5.2.2 License Agreement** section in the Specification.
- Determines and satisfies the application requirements. For details, refer to the **5.2.6. Requirements, 5.3.2.2.3. Aspect-defined Environment Variables** sections in the Specification.
- Prompts the user to configure the application installation settings and processes the submitted settings. For details, refer to the **5.3.2.2. Environment Variables**, **5.2.4. Entry points**, **5.2.5. Service Settings** sections in the Specification.
- Retrieves the application's global configuration and prepares to apply it to the instance. For details, refer to **5.2.5. Service Settings, 5.3.2.2.2. Settings** sections in the Specification.

Note: The order of these preparatory sub-stages depends on the Controller implementation.

2 Deploys the application files to a site. For details, refer to the **5.3.1. URL mapping**, **5.3.2.2.3. Aspect-defined Environment Variables** sections in the Specification.

3 Runs the application configuration script passing to it all the available information about the application in the form of environment variables created on the previous stages. For details, refer to the **6.6. Configuration Script Language**, **5.3.2. Configuration script**, **6.4. Environment Variables** sections in the Specification.

Updating Application Instance

Controller can update an application instance in case a suitable update package is available. Updating application instance supposes that the application of older version is replaced with the one of newer version, and the user settings and files of the old application are picked up by the new one.

APS supports two types of application updates: *patch* and *upgrade*. Generally speaking, the difference is that patching supposes unattended instance update, while upgrading brings more serious changes to the instance and may require the user attendance. Application patch and upgrade are defined in detail by the Specification in section **5.1.13. Updates**. Depending on what kind of update is required - patch or upgrade - a Controller behavior slightly differs:

- When patching, the Controller skips reading new requirements, and preserves all resources allocated to the old version so that the new version could painlessly pick them all up, as defined by the Specification in section **5.2.6. Requirements**.
- When upgrading, the Controller analyzes new requirements and provides a mechanism of satisfying them together with migrating the application vitally-important data. (For example, when a new application version requires using a database of another version, the Controller provides mechanism of allocating the new database and migrating to it the old data.)

Removing Application Instance

When removing an application instance, a Controller invokes the application configuration script with `remove` argument. It removes the application files and deallocates resources used by it, as defined by the Specification in section **5.3.2.1. Configuration script actions**. The Controller passes to the configuration script all application settings declared in metadata, except marked as `installation-only`, with the `SETTINGS_`<*id*> environment variables, as defined by the Specification in section **5.3.2.2.2. Settings**.

APPENDIX F

APS Application Categories

1, Web

- Analytics
- Blog
- Catalog
- Gallery
- E-commerce
- eLearning
- Forum
- Wiki
- Search
- Site editing
- Tools
- Content management

2. Collaboration

- Email
- Calendaring
- Chat
- IP telephony
- Telecommunications
- Web conferencing
- Project management
- Portal

3. Back office

- Billing
- Payroll
- Customer Relationship Management
- Sales Force Automation
- Human Resource Management
- Business intelligence

- Business Process Management
- Product planning
- Asset Management
- Accounting and Financial
- Channel Management
- Compliance and Risk Management
- Dispatch Management
- Enterprise Resource Planning
- Expense Management
- Knowledge Management
- Manufacturing Solutions
- Portfolio Management
- Procurement
- Product Lifecycle Management
- Professional Services Automation
- Sales Compensation Management
- Supply Chain Management
- Talent Management
- Transportation Management
- Transportation and Logistics
- Vendor Management
- Work Order Management
- Workforce and Field Service Management

4. Front office

- Appointment Scheduling
- Help Desk
- Call Center
- Interactive Voice Response
- On-line Marketing
- Productivity
- Survey Solutions
- Infrastructure
- Security
- Backup

- Management
- Application Management

6. Customization
    - Localization
    - Themes

Index

S

T

U

V

W