



SWsoft

PEM 2.6 Update 2

Public API Reference

Revision 4.8.1 (March 13, 2007)



(c) 1999-2007

ISBN: N/A
SWsoft.
13755 Sunrise Valley Drive
Suite 325
Herndon
VA 20171 USA
Phone: +1 (703) 815 5670
Fax: +1 (703) 815 5675

Copyright © 1999-2007 by SWsoft. All rights reserved
Distribution of this work or derivative of this work in any form is prohibited unless prior written permission is obtained from the copyright holder.
PEM is a registered trademark of SWsoft, Inc.
Linux is a registered trademark of Linus Torvalds.
RedHat is a registered trademark of Red Hat Software, Inc.
UNIX is a registered trademark of The Open Group.
Intel, Pentium, Xeon and Celeron are registered trademarks of Intel Corporation.
SSH and Secure Shell are trademarks of SSH Communications Security, Inc.
Request Tracker is a trademark of Best Practical Solutions, LLC.

Contents

Preface	6
Typographical Conventions.....	6
Feedback.....	7
Introduction	8
Purpose.....	9
Scope.....	9
References.....	9
Definitions, Acronyms, and Abbreviations.....	10
Overview.....	10
Overall Description	11
Management of Accounts and Account's Staff Members.....	12
Subscription Management.....	13
Domains Management.....	14
Webspace Management.....	15
Mail management.....	16
Database management.....	16
Resource Accounting.....	17
Applications Management.....	17
Security Management.....	18
Native Package Management.....	18
Branding Management.....	18
Plesk Management.....	18
Transactional Extensions.....	18
Public API Reference	19
Management of Accounts and Account's Staff Members.....	24
pem.addAccount.....	24
pem.addAccountMember.....	27
pem.disableAccount.....	30
pem.doLogin.....	31
pem.getAccountInfo.....	31
pem.getAccountMemberByLogin.....	32
pem.getAccountMemberInfo.....	32
pem.getAccountSubscriptions.....	33
pem.enableAccount.....	33
pem.removeAccount.....	34
pem.removeAccountMember.....	34
pem.setAccountAuthData.....	35
pem.setAccountInfo.....	36
pem.setAccountStatus.....	38
pem.setMemberInfo.....	38
pem.setMemberPassword.....	40
pem.updateAccountAndAccountMember.....	40
Subscriptions Management.....	42
Provisioning Models.....	42

pem.activateSubscription	42
pem.addSubscription	43
pem.getServiceTemplate	44
pem.getSubscription	45
pem.disableSubscription	46
pem.enableSubscription	47
pem.migrateSubscription	47
pem.orderSubscription (deprecated)	48
pem.provisionSubscription (deprecated)	49
pem.removeSubscription	50
pem.setSubscriptionName	51
pem.upgradeSubscription (deprecated)	51
Domains Management	52
pem.addDomain	52
pem.addSubdomain	54
pem.disableDomain	55
pem.enableDomain	55
pem.getDomainInfo	56
pem.getDomainList	57
pem.getDomainSubscription	58
pem.getWebspacesList	58
pem.getNameServers	59
pem.importCertificate	59
pem.removeDomain	60
pem.removeSubdomain	60
pem.setDomainHostingType	61
pem.setDomainRegistrarStatus	62
pem.setSubdomainsHostingType	63
WebSpace Management	64
pem.getFTPUser	64
pem.setFTPPassword	64
pem.web.getFileManagerInfo	65
pem.web.getFrontPageInfo	66
pem.web.getFTPConf	66
pem.web.getHttpErrorDocs	67
pem.web.getMimeTypes	67
pem.web.getSiteProps	68
pem.web.getSSLInfo	69
pem.addWebSpaceBackup	69
pem.setFrontPageParameters	70
Mail Management	71
pem.cqmail.addMailbox	71
pem.cqmail.addMailForwarding	73
pem.cqmail.getAutoresponderInfo	73
pem.cqmail.getAutoresponderList	74
pem.cqmail.getForwardingsList	74
pem.cqmail.getMailboxInfo	75
pem.cqmail.getMailHostingInfo	75
pem.cqmail.getMailLists	76
pem.cqmail.getMailListInfo	77
pem.cqmail.getMailListOwners, pem.cqmail.getMailListModerators,	
pem.cqmail.getMailListMembers	77
pem.cqmail.getMailnameList	78
Database Management	78
pem.createDatabase	78
pem.createDatabaseUser	79
pem.getDatabaseAccessHostList	80
pem.getDatabaseInfo	81
pem.getDatabaseList	81
pem.getDatabaseUserList	82

Resource Accounting	83
pem.getResourceUsage	83
pem.getResourceUsageForPeriod	84
pem.resetResourceUsage	87
pem.setResourceTypeLimit	87
pem.setResourceTypeLimits	88
Applications Management	89
pem.installWebApplication	89
pem.removeWebSpaceBackup	90
pem.restoreWebSpaceFromBackup	90
pem.uninstallWebApplication	91
Security Management	91
pem.checkPassword	91
Native Package Management	92
pem.packaging.native_repository.createRepository	92
pem.packaging.native_repository.getRepository	93
pem.packaging.native_repository.reindex	93
pem.packaging.native_repository.removeRepository	93
Branding Management	94
pem.brandDomain	94
pem.unbrandDomain	95
Plesk Management	95
pem.installPleskLicense	95
pem.revokePleskLicense	95
Transactional Extensions	96
pem.batchRequest	96
pem.getRequestStatus	96
XML-RPC Samples	98
Common Response Codes	99
Success Response	99
Failure Response	100
pem.activateSubscription Sample	100
pem.addAccount Sample	103
pem.addAccountMember Sample	106
pem.addDomain Sample	109
pem.batchRequest Sample	109
Diagrams	119
Obsolete Ordering a Subscription Model	119
Index	120

CHAPTER 1

Preface

In This Chapter

Typographical Conventions.....	6
Feedback	7

Typographical Conventions

Formatting convention	Type of Information	Example
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.	Go to the QoS tab.
<i>Italics</i>	Titles of chapters, sections, and subsections. 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.	Read the Basic Administration chapter. The system supports the so called <i>wildcard character</i> search.
Monospace	The names of commands, files, and directories.	The license file is located in the httpdocs/common/licenses directory.
Preformatted	On-screen computer output in your command-line sessions; source code in XML, C++, or other programming languages.	<pre># ls -al /files total 14470</pre>
Preformatted Bold	What you type, contrasted with on-screen computer output.	<pre># cd /root/rpms/php</pre>
[...]	Square brackets indicate the the parameter used inside them is optional.	[host_id]

Feedback

If you have found a mistake in this guide, or if you have suggestions or ideas on how to improve this guide, please send your feedback to userdocs@swsoft.com. Please include in your report the guide's title, chapter and section titles, and the fragment of text in which you have found an error.

CHAPTER 2

Introduction

In This Chapter

Purpose.....	9
Scope.....	9
References.....	9
Definitions, Acronyms, and Abbreviations.....	10
Overview	10

Purpose

The purpose of this Reference document is to describe the *Public API* of PEM. This API provides functionality enough to integrate PEM with external billing system.

This document is intended for any person participating in integration of PEM with external billing system: analysts gathering and elaborating the requirements, designers and software architects designing the system, developers implementing the solution, testers and test designers performing quality assurance, technical writers composing user documentation, etc.

The reader of this document is expected to be familiar with basic PEM® concepts like: objects hierarchy, including accounts; account's staff members; domains; hosts; customer services; etc. The reader should also be familiar with resources accounting system used in PEM and understand such terms as: resource type, service template, subscription, resource.

Scope

This document describes all methods constituting the Public API of PEM in terms of their functionality. It also provides comprehensive explanation of methods' input and output parameters and gives examples of their usage.

References

The following documents are referenced elsewhere in this Reference:

- 1 XML-RPC Specification (<http://www.xmlrpc.com/spec>).
- 2 Extensible Markup Language (XML) (<http://www.w3.org/xml/>).
- 3 HTTP "Hypertext Transfer Protocol" (<http://www.w3.org/protocols/>).

Definitions, Acronyms, and Abbreviations

The following definitions, acronyms, and abbreviations are used throughout this document:

HTTP – Hypertext Transfer Protocol.

RPC – Remote Procedure Calls.

XML – eXtensible Markup Language.

XML-RPC - Remote procedure calling using HTTP as the transport and XML as the encoding.

Optional parameter (parameter name enclosed in square brackets) - when you come over the method parameter enclosed in square brackets [...], it means that the parameter is optional and that you can remove the tag that corresponds to the parameter request. Eg. [host_id]

Deprecated method - if a method is marked as deprecated, it means that there exist other newer methods that can deliver the same result as the deprecated method. Still, the deprecated method works fine as well until you see it in the document.

Overview

The Reference document first provides general description of PEM® Public API, then describes how it is structured according to functional groups, and finally lists all methods constituting the API in alphabetical order. The examples for methods are given in the appendix A (on page 98).

CHAPTER 3

Overall Description

The purpose of PEM Public API is to provide external billing systems with a way to perform accounting and billing operations together with PEM.

The API represents a set of methods implemented over XML-RPC. All methods conform to XML-RPC specification so that the external system can use any third-party software implementing this protocol.

All methods constituting the API are transactional on per-call basis. In other words, if an XML-RPC call succeeds, PEM can guarantee that all operations implied by the called method have been performed. And vice versa, if a call fails, PEM guarantees that no changes implied by this method have been made to the system. However, a sequence of calls is not transactional as a whole and the external system shall handle failed calls in the sequence itself.

The whole Public API of PEM can be subdivided into several groups according to the primary functionality they support. The following subsections briefly describe these functional groups.

In This Chapter

Management of Accounts and Account's Staff Members	12
Subscription Management	13
Domains Management	14
Webspace Management	15
Mail management.....	16
Database management.....	16
Resource Accounting	17
Applications Management.....	17
Security Management	18
Native Package Management.....	18
Branding Management.....	18
Plesk Management	18
Transactional Extensions	18

Management of Accounts and Account's Staff Members

This functional group contains the following methods:

pem.addAccount (on page 24)

pem.addAccountMember (on page 27)

pem.disableAccount (on page 30)

pem.doLogin (on page 31)

pem.enableAccount (on page 33)

pem.getAccountInfo (on page 31)

pem.getAccountMemberByLogin (on page 32)

pem.getAccountMemberInfo (on page 32)

pem.getAccountSubscriptions (on page 33)

pem.removeAccount (on page 34)

pem.removeAccountMember (on page 34)

pem.setAccountAuthData

pem.setAccountInfo (on page 36)

pem.setAccountStatus (on page 38)

pem.setMemberInfo (on page 38)

pem.setMemberPassword (on page 40)

pem.updateAccountAndAccountMember (on page 40)

Subscription Management

This functional group contains the following methods:

pem.activateSubscription

pem.addSubscription

pem.disableSubscription (on page 46)

pem.enableSubscription (on page 47)

pem.getServiceTemplate (on page 44)

pem.getSubscription (on page 45)

pem.migrateSubscription (on page 47)

pem.orderSubscription (deprecated) (on page 48)

pem.provisionSubscription (on page 49)

pem.removeSubscription (on page 50)

pem.setSubscriptionName (on page 51)

pem.upgradeSubscription (deprecated) (on page 51)

Domains Management

This functional group contains the following methods:

pem.addDomain

pem.addSubdomain (on page 54)

pem.getDomainSubscription (on page 58)

pem.disableDomain (on page 55)

pem.enableDomain (on page 55)

pem.importCertificate (on page 59)

pem.getDomainInfo (on page 56)

pem.getDomainList (on page 57)

pem.getWebspacesList (on page 58)

pem.getNameServers (on page 59)

pem.removeDomain (on page 60)

pem.removeSubdomain (on page 60)

pem.setDomainHostingType (on page 61)

pem.setDomainRegistrarStatus (on page 62)

pem.setSubdomainsHostingType (on page 63)

Webspace Management

This functional group contains the following methods:

`pem.getFTPUser` (on page 64)

`pem.setFTPPassword` (on page 64)

`pem.web.getFileManagerInfo` (on page 65)

`pem.web.getFrontPageInfo` (on page 66)

`pem.web.getFTPConf` (on page 66)

`pem.web.getHttpErrorDocs` (on page 67)

`pem.web.getMimeTypes` (on page 67)

`pem.web.getSiteProps` (on page 68)

`pem.web.getSSLInfo` (on page 69)

`pem.addWebSpaceBackup` (on page 69)

`pem.setFrontPageParameters` (on page 70)

Mail management

pem.cqmail.addMailbox (on page 71)

pem.cqmail.getAutoresponderInfo (on page 73)

pem.cqmail.getAutoresponderList (on page 74)

pem.cqmail.getForwardingsList (on page 74)

pem.cqmail.getMailboxInfo (on page 75)

pem.cqmail.getMailHostingInfo (on page 75)

pem.cqmail.getMailLists (on page 76)

pem.cqmail.getMailListInfo (on page 77)

pem.cqmail.getMailListOwners, pem.cqmail.getMailListModerators,
pem.cqmail.getMailListMembers (on page 77)

pem.cqmail.getMailnameList (on page 78)

Database management

pem.createDatabase (on page 78)

pem.createDatabaseUser (on page 79)

pem.getDatabaseAccessHostList (on page 80)

pem.getDatabaseInfo (on page 81)

pem.getDatabaseList (on page 81)

pem.getDatabaseUserList (on page 82)

Resource Accounting

This functional group contains the following methods:

`pem.getResourceUsage` (on page 83)

`pem.getResourceUsageForPeriod` (on page 84)

`pem.resetResourceUsage` (on page 87)

`pem.setResourceTypeLimit`

`pem.setResourceTypeLimits` (on page 88)

Applications Management

This functional group contains the following methods:

`pem.installWebApplication` (on page 89)

`pem.removeWebSpaceBackup` (on page 90)

`pem.restoreWebSpaceFromBackup` (on page 90)

`pem.uninstallWebApplication` (on page 91)

Security Management

pem.checkPassword (on page 91)

Native Package Management

pem.packaging.native_repository.createRepository (on page 92)

pem.packaging.native_repository.getRepository (on page 93)

pem.packaging.native_repository.reindex (on page 93)

pem.packaging.native_repository.removeRepository (on page 93)

Branding Management

pem.brandDomain (on page 94)

pem.unbrandDomain (on page 95)

Plesk Management

pem.installPleskLicense (on page 95)

pem.revokePleskLicense (on page 95)

Transactional Extensions

pem.batchRequest

pem.getRequestStatus (on page 96)

CHAPTER 4

Public API Reference

This section lists all methods constituting the PEM Public API along with comprehensive explanation of their input and output parameters. The list is sorted in alphabetical order for better navigation.

Descriptions of all methods follow the same way: first, a short explanation is given of what the method does, and then lists of its input and output parameters are provided in tabular form. If the method participates in complex scenario involving other methods, a link to the corresponding diagram, explaining this scenario, is given.

Types of all arguments are given in terms of XML-RPC specification.

All methods described in this document are supported by PEM version 2.3 or later, if not otherwise stated explicitly. New methods and new optional parameters for existing methods introduced in subsequent releases of PEM are marked with the version of PEM where they were introduced for the first time.

For the sake of convenience, the following table groups all methods from PEM Public API by releases where they were introduced for the first time.

Method Name	Effective since version
pem.activateSubscription	2.3.01
pem.addAccount (on page 24)	2.3
pem.addAccountMember (on page 27)	2.3
pem.addDomain	2.3
pem.addSubdomain (on page 54)	2.3
pem.addSubscription	2.3
pem.addWebSpaceBackup (on page 69)	2.7
pem.batchRequest	2.3

pem.brandDomain (on page 94)	2.6
pem.checkPassword (on page 91)	2.4
pem.cqmail.addMailbox (on page 71)	2.4
pem.cqmail.getAutoresponderInfo (on page 73)	2.7
pem.cqmail.getAutoresponderList (on page 74)	2.7
pem.cqmail.getForwardingsList (on page 74)	2.7
pem.cqmail.getMailboxInfo (on page 75)	2.7
pem.cqmail.getMailHostingInfo (on page 75)	2.7
pem.cqmail.getMailLists (on page 76)	2.7
pem.cqmail.getMailListInfo (on page 77)	2.7
pem.cqmail.getMailListOwners, pem.cqmail.getMailListModerators, pem.cqmail.getMailListMembers (on page 77)	2.7
pem.cqmail.getMailnameList (on page 78)	2.7
pem.createDatabase (on page 78)	2.3
pem.createDatabaseUser (on page 79)	2.3
pem.disableAccount (on page 30)	2.3
pem.disableDomain (on page 55)	2.3
pem.disableSubscription (on page 46)	2.3
pem.doLogin (on page 31)	2.7
pem.enableAccount (on page 33)	2.3
pem.enableDomain (on page 55)	2.3
pem.enableSubscription (on page 47)	2.3
pem.getAccountInfo (on page 31)	2.4
pem.getAccountMemberByLogin (on page 32)	2.7
pem.getAccountMemberInfo (on page 32)	2.7
pem.getAccountSubscriptions (on page 33)	2.4
pem.getDatabaseAccessHostList (on page 80)	2.7

pem.getDatabaseInfo (on page 81)	2.7
pem.getDatabaseList (on page 81)	2.7
pem.getDatabaseUserList (on page 82)	2.7
pem.getDomainInfo (on page 56)	2.7
pem.getDomainSubscription (on page 58)	2.4
pem.getDomainList (on page 57)	2.5
pem.getFTPUser (on page 64)	2.4
pem.getNameServers (on page 59)	2.3
pem.getRequestStatus (on page 96)	2.6.2
pem.getResourceUsage (on page 83)	2.3
pem.getResourceUsageForPeriod (on page 84)	2.3
pem.getServiceTemplate (on page 44)	2.6.1
pem.getSubscription (on page 45)	2.6.1
pem.getWebspacesList (on page 58)	2.4
pem.importCertificate (on page 59)	2.3
pem.installPleskLicense (on page 95)	2.6
pem.installWebApplication (on page 89)	2.7
pem.migrateSubscription (on page 47)	2.5
pem.orderSubscription (deprecated) (on page 48)	2.3
pem.packaging.native_repository.createRepository (on page 92)	2.6
pem.packaging.native_repository.getRepository (on page 93)	2.6
pem.packaging.native_repository.reindex (on page 93)	2.6
pem.packaging.native_repository.removeRepository (on page 93)	2.6
pem.provisionSubscription (deprecated) (on page 49)	2.3

pem.removeAccount (on page 34)	2.3.01
pem.removeAccountMember (on page 34)	2.3.01
pem.removeDomain (on page 60)	2.3
pem.removeSubdomain (on page 60)	2.7
pem.removeSubscription (on page 50)	2.3
pem.removeWebSpaceBackup (on page 90)	2.7
pem.resetResourceUsage (on page 87)	2.3
pem.restoreWebSpaceFromBackup (on page 90)	2.7
pem.revokePleskLicense (on page 95)	2.6
pem.setAccountAuthData	2.3
pem.setAccountInfo (on page 36)	2.3
pem.setAccountStatus (on page 38)	2.3
pem.setDomainHostingType (on page 61)	2.7
pem.setDomainRegistrarStatus (on page 62)	2.3
pem.setFrontPageParameters (on page 70)	2.7
pem.setFTPPassword (on page 64)	2.4
pem.setMemberInfo (on page 38)	2.6.2
pem.setMemberPassword (on page 40)	2.6.2
pem.setResourceTypeLimit	2.3
pem.setResourceTypeLimits (on page 88)	2.6.1
pem.setSubdomainsHostingType (on page 63)	2.7
pem.setSubscriptionName (on page 51)	2.6.1
pem.unbrandDomain (on page 95)	2.6
pem.uninstallWebApplication (on page 91)	2.7
pem.updateAccountAndAccountMember (on page 40)	2.3
pem.upgradeSubscription (deprecated) (on page 51)	2.3

pem.web.getFrontPageInfo (on page 66)	2.7
pem.web.getFTPConf (on page 66)	2.7
pem.web.getHttpErrorDocs (on page 67)	2.7
pem.web.getMimeTypes (on page 67)	2.7
pem.web.getSiteProps (on page 68)	2.7
pem.web.getSSLInfo (on page 69)	2.7

In This Chapter

Management of Accounts and Account's Staff Members	24
Subscriptions Management.....	42
Domains Management	52
Webspace Management	64
Mail Management.....	71
Database Management.....	78
Resource Accounting.....	83
Applications Management.....	89
Security Management	91
Native Package Management.....	92
Branding Management.....	94
Plesk Management	95
Transactional Extensions.....	96

Management of Accounts and Account's Staff Members

PEM 2.4 introduces an ability to work in conjunction with the external system that stores accounts' information in its own database. If you configure PEM to work with external account information storage, the user should apply alternative public API methods' interface where needed. Note that some public API methods become unavailable under this scheme.

System ID parameter, that is used to refer to external system, is generated during registering of this external system in PEM.

pem.addAccount

This method is supported by PEM starting from version 2.3.

This method creates an account for Reseller or Customer in PEM. Note that this method does not create an account's member; use pem.addAccountMember (on page 27) method for this purpose.

Important: You should always call pem.addAccountMember (on page 27) just after pem.addAccount (on page 24). Otherwise, the deployment will be considered incorrect and, in general, subsequent call of pem.activateSubscription will lead to the unpredictable result.

The method has the following input parameters:

Name	Type	Short Description
[account_id]	int	id of newly created account, generated by external system.
[account_type]	string	type of account to create, (C)ustomer or (R)eseller. (C)ustomer is the default value.
[subscription_id]	int	The ID of subscription that provides resource "Client accounts" or "Reseller accounts" depending on account type.
[parent_account_id]	int	account id of a parent account. PEM provider (OID_ADMIN) is the default value.
[branded_domain_name]	string	The domain, which the brand is created on. If such domain is specified, the created account will be bound to this domain. If you don't specify this parameter, the customer will be bound to the first created brand even if there is only one brand.
<i>Send the following info, if PEM uses external account info storage:</i>		
external_info	struct	

▪ <code>system_id</code>	string	reference ID to external system that stores account info
▪ <code>external_account_id</code>	string	Some account identifier that will be passed to the external system for identification of this account. It should have the following format: Brand CustomerNr.

Send the following info, if PEM stores account info in its own database:

<code>person</code>	struct	general information about person/company
<code>address</code>	struct	address information
<code>phone</code>	struct	contact phone number
<code>[fax]</code>	struct	contact fax number
<code>email</code>	string	contact e-mail address

The method has the following output parameters (available since PEM 2.5 hotfix 12):

Name	Type	Short Description
<code>account_id</code>	int	id of the created account

For XML-RPC sample of using this method see `pem.addAccount` Sample section (on page 103) in XML-RPC Samples appendix (on page 98).

See also:

`pem.addAccountMember` method (on page 27).

Parameters Details

The request must contain either the set of account information parameters such as person, address, phone, fax, email or the `external_info` that uniquely identifies external system where PEM should get appropriate data.

The `external_info` field appears in PEM starting from version 2.4.

`account_id` - The ID of the account to create. This ID is generated by external billing system. Optional parameter. If this parameter is not specified, PEM will generate it according its internal sequence of IDs.

`account_type` - Indicates the type of the account to create. It can take the following values:

“C”. Indicates that the account is created for Customer. Default value.

“R”. Indicates that the account is created for Reseller.

`subscription_id` - The ID of subscription that provide resource "Client accounts" or "Reseller accounts" depending on account type. If `subscription_id` parameter is specified, then the OpenAPI searches that concrete subscription for appropriate resource. Method would fail if no resource found. If `subscription_id` is not specified, then the method will look at all owner's subscriptions and search least loaded resource of the needed type.

`parent_account_id` - ID of the account that owns (has administrative rights with respect to) the account being created.

“*OID_ADMIN*” stands for PEM provider. Default value.

`person` - Contains general information about person/company represented by the account being created. It is a struct with the following members:

- [`title`] of “string” type. This is salutation like “Mr.” or “Mrs.”, etc.
- [`first_name`] of “string” type. The first name of the person represented by the account.
- [`middle_name`] of “string” type. The middle name of the person represented by the account.
- [`last_name`] of “string” type. The last name of the person represented by the account.
- [`company_name`] of “string” type. The name of the company represented by the account. If the company is not specified, first name, middle name and last name will be used instead.

`address` - Contains address information for the person/company represented by the account being created. It is a struct with the following members:

- [`street_name`] of “string” type. Name of the street.
- [`house_num`] of “string” type. The number of the house.
- [`address2`] of “string” type. Second address.
- [`zipcode`] of “string” type. Postal code.

- “*city*” of “string” type. Name of the city.
- “*country*” of “string” type. This is the code of the country represented by two lowercase letters, like “us”, “uk”, etc.
- “*state*” of “string” type. Name of the state/province.

Note: Because of the fields inconsistency between PEM and PEM.Billing, the following behaviour is foreseen:

If the *house_num* is indicated, then address will go to the *street_name* field in PEM.Billing, *address2* => *house_num*, *address3* => *address2*, otherwise *address* => *street_name*, *address2* => *address2*.

phone - Contains fax information for the person/company represented by the account being created. It is a struct with the following members:

- “*country_code*” of “string” type. Country code part of the fax number.
- “*area_code*” of “string” type. Area code part of the fax number.
- “*phone_num*” of “string” type. The fax number itself.
- “*ext_num*” of “string” type. Extension to the fax number, if present.

[fax] - Contains fax information for the person/company represented by the account being created. It is a struct with the following members:

- “*country_code*” of “string” type. Country code part of the fax number.
- “*area_code*” of “string” type. Area code part of the fax number.
- “*phone_num*” of “string” type. The fax number itself.
- “*ext_num*” of “string” type. Extension to the fax number, if present.

email - The e-mail address of the person/company represented by the account being created.

pem.addAccountMember

This method is supported by PEM starting from version 2.3.

This method creates a member for Reseller’s or Customer’s account in PEM.

The method has the following input parameters:

Name	Type	Short Description
account_id	int	id of account new member will belong to
[subscription_id]	int	The ID of the subscription that provide "users" resource.
[user_id]	int	id of the member to create, generated by external system

Send the following data, when PEM uses external account auth info storage:

login	string	Login for newly created member
-------	--------	--------------------------------

Send the following data set, when PEM stores account auth info by it-self:

auth	struct	auth info (login/password) for new member
person	struct	general personal information
address	struct	postal address
phone	struct	contact phone number
[fax]	struct	contact fax number
email	string	contact email address

The method has the following output parameters (available since PEM 2.5 hotfix 12):

Name	Type	Short Description
user_id	int	id of the created user

For XML-RPC sample of using this method see pem.addAccountMember Sample (on page 106) section in XML-RPC Samples appendix (on page 98).

See also:

pem.addAccount (on page 24) method.

Parameters Details

Request must contain either the set of account information parameters such as `auth`, `person`, `address`, `phone`, `fax`, `email` or the `login` parameter for an external system account.

The login parameter appears in PEM starting from version 2.4.

`account_id` - The ID of the account, which the member is created for.

`subscription_id` - The ID of the subscription that provides "users" resource. If `subscription_id` parameter is specified, then the OpenAPI searches "users" resource in that concrete subscription. If no resource found, there are two options to appear:

- if an account has no members yet, then the member creation will succeed;
- if account already have any members, then the method will fail.

If `subscription_id` is not specified, then the method will look at all account's subscriptions and search least loaded "users" resource.

`user_id` - The ID of the member to create. This ID is generated by external billing system.

`auth` - Contains information to be used for authenticating the member being created. It is a struct with the following members:

- `"login"` of "string" type.
- `"password"` of "string" type.
- `["check_password"]` of "boolean" type. Optional parameter. It checks the quality of the entered password according to the quality level defined in PEM.

`person` - Contains general information about person represented by the member being created. It is a struct with the following members:

- `"title"` of "string" type. This is salutation like "Mr." or "Mrs.", etc.
- `"first_name"` of "string" type. The first name of the person represented by the member.
- `"middle_name"` of "string" type. The middle name of the person represented by the member.
- `"last_name"` of "string" type. The last name of the person represented by the member.

`address` - Contains address information for the person represented by the member being created. It is a struct with the following members:

- `"street_name"` of "string" type. Name of the street.
- `"house_num"` of "string" type. The number of the house.

- “*address2*” of “string” type. Second address.
- “*zipcode*” of “string” type. Postal code.
- “*city*” of “string” type. Name of the city.
- “*country*” of “string” type. This is the code of the country represented by two lowercase letters, like “us”, “uk”, etc.
- “*state*” of “string” type. Name of the state/province.

Note: Because of the fields inconsistency between PEM and PEM.Billing, the following behaviour is foreseen:

If the *house_num* is indicated, then address will go to the *street_name* field in PEM.Billing, *address2* => *house_num*, *address3* => *address2*, otherwise *address* => *street_name*, *address2* => *address2*.

phone - Contains phone information for the person represented by the member being created. It is a struct with the following members:

- “*country_code*” of “string” type. Country code part of the phone number.
- “*area_code*” of “string” type. Area code part of the phone number.
- “*phone_num*” of “string” type. The phone number itself.
- “*ext_num*” of “string” type. Extension to the phone number, if present.

fax - Contains fax information for the person represented by the member being created. It is a struct with the following members:

- “*country_code*” of “string” type. Country code part of the fax number.
- “*area_code*” of “string” type. Area code part of the fax number.
- “*phone_num*” of “string” type. The fax number itself.

“*ext_num*” of “string” type. Extension to the fax number, if present. email - The e-mail address of the person represented by the member being created.

pem.disableAccount

This method is supported by PEM starting from version 2.3.

This method disables all services provisioned under specified account. The corresponding Reseller or Customer will be still able to login to PEM, but will not be able to use and manage services provisioned under this account.

The method has the following input parameters:

Name	Type	Short Description
account_id	int	id of account to disable

The method has no output parameters.

See also:

`pem.enableAccount` method (on page 33).

Parameters Details

`account_id` - The ID of the account to disable.

`pem.doLogin`

This method is supported by PEM starting from version 2.7.

This method passes login information to PEM, verifies a pair of login/password, and returns the corresponding customer's `account_id` on success.

The method has the following input parameters:

Name	Type	Short Description
<code>login</code>	string	PEM login name
<code>password</code>	string	Password in a plain text

Method returns a structure with following content:

Name	Type	Short Description
<code>account_id</code>	int	Customer's account ID
<code>default_sub_id</code>	int	An ID of a customer's default subscription

`pem.getAccountInfo`

This method is supported by PEM starting from version 2.4.

This method returns information about account: contacts, parent accounts, etc.

The method has the following input parameters:

Name	Type	Short Description
<code>account_id</code>	string	PEM account ID

The output parameters are the following:

Name	Type	Short Description
<code>account_type</code>	string	Account type; 'C' for end customers, 'R' for resellers
<code>parent_account_id</code>	int	Parent account ID

person	struct	General information about person/company
address	struct	Customer/reseller address
phone	struct	Customer/reseller phone number
fax	struct	Customer/reseller fax number
email	sting	Customer/reseller E-mail address

Look into pem.addAccount method (on page 24) description for complete specification of these structures.

pem.getAccountMemberByLogin

This method is supported by PEM starting from version 2.7.

This method returns information about account member for specific login if there is any.

The method has the following input parameters:

Name	Type	Short Description
login	string	Login to find in PEM system

This method has the following output parameters:

Name	Type	Short Description
user_id	int	Account member ID
account_id	int	ID of an account that owns the member

pem.getAccountMemberInfo

This method is supported by PEM starting from version 2.7.

The method returns information about account member: contacts, parent account, etc.

The method has the following input parameters:

Name	Type	Short Description
user_id	string	PEM account ID

This method has the following output parameters:

Name	Type	Short Description
auth	struct	Account member authentication information
person	struct	General information on account member
address	struct	Account member's address
phone	struct	Account member's phone number

fax	struct	Account member's fax number
email	string	Account member's E-mail address

Look into pem.addAccountMember method (on page 27) for complete specification of these structures.

pem.getAccountSubscriptions

This method is supported by PEM starting from version 2.4.

This method returns a list of subscriptions that customer has bought.

The method has the following input parameters:

Name	Type	Short Description
account_id	int	Customer's account ID

Method returns an array(int) that contain a list of customer's subscription IDs.

pem.enableAccount

This method is supported by PEM starting from version 2.3.

This method enables all services provisioned under specified account if they were disabled. The corresponding Reseller or Customer will be able to login to PEM and manage services provisioned under this account.

Note: if pem.disableAccount method (on page 30) was called for an account, then it is not guaranteed that, after calling pem.enableAccount method, all services will be in the same state as they were just before pem.disableAccount method (on page 30) was called; it depends on particular kind of services.

The method has the following input parameters:

Name	Type	Short Description
account_id	int	id of account to enable

The method has no output parameters.

See also:

[pem.disableAccount](#) method (on page 30).

Parameters Details

account_id - The ID of the account to enable.

pem.removeAccount

This method is supported by PEM starting from version 2.3.01

This method removes an account from PEM. Note that this method automatically removes all account's data.

The method has the following input parameters:

Name	Type	Short Description
account_id	int	id of account to remove

Note: After account deletion the account ID remains in the database and can not be reused anew.

The method has no output parameters.

Parameters Details

account_id - The ID of the account to remove.

pem.removeAccountMember

This method is supported by PEM starting from version 2.3.01.

This method removes an account member from PEM.

The method has the following input parameters:

Name	Type	Short Description
user_id	int	id of member to remove

The method has no output parameters.

Parameters Details

`user_id` - the ID of the member to remove.

`pem.setAccountAuthData`

This method is supported by PEM starting from version 2.3.

Note: Starting from PEM version 2.4, the calling of this method is considered incorrect, if PEM is configured to gather account and account auth info from external system.

This method is used to change account member password.

Name	Type	Short Description
<code>auth</code>	<code>struct</code>	Account authentication info

▪ [member_id]	int	ID of an account member
▪ [login]	string	account login
▪ password	string	password to set
▪ [check_password]	boolean	checks the quality of the issued password

member_id and login parameters are optional to each other. It means that one of them must be explicitly specified.

This method has no output parameters.

Parameters Details

member_id - Identifies the account member to set the new password for.

login - Login of an account member

password - password to be set.

check_password - checks the quality of the issued password

pem.setAccountInfo

This method is supported by PEM starting from version 2.3.

Note: Starting from PEM version 2.4, the calling of this method is considered incorrect, if PEM is configured to gather account and account auth info from external system.

This method updates information of Reseller's or Customer's Account.

This method has the following input parameters:

Name	Type	Short Description
account_id	int	ID of an account
person	struct	general information about person/company
address	struct	address information
phone	struct	contact phone number
fax	struct	contact fax number
email	string	contact E-mail address

This method has no output parameters.

Parameters Details

`account_id` - Identifies the account to change.

`person` - Contains general information about person represented by the member being updated. It is a struct with the following members:

- `"title"` of "string" type. This is salutation like "Mr." or "Mrs.", etc.
- `"first_name"` of "string" type. The first name of the person represented by the member.
- `"middle_name"` of "string" type. The middle name of the person represented by the member.
- `"last_name"` of "string" type. The last name of the person represented by the member.
- `"company_name"` of "string" type. The name of the company represented by the member.

`address` - Contains address information for the person represented by the member being updated. It is a struct with the following members:

- `"street_name"` of "string" type. Name of the street.
- `"house_num"` of "string" type. The number of the house.
- `"address2"` of "string" type. Second address.
- `"zipcode"` of "string" type. Postal code.
- `"city"` of "string" type. Name of the city.
- `"country"` of "string" type. This is the code of the country represented by two lowercase letters, like "us", "uk", etc.
- `"state"` of "string" type. Name of the state/province.

`phone` - Contains phone information for the person represented by the member being updated. It is a struct with the following members:

- `"country_code"` of "string" type. Country code part of the phone number.
- `"area_code"` of "string" type. Area code part of the phone number.
- `"phone_num"` of "string" type. The phone number itself.
- `"ext_num"` of "string" type. Extension to the phone number, if present.

`fax` - Contains fax information for the person represented by the member being updated. It is a struct with the following members:

- `"country_code"` of "string" type. Country code part of the fax number.
- `"area_code"` of "string" type. Area code part of the fax number.
- `"phone_num"` of "string" type. The fax number itself.

`"ext_num"` of "string" type. Extension to the fax number, if present. `email` - The e-mail address of the person represented by the member being updated.

pem.setAccountStatus

This method is supported by PEM starting from version 2.3.

This method is used by external billing system to set account's status in PEM.

The method has the following input parameters:

Name	Type	Short Description
account_id	int	id of account to modify
account_status	string	new account status name

The method has no output parameters.

Parameters Details

account_id - Identifies the account to set the status for.

account_status - The status to set.

The following account statuses are supported in Game hosting: undefined, uncertified, verified, expired, validated.

For all the other hosting types only "undefined" status can be used.

pem.setMemberInfo

This method is supported by PEM starting from version 2.6.2

This method sets the contact information for member identified by login. The semantics of structure and field names are fully compatible with pem.setAccountInfo method (on page 36).

This method has the following input parameters:

Name	Type	Short Description
account_id	int	ID of an account
person	struct	general information about person/company
address	struct	address information
phone	struct	contact phone number
fax	struct	contact fax number
email	string	contact E-mail address

This method has no output parameters.

Parameters Details

`account_id` - Identifies the account to change.

`person` - Contains general information about person represented by the member being updated. It is a struct with the following members:

- `"title"` of "string" type. This is salutation like "Mr." or "Mrs.", etc.
- `"first_name"` of "string" type. The first name of the person represented by the member.
- `"middle_name"` of "string" type. The middle name of the person represented by the member.
- `"last_name"` of "string" type. The last name of the person represented by the member.
- `"company_name"` of "string" type. The name of the company represented by the member.

`address` - Contains address information for the person represented by the member being updated. It is a struct with the following members:

- `"street_name"` of "string" type. Name of the street.
- `"house_num"` of "string" type. The number of the house.
- `"address2"` of "string" type. Second address.
- `"zipcode"` of "string" type. Postal code.
- `"city"` of "string" type. Name of the city.
- `"country"` of "string" type. This is the code of the country represented by two lowercase letters, like "us", "uk", etc.
- `"state"` of "string" type. Name of the state/province.

`phone` - Contains phone information for the person represented by the member being updated. It is a struct with the following members:

- `"country_code"` of "string" type. Country code part of the phone number.
- `"area_code"` of "string" type. Area code part of the phone number.
- `"phone_num"` of "string" type. The phone number itself.
- `"ext_num"` of "string" type. Extension to the phone number, if present.

`fax` - Contains fax information for the person represented by the member being updated. It is a struct with the following members:

- `"country_code"` of "string" type. Country code part of the fax number.
- `"area_code"` of "string" type. Area code part of the fax number.
- `"phone_num"` of "string" type. The fax number itself.

`"ext_num"` of "string" type. Extension to the fax number, if present. `email` - The e-mail address of the person represented by the member being updated.

pem.setMemberPassword

This method is supported by PEM starting from version 2.6.2

This method changes the password for member identified by login.

Name	Type	Short Description
login	string	Login name of the member
password	string	Password to be issued to member
[check_password]	boolean	Checks the quality of password.

This method has no output parameters.

pem.updateAccountAndAccountMember

This method is supported by PEM starting from version 2.3.

Note: Starting from PEM version 2.4, the calling of this method is considered incorrect, if PEM is configured to gather account and account auth info from external system.

This method updates information on a member of Reseller's or Customer's account.

The method has the following input parameters:

Name	Type	Short Description
auth	struct	Auth information of an account to be updated
person	struct	new personal information
address	struct	new address
phone	struct	new contact phone number
fax	struct	new contact fax number
email	string	new contact e-mail address

The method has no output parameters.

See also:

`pem.addAccountMember` method (on page 27).

Parameters Details

`auth` - Identifies the member to update information for.

It is a struct with the following members:

- `"login"` of "string" type.
- `"password"` of "string" type.
- `"[check_password]"` of boolean type. This is an optional parameter. It checks the quality of the issued password.

`person` - Contains general information about person represented by the member being updated. It is a struct with the following members:

- `"title"` of "string" type. This is salutation like "Mr." or "Mrs.", etc.
- `"first_name"` of "string" type. The first name of the person represented by the member.
- `"middle_name"` of "string" type. The middle name of the person represented by the member.
- `"last_name"` of "string" type. The last name of the person represented by the member.
- `"company_name"` of "string" type. The name of the company represented by the member.

`address` - Contains address information for the person represented by the member being updated. It is a struct with the following members:

- `"street_name"` of "string" type. Name of the street.
- `"house_num"` of "string" type. The number of the house.
- `"address2"` of "string" type. Second address.
- `"zipcode"` of "string" type. Postal code.
- `"city"` of "string" type. Name of the city.
- `"country"` of "string" type. This is the code of the country represented by two lowercase letters, like "us", "uk", etc.
- `"state"` of "string" type. Name of the state/province.

`phone` - Contains phone information for the person represented by the member being updated. It is a struct with the following members:

- `"country_code"` of "string" type. Country code part of the phone number.
- `"area_code"` of "string" type. Area code part of the phone number.
- `"phone_num"` of "string" type. The phone number itself.
- `"ext_num"` of "string" type. Extension to the phone number, if present.

fax - Contains fax information for the person represented by the member being updated. It is a struct with the following members:

- “*country_code*” of “string” type. Country code part of the fax number.
- “*area_code*” of “string” type. Area code part of the fax number.
- “*phone_num*” of “string” type. The fax number itself.

“*ext_num*” of “string” type. Extension to the fax number, if present. email - The e-mail address of the person represented by the member being updated.

Subscriptions Management

Provisioning Models

3 provisioning models could be used in PEM:

- 1 Single call (`pem.activateSubscription`) model that supports variable parameters set to provide any kinds of resources supported by PEM. Additionally you could use `pem.addSubscription` call to create subscription, but in this case you will be not able to initiate provisioning of it from Open API if it is not shared hosting subscription.
- 2 Obsolete model that supports only shared hosting subscriptions provisioning. In this model you should call `pem.addSubscription` to create subscription then set limits on resources using a series of `pem.setResourceTypeLimit` calls, and, finally, call `pem.provisionSubscription` method (on page 49) to provision the subscription. The external billing system shall use a separate method `pem.addDomain` to add domains. Please refer to the diagram (on page 119).
- 3 Starting from PEM version 2.5, the new scheme of "Pure Domains" is introduced. Using this scheme, you can create domains without hosting. In this case, the parameter `hosting_type` should be turned off, when you call `pem.addDomain` method. At the same time, there is a possibility to create domains with automatically included Mail Hosting. In this case, the activation parameter `Auto host domains` should be turned to "yes" in the Service Template.

`pem.activateSubscription`

This method is supported by PEM starting from version 2.3.01.

This method creates subscription, sets resource limits for this subscription, and then executes subscription provisioning:

Name	Type	Short Description
<code>[subscription_id]</code>	int	id of subscription to activate. If the parameter is not specified then ID for new subscription will be generated by the system.
<code>account_id</code>	int	id of account to subscribe

<code>service_template_id</code>	int	id of service template
<code>[resource_limits]</code>	array of struct { <code>rt_id</code> , <code>limit</code> }	resource limits
<code>[parameters]</code>	array of struct { <code>var_name</code> , <code>var_value</code> }	activation parameters for resources

The method has the following output parameter.

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription to activate

For XML-RPC sample of using this method see `pem.activateSubscription` Sample (on page 100) in XML-RPC Samples appendix (on page 98).

Parameters Details

`subscription_id` - The ID of the subscription to activate.

`account_id` - The ID of account to subscribe.

`service_template_id` - The ID of the service template that should be used for subscription creation.

`resource_limits` - Subscription resource limits. It is optional parameter. If omitted then Service Template's limits used. Each element of array is struct of two integer values. The first is the resource type id, the second is the resource type limit.

`parameters` - The list of parameters (`var_name`, `var_value`) pairs that should be used during subscription resources provisioning. It is optional parameter. Each element of array is struct of two string values, the first is the parameter name, the second is the parameter value.

`pem.addSubscription`

This method is supported by PEM starting from version 2.3.

This method adds a subscription, created by external billing system, into PEM without provisioning it.

The method has the following input parameters:

Name	Type	Short Description
<code>account_id</code>	int	id of account
<code>[subscription_id]</code>	int	id of subscription to add, generated by external system. If the parameter is not specified then ID for new subscription will be generated by the system.
<code>service_template_id</code>	int	id of base service template

The method has the following output parameter.

Name	Type	Short Description
subscription_id	int	id of subscription to add

See also:

Ordering a Subscription diagram (on page 119).

pem.addDomain method.

4 pem.provisionSubscription method (on page 49).

Parameters Details

account_id - Indicates the account, which the subscription being added relates to.

subscription_id - The ID of the subscription to add. This ID is generated by external billing system.

service_template_id - Indicates the existing service template, which the subscription being added is based on.

pem.getServiceTemplate

This method is supported by PEM starting from version 2.6.1

The pem.getServiceTemplate allows to retrieve information about specific PEM Service Template.

The method has the following input parameters:

Name	Type	Short Description
st_id	int	Service Template ID
get_resources	boolean	If method should return ST resources along with general information.

This method has the following output parameters:

Name	Type	Short Description
st_id	int	Service Template ID
version	int	Service Template Version
name	string	Service Template Name
owner_id	int	ID of account that own the Service Template
is_active	boolean	Is ST active (can be subscribed)

autoprovidable	boolean	Is the subscription based on this ST can be auto-provided?
[resource_types]	array of structs	
▪ name	string	Name of Resource Type
▪ resource_type_id	int	ID of the Resource Type
▪ [parent_resource_type_id]	int	ID of parent Resource Type (if any)
▪ measurement_unit	string	Unit that used for measurement of resource quantity
▪ limit	bigint	Limit set for specific Resource Type

The *resource_types* output parameter will be present only if *get_resources* input parameter is set to true.

Note: PEM resource types are organizing tree-like structure. The `pem.getServiceTemplate` method returns plain list of resource types in a sake of simplicity. However, the actual resource type tree can be reproduced using *parent_resource_type_id* member.

pem.getSubscription

This method is supported by PEM starting from version 2.6.1

The `pem.getSubscription` allows to retrieve information about specific PEM Subscriptions.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	Subscription ID
get_resources	boolean	If method should return Subscription resources along with general information.

This method has the following output parameters:

Name	Type	Short Description
subscription_id	int	Subscription ID
name	string	Subscription Name
st_id	int	Service Template ID
st_version	int	Service Template Version
owner_id	int	ID of account that own the Subscription
is_active	boolean	Is Subscription active

[resource_types]	array of structs	
▪ name	string	Name of Resource Type
▪ resource_type_id	int	ID of the Resource Type
▪ [parent_resource_type_id]	int	ID of parent Resource Type (if any)
▪ measurement_unit	string	Unit that used for measurement of resource quantity
▪ limit	bigint	Limit set for specific Resource Type
▪ usage	bigint	Amount of consumed resources of specific Resource Type

The *resource_types* output parameter will be present only if *get_resources* input parameter is set to true.

Note: PEM resource types are organizing tree-like structure. The `pem.getServiceTemplate` method returns plain list of resource types in a sake of simplicity. However, the actual resource type tree can be reproduced using *parent_resource_type_id* member.

pem.disableSubscription

This method is supported by PEM starting from version 2.3.

This method disables all services provisioned under specified subscription. Moreover, new services under this subscription cannot be provisioned until the subscription is enabled.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	id of subscription to disable

The method has no output parameters.

See also:

`pem.enableSubscription` method (on page 47).

Parameters Details

`subscription_id` - The ID of the subscription to disable.

`pem.enableSubscription`

This method is supported by PEM starting from version 2.3.

This method enables all services provisioned under specified subscription if they were disabled. Moreover, new services can be provisioned under the subscription after this method is called for it.

Note: if `pem.disableSubscription` (on page 90) method was called for a subscription, then it is not guaranteed that, after calling `pem.enableSubscription` method, all services will be in the same state as they were just before `pem.disableSubscription` (on page 90) method was called; it depends on particular kind of services.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	<code>int</code>	id of subscription to enable

The method has no output parameters.

See also:

`pem.disableSubscription` (on page 90) method.

Parameters Details

`subscription_id` - The ID of the subscription to enable.

`pem.migrateSubscription`

This method is supported by PEM starting from version 2.5.

This method migrates a subscription to the specified service template.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	<code>int</code>	id of subscription to migrate

<code>service_template_id</code>	int	id of the service template to migrate the subscription to
----------------------------------	-----	---

The method has no output parameters.

Parameters Details

`subscription_id` - The ID of the subscription to migrate.

`service_template_id` - The ID of the service template to migrate the subscription to.

pem.orderSubscription (deprecated)

This method is supported by PEM starting from version 2.3.

This deprecated method was used for creation only Shared Hosting subscriptions and will be removed from API in next PEM versions. This method creates a subscription, adds a domain and provisions the subscription in one call.

Instead of calling “pem.orderSubscription”, external billing system shall now call pem.activateSubscription method. Following parameters set should be passed as “parameters” parameter to pem.activateSubscription call for Shared Hosting subscriptions: *domain_name*, *domain_id (optional)*, *registrar_status*.

The method has the following input parameters:

Name	Type	Short Description
<code>account_id</code>	int	id of account
<code>subscription_id</code>	int	id of subscription to order
<code>service_template_id</code>	int	base service template id
<code>[domain_id]</code>	int	id of domain to add
<code>domain_name</code>	string	name of domain to add
<code>registrar_status</code>	int	initial registrar status for domain

The method has no output parameters.

See also:

Ordering a Subscription diagram (on page 119).

`pem.addDomain` method.

`pem.addSubscription` method.

5 `pem.provisionSubscription` method (on page 49).

`pem.setResourceTypeLimit` method.

Parameters Details

`account id` - Indicates the account, which the subscription being added/provisioned relates to.

`subscription_id` - The ID of the subscription to add/provision. This ID is generated by external billing system.

`service_template_id` - Indicates the existing service template, which the subscription being added/provisioned is based on.

`domain_id` - The ID of the domain to add. This parameter is optional and present here for backward compatibility purposes.

`domain_name` - Name of the domain to add.

`registrar_status` - Indicates the registrar domain status. It can take the following values:

- 0. Domain check result is undefined.
- 1. Domain is not registered.
- 2. Domain is registered.

`pem.provisionSubscription` (deprecated)

This method is supported by PEM starting from version 2.3.01.

This method was used for provisioning Shared Hosting subscriptions only. Use `pem.activateSubscription` method to create and provisions a subscription in PEM. In the future this method will be enhanced to support all kinds of PEM subscriptions to initiate provisioning of subscriptions created by `pem.addSubscription` call.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription to provision
<code>[provisioned_resource_types]</code>	array (struct)	resource types to provide

The method has no output parameters.

See also:

Ordering a Subscription diagram (on page 119).

`pem.addDomain` method.

`pem.addSubscription` method.

`pem.setResourceTypeLimit` method.

Parameters Details

`subscription_id` - The ID of the subscription to provision.

`provisioned_resource_types` - This parameter indicates which resource types must be auto-provisioned and in what amount. It is an optional parameter. If it is omitted, then the auto-provisioning of resource types is performed according to default rules in PEM. This parameter represents an array of structs, each describing auto-provisioning for a separate resource type. Structs have the following fields:

- `"resource_type_id"` of type "int". Indicated the resource type to auto-provision.
- `"amount"` of type "int". Specifies the amount of resources to auto-provision.

`pem.removeSubscription`

This method is supported by PEM starting from version 2.3.

This method unsubscribes Reseller or Customer from services provisioned under the specified subscription.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription to remove

The method has no output parameters.

Parameters Details

subscription_id - The ID of the subscription to remove.

pem.setSubscriptionName

This method is supported by PEM starting from version 2.6.1

The pem.setSubscriptionName method allows to set new name and description for an existent subscription.

This method has the following input parameters:

Name	Type	Short Description
sub_id	int	Subscription ID
[name]	string	
[descr]	string	

This method has no output parameters.

pem.upgradeSubscription (deprecated)

This method is supported by PEM starting from version 2.3.

This method was used to upgrade a current subscription to the specified Service Template. Now, when subscription migration technology is implemented for PEM 2.5., please use the pem.migrateSubscription method (on page 47).

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	id of subscription to upgrade
service_template_id	int	id of the service template to upgrade the subscription to

The method has no output parameters.

Parameters Details

subscription_id - The ID of the subscription to upgrade.

service_template_id - The ID of the service template to upgrade the subscription to.

Domains Management

pem.addDomain

This method is supported by PEM starting from version 2.3.

- 6** This method adds a domain, registered by external billing system, into PEM under the corresponding subscription. It does not provision the corresponding subscription in PEM; use pem.provisionSubscription (on page 49) method for that purpose.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	id of subscription under which domain is added
[domain_id]	int	domain id parameter
domain_name	string	name of domain to create
registrar_status	int	initial registrar status for domain
[hosting_type]	int	hosting (physical or parked) for new domain
[domain_parking_type]	int	Type of domain parking.
[path]	string	path or url to redirect

The method has the following output parameter.

Name	Type	Short Description
domain_id	int	id of domain to add

For XML-RPC sample of using this method see pem.addDomain Sample (on page 109) in the XML-RPC Samples Appendix (on page 98).

See also:

Ordering a Subscription diagram (on page 119).

7 pem.provisionSubscription method (on page 49).

Parameters Details

`subscription_id` - Indicates the existing subscription under which the domain is added.

`domain_id` - The ID of the domain to add. This parameter is optional and present here for backward compatibility purposes.

`domain_name` - Name of the domain to add.

`registrar_status` - Indicates the registrar domain status. It can take the following values:

- 0. Domain check result is undefined.
- 1. Domain is not registered.
- 2. Domain is registered.

`hosting_type` - Optional parameter. If specified, indicates hosting type for the domain. It can take the following values:

- 0. Physical hosting.
- 1. RESERVED. Do not use!
- 2. Parked domain.

Note: If you wish to add new Domain without hosting at all, do not issue this parameter in the XML-RPC request.

`domain_parking_type` - this parameter can be issued only if `hosting_type` parameter is set to 2 (parked domain). It can take the following values:

- 0. Domain parking.
- 1. Standard forwarding (welcome page customization).
- 2. Single page website.
- 3. Frame forwarding.

If this parameter is omitted and `hosting_type` is set to 2, then the domain parking will be used.

`path` - Optional parameter. If specified, its meaning depends on the hosting type:

- Physical hosting: path to document root location relatively to document root of main domain.
- Parked domain: destination URL for forwarding. If it is omitted, requests will be forwarded to a default location defined by provider.

pem.addSubdomain

This method is supported by PEM starting from version 2.3.

- 8** This method adds a subdomain into PEM under the corresponding subscription. It does not provision the corresponding subscription in PEM; use `pem.provisionSubscription` (on page 49) method for that purpose.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription to use
<code>domain_name</code>	string	name of the domain to add the new subdomain under
<code>prefix</code>	string	prefix of subdomain
<code>hosting_type</code>	int	hosting type
<code>path</code>	string	path to document root or url to redirect

The method has no output parameters.

Parameters Details

`subscription_id` - Indicates the existing subscription under which the subdomain is added.

`domain_name` - Name of the domain to add the new subdomain under.

`prefix` - Name of the prefix for subdomain to add. The new subdomain will be accessible at `http://<prefix>.<domain_name>`

`hosting_type` - Indicates hosting type for the subdomain. It can take the following values:

- 0. Physical hosting.

`path` - The meaning of this parameter depends on the hosting type:

- Physical hosting: path to document root location relatively to document root of main domain.

pem.disableDomain

This method is supported by PEM starting from version 2.3.

This method disables domain in PEM. It means that all domain's DNS records (except for SOA) will be substituted for record that points to a designated web site. This site may show a notification message like, for example, "Please, check your domain subscription".

The method has the following input parameters:

Name	Type	Short Description
domain_id	int	id of domain to disable
[domain_name]	string	name of domain to disable, if specified – domain_id is ignored)

The method has no output parameters.

See also:

pem.enableDomain method (on page 55).

Parameters Details

domain_id - The ID of the domain to disable.

domain_name - Name of the domain to disable. This parameter is optional. If it is specified, then <domain_id> parameter is ignored. *This parameter is supported by PEM starting from version 2.5.*

pem.enableDomain

This method is supported by PEM starting from version 2.3.

This method enables the specified domain in PEM. It means that all domain's DNS records will be restored to the values they have before the domain was disabled.

The method has the following input parameters:

Name	Type	Short Description
domain_id	int	id of domain to enable
[domain_name]	string	name of domain to enable, if specified – domain id is ignored)

The method has no output parameters.

See also:

`pem.disableDomain` method (on page 55).

Parameters Details

`domain_id` - The ID of the domain to enable.

`domain_name` - Name of the domain to enable. This parameter is optional. If it is specified, then `<domain_id>` parameter is ignored. *This parameter is supported by PEM starting from version 2.5.*

`pem.getDomainInfo`

This method is supported by PEM starting from version 2.7.

This method returns information about a domain.

The method has the following input parameters:

Name	Type	Short Description
<code>domain_id</code>	int	Domain ID to provide information about

Method output parameters are the following:

Name	Type	Short Description
<code>domain_name</code>	string	Fully qualified domain name
<code>domain_type</code>	string	PEM domain type (see <code>pem.getDomainList</code> (on page 57) for possible values list)
<code>hosting_type</code>	string	PEM hosting type (see <code>pem.getDomainList</code> (on page 57) for possible values list)
<code>[webspaces_id]</code>	int	Domain ID of the default domain which actually hosts this one
<code>location</code>	string	Webspaces location URI
<code>in_sync</code>	boolean	Whether domain is in-sync with PEM database

pem.getDomainList

This method is supported by PEM starting from version 2.5.

This method returns the list of all domains for a subscription.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	Customer's subscription ID

This method has the following output parameters:

Name	Type	Short Description
domain_id	int	ID of a customer's domain
domain_name	string	Fully qualified domain name
domain_type	string	PEM domain type
domain_hosting_type	string	PEM hosting type
registrar_status	int	Numeric registrar status
is_locked	boolean	If domain is disabled
dns_management_allowed	boolean	If a customer is allowed to manage domain's DNS records

Parameters Details

Output details:

domain_type contains symbolic alias for PEM domain type.

The current set of the possible domain types are:

- DOMAIN
- SUBDOMAIN

domain_hosting_type contains symbolic alias for PEM hosting type.

Current set of possible hosting types are:

- REGULAR
- SLAVE
- NONE

pem.getDomainSubscription

This method is supported by PEM starting from version 2.4.

This method returns an ID of subscription the particular domain belongs to.

This method has the following input parameters:

Name	Type	Short Description
domain_id	int	id of a domain
domain_name	string	Domain name

These parameters are optional to each other. It means that you can specify either domain_id parameter or domain_name one in your request.

The output parameters:

Name	Type	Short Description
subscription_id	int	An ID of subscription that owns the specified domain.

pem.getWebspacesList

This method is supported in PEM starting from version 2.4.

This method returns the list of webspaces' home directories for specified subscription.

It has the following input parameters:

Name	Type	Short Description
subscription_id	int	ID of the subscription

Output is an array of following structures:

Name	Type	Short Description
host_ip	int	IP address of the host the corresponding service is running on
webspace_id	int	ID of a webspace
path	string	Path to webspace on the host
domain_id	int	ID of the instant access URL for the webspace
domain_name	string	Instant access URL for the webspace

pem.getNameServers

This method is supported by PEM starting from version 2.3.

This method returns name servers that serve the specified domain.

The method has the following input parameters:

Name	Type	Short Description
domain_name	string	name of domain

The method has the following output parameters:

Name	Type	Short Description
N/A	array (string)	list of nameservers names

Parameters Details

Input Parameters:

domain_name - Name of the domain to return name servers for.

Output Parameters:

N/A - Contains names of name servers serving the requested domain.

pem.importCertificate

This method is supported by PEM starting from version 2.3.

This method allows import SSL certificate on a specific site.

Name	Type	Short Description
domain_id	int	id of domain to add a certificate to
cert	string	certificate sent in raw data form
priv_key	string	certificate's private key in raw data
ca_cert	string	CA certificate

The `ca_cert` parameter and the pair of parameters of `cert`, `priv_key` are optional each to other. They can be transmitted both or any one of these as well. Method has no output values.

Parameters Details

`domain_id` - ID of domain to add a certificate to.

`cert` - Certificate sent in a raw data form.

`private_key` - Certificate's private key in raw data.

`ca_cert` - CA Certificate

pem.removeDomain

This method is supported by PEM starting from version 2.3.

"DNS Hosting" should be removed for specified domain. It means that PEM named services does not longer responsible for DNS records of domain. All customer services should stay untouched (Apache, FTP, File Manager).

The method has the following input parameters:

Name	Type	Short Description
<code>domain_id</code>	int	id of domain to remove
<code>[domain_name]</code>	string	name of domain to remove, if specified – <code>domain_id</code> is ignored)

The method has no output parameters.

Parameters Details

`domain_id` - The ID of the domain to remove.

`domain_name` - Name of the domain to remove. This parameter is optional. If it is specified, then `<domain_id>` parameter is ignored. *This parameter is supported by PEM starting from version 2.5.*

pem.removeSubdomain

This method is supported by PEM starting from version 2.7.

This method removes subdomain from PEM.

The method has the following input parameters:

Name	Type	Short Description
<code>domain_name</code>	string	name of parent domain

<code>prefix</code>	string	prefix of subdomain to be removed
---------------------	--------	-----------------------------------

The method has no output parameters.

Parameters Details

`domain_name` - Name of the domain to remove the subdomain under.

`prefix` - Name of the prefix for subdomain to remove.

pem.setDomainHostingType

This method is supported by PEM starting from version 2.7.

This method changes hosting type for domain in PEM.

The method has the following input parameters:

Name	Type	Short Description
<code>domain_name</code>	string	name of domain
<code>hosting_type</code>	int	new hosting type
<code>path</code>	string	path to document root or url for redirect
<code>[tld_name]</code>	string	TLD name, for parked domain only

The method has no output parameters.

Parameters Details

`domain_name` - Name of the domain to change hosting type for.

`hosting_type` - Indicates new hosting type for the domain. It can take the following values:

- 0. Physical hosting.
- 1. Forwarding.
- 2. Parked domain.

`path` - Meaning of this parameter depends on the hosting type:

- Physical hosting: path to document root location relatively to document root of main domain.
- Forwarding: destination URL for forwarding.
- Parked domain: destination URL for forwarding. If it is omitted, requests will be forwarded to a default location defined by provider.

`tld_name` - Name of the Top Level Domain for parked domain. This parameter is required for parked domain only.

pem.setDomainRegistrarStatus

This method is supported by PEM starting from version 2.3.

This method sets new registrar status for given domain in PEM.

The method has the following input parameters:

Name	Type	Short Description
<code>[domain_id]</code>	int	id of domain, if specified – <code>domain_name</code> is ignored
<code>domain_name</code>	string	name of domain
<code>registrar_status</code>	int	new registrar status

The method has no output parameters.

Parameters Details

`domain_id` - [Optional] Unique PEM domain identifier to set new registrar status for. If specified `<domain_name>` parameter ignored.

`domain_name` - Name of the domain to set new registrar status for.

`registrar_status` - Indicates the new registrar domain status. It can take the following values:

- 0. Domain check result is undefined.
- 1. Domain is not registered.
- 2. Domain is registered.

pem.setSubdomainsHostingType

This method is supported by PEM starting from version 2.7.

This method changes hosting type for one or all subdomains under given domain in PEM.

The method has the following input parameters:

Name	Type	Short Description
domain_name	string	name of parent domain
[prefix]	string	prefix of subdomain to modify
hosting_type	int	new hosting type
path	string	path to document root or url to redirect

The method has no output parameters.

Parameters Details

domain_name - Name of the domain to change hosting type for subdomain(s).

prefix - Name of the prefix for subdomain to change hosting type for. This is optional parameter. If omitted, then changes are applied for all subdomains under selected domain (<domain_name>).

hosting_type - Indicates new hosting type for the subdomain. It can take the following values:

- 0. Physical hosting.
- 1. Forwarding.

path - Meaning of this parameter depends on the hosting type:

- Physical hosting: path to document root location relatively to document root of main domain.
- Forwarding: destination URL for forwarding.

Webspace Management

pem.getFTPUser

This method is supported by PEM starting from version 2.4.

This method returns the login name for primary FTP user for FTP hosting associated with the specific subscription. The method supports proFTPD and IIS hosting.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	ID of the subscription
[webspace_id]	int	ID of the webspace served by the chosen FTP service
[domain_name]	string	Name of the domain served by the chosen FTP service

The [webspace_id] or [domain_name] parameters should be used when a subscription contains more than one webspace served by the FTP service in order to determine, which exactly service to use. These parameters are optional to each other, so only one of them can be used in a single request. If neither of them is specified, the method will act as if there is only one webspace in the subscription and will result in error if it's not so.

Output is a structure with following content:

Name	Type	Short Description
username	string	The requested login name.

pem.setFTPPassword

This method is supported by PEM starting from version 2.4.

This method changes the password of the primary FTP user from FTP hosting associated with the specific subscription. The method supports proFTPD and IIS hosting.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	ID of the subscription
password	string	The password to set

<code>password_crypted</code>	boolean	Whether the password is encrypted
<code>[webpace_id]</code>	int	ID of the webpace served by the chosen FTP service
<code>[domain_name]</code>	string	Name of the domain served by the chosen FTP service

The `[webpace_id]` or `[domain_name]` parameters should be used when a subscription contains more than one webpace served by the FTP service in order to determine, which exactly service to use. These parameters are optional to each other, so only one of them can be used in a single request. If neither of them is specified, the method will act as if there is only one webpace in the subscription and will result in error if it's not so.

This method has no output parameters.

Note: IIS does not support the encrypted passwords.

pem.web.getFileManagerInfo

This method is supported by PEM starting from version 2.7.

This method returns FileManager configuration for a webpace.

The method has the following input parameters:

Name	Type	Short Description
<code>webpace_id</code>	int	ID of a webpace domain

Output is a structure with following content:

Name	Type	Short Description
<code>is_enabled</code>	boolean	Whether FileManager service is enabled
<code>url</code>	string	FileManager site URL.
<code>[login]</code>	string	This field contain login in case when FileManager engine does not accept login/password in URL
<code>[password]</code>	string	This field contain password in case when FileManager engine does not accept login/password in URL

pem.web.getFrontPageInfo

This method is supported by PEM starting from version 2.7.

Returns current FrontPage configuration for the webspace.

The method has the following input parameters:

Name	Type	Short Description
webspace_id	int	ID of a webspace domain

Output is a structure with the following content:

Name	Type	Short Description
fp_supported	boolean	Whether FrontPage is supported
fp_ssl_supported	boolean	Whether FrontPage over SSL is supported.
authoring_enabled	boolean	Whether FrontPage authoring is enabled.
login	string	Login
password	string	Password in plain text

pem.web.getFTPConf

This method is supported by PEM starting from version 2.7.

This method returns FTP configuration for a webspace.

The method has the following input parameters:

Name	Type	Short Description
webspace_id	int	ID of a webspace domain

This method has the following output parameters:

Name	Type	Short Description
is_enabled	boolean	Whether FTP service is enabled.
server_name	string	Fully qualified URL of FTP server.
server_ip	string	IP address of FTP server.
server_port	int	Address port of FTP server.
login	string	Username to access FTP webspace repository.
password	string	Password to access FTP webspace repository.

pem.web.getHttpErrorDocs

This method is supported by PEM starting from version 2.7.

This method returns a list of currently configured HTTP error responses.

The method has the following input parameters:

Name	Type	Short Description
webspaces_id	int	ID of a webspaces domain

Output is an array of structures with following content:

Name	Type	Short Description
code	int	HTTP error response code.
description	string	Short description about error sense.
mode	string	Error treating mode {DEFAULT, CUSTOM, FILE, URL}.
custom_msg	string	Custom error message in case when mode == CUSTOM.
uri	string	File path or document's URL in case when mode == FILE or URL.

pem.web.getMimeTypes

This method is supported by PEM starting from version 2.7.

This method returns a list of MIME types specifications for a particular webspaces.

The method has the following input parameters:

Name	Type	Short Description
webspaces_id	int	ID of a webspaces domain

This method has the following output parameters:

Name	Type	Short Description
mime_type	string	MIME type specification
extensions	array (string)	An array of file extensions that match this MIME type

pem.web.getSiteProps

This method is supported by PEM starting from version 2.7.

This method returns webspace-wide configuration information.

The method has the following input parameters:

Name	Type	Short Description
webspace_id	int	ID of a webspace domain

Output is a structure with following content:

Name	Type	Short Description
webspace_name	string	Fully qualified domain name of subscription's default domain.
webspace_ip	string	Default domain's IP
webspace_ip_type	string	IP hosting type for default domain {SHARED, EXCLUSIVE}
server_type	string	Name of HTTP server {Apache, IIS}
server_version	int	Major number of HTTP server version
features_support	array (struct)	List of web-features with information about supporting
▪ name	string	Feature name
▪ is_supported	boolean	Whether its support is turned on
▪ is_available	boolean	Webspace's resources provide ability to turn the feature on

The list of possible features:

Feature Name	Server Support	Notes
SSL	Apache, IIS	
CGI	Apache, IIS	
SSI	Apache, IIS	
PHP	Apache, IIS	
ErrorDocs	Apache, IIS	Error documents customization ability. Always supported.
WAP	Apache	
FPSE	Apache, IIS	Front Page Server Extensions.
SharePoint	IIS	
FTP	IIS	Linux hosting has standalone FTP support.
ASP	IIS	
ASP.NET	IIS	

ColFusion	IIS	
-----------	-----	--

pem.web.getSSLInfo

This method is supported by PEM starting from version 2.7.

The method returns information about SSL support of particular webspace.

The method has the following input parameters:

Name	Type	Short Description
webspace_id	int	ID of a webspace domain
[domain_id]	int	Domain ID. Specify it if your web-server provides per-domain SSL support only (i.e. IIS).

Output is a structure with following content:

Name	Type	Short Description
is_supported	boolean	Whether there is SSL access supported for this webspace/domain.
document_root	string	If web-server provides different roots for secured and usual content, this field contains a path to secured content root.

pem.addWebSpaceBackup

This method is supported by PEM starting from version 2.7.

This method creates new backup for selected items in a Web Space.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	id of subscription
webspace_id	int	id of webspace to backup
backup_name	string	name for new backup
backup_items_ids	array (int)	ids of items to backup

The method has the following output parameters:

Name	Type	Short Description
backup_id	int	id of new backup

Parameters Details

Input Parameters:

subscription_id - The ID of the subscription to create backup under.

webspace_id - The ID of the Web Space to create backup for. This parameter is optional. If omitted, the default Web Space is selected.

backup_name - Name of the backup to create.

backup_items_ids - Contains IDs of items in the Web Space to backup.

Output parameters:

backup_id - ID of the created backup.

pem.setFrontPageParameters

This method is supported by PEM starting from version 2.7.

This method changes parameters of FrontPage Extensions for given Web Space.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	id of subscription
[webspace_id]	int	id of webspace
enabled	int	enable FrontPage extensions
ssl_enabled	int	enable SSL
auth_enabled	int	enable authorization
login	string	login name
password	string	password

The method has no output parameters.

Parameters Details

`subscription_id` - The ID of the subscription to change parameters of FrontPage Extensions under.

`webspace_id` - The ID of the Web Space to change parameters of FrontPage Extensions for. This parameter is optional. If omitted, the default Web Space is selected.

`enabled` - Identifies if support for FrontPage Extensions is enabled or disabled for this Web Space. This parameter can take the following values:

- 0. Support for FrontPage Extensions is disabled.
- 1. Support for FrontPage Extensions is enabled.

`ssl_enabled` - Identifies if support for FrontPage Extensions over SSL is enabled or disabled for this Web Space. This parameter can take the following values:

- 0. Support for FrontPage Extensions over SSL is disabled.
- 1. Support for FrontPage Extensions over SSQL is enabled.

`auth_enabled` - Indicates if authorization is enabled or disabled. This parameter can take the following values:

- 0. Authorization is disabled.
- 1. Authorization is enabled.

`login` - The login name.

`password` - Password for the login name.

Mail Management

`pem.cqmail.addMailbox`

This method is supported by PEM starting from version 2.4.

This method creates new mailbox using specified subscription.

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription
<code>name</code>	string	name of mailbox to be created
<code>[domain_name]</code>	string	domain name
<code>[password]</code>	string	mailbox password
<code>[password_crypted]</code>	boolean	whether the password is encrypted
<code>quota</code>	int	mailbox quota, 0 for no quota
<code>[mail_to_notify]</code>	string	mail address to be notified

<code>antispam</code>	boolean	use spam protection
<code>antivirus</code>	struct	use antivirus protection
▪ <code>check_in</code>	boolean	checking incoming mail
▪ <code>check_out</code>	boolean	checking outgoing mail
▪ <code>notify_sender</code>	boolean	notification e-mail will be sent to the mail sender if a virus is found
▪ <code>notify_receipients</code>	boolean	notification e-mail will be sent to recipients if a virus is found

The method has no output parameters.

Parameters Details

`subscription_id` - The ID of the subscription.

`name` - The name of the mailbox. It should not contain domain part.

`domain_name` - Domain name to create mailbox in. If empty (or missed), the "all-domain" mailbox will be created.

`password` - Password for accessing a mailbox. Optional parameter.

`password_crypted`- True if password was encrypted (DES or BSD MD5), false otherwise.

Parameters "password" and "password_crypted" are optional. If these parameters are omitted, then the method's code supposes that the corresponding e-mail name is already registered in PEM and just tries to add mailbox to it. If they are not, then the method tries to create an e-mail name in PEM and then associate the mailbox with it.

`quota` - Disk quota, 0 for unlimited.

`mail_to_notify` - Mail address to send notification when the mailbox will be created (optional).

`antispam` - defines whether this mailbox will be filtered through SPAM protection.

`antivirus` - defines whether this mailbox will be filtered through antivirus protection system.

- `check_in` - checks all the incoming mail for viruses.
- `check_out` - checks all the outgoing mail for viruses.
- `notify_sender` - sends E-mail notifications when a virus found in the incoming mail.
- `notify_receipients` - sends E-mail notifications when a virus found in the outgoing mail.

pem.cqmail.addMailForwarding

This method is supported by PEM starting from version 2.5.

This method associates forwarding address with the specified e-mail name.

Name	Type	Short Description
subscription_id	int	id of subscription
name	string	name of mailbox to be created
[domain_name]	string	domain name
forward_to	string	address to forward e-mail to
[password]	string	mailbox password
[password_crypted]	boolean	whether the password is encrypted

This method is not able to override password for already existent mailbox.

pem.cqmail.getAutoresponderInfo

This method is supported by PEM starting from version 2.7.

This method returns autoresponder's properties.

It has the following input parameters:

Name	Type	Short Description
autoresp_id	int	An autoresponder ID

Output is an array of structure with following scheme:

Name	Type	Short Description
autoresp_name	string	An autoresponder name
is_active	boolean	Whether this autoresponder is operational.
what_filter	string	Where search for filter match or process autoresponding anyway. One of {BODY, SUBJECT, ALWAYS}.
filter	string	Filter string
resp_subject	string	Custom response subject. If empty then it will be: "Re: <Original subject>"
resp_body	string	Actual auto-response text.
replyto	string	Set reply address for autoresponse message.

pem.cqmail.getAutoresponderList

This method is supported by PEM starting from version 2.7.

This method returns the list of autoresponders configured for the specified mailname.

The method has the following input parameters:

Name	Type	Short Description
mailname_id	int	Mailname ID to find out about its autoresponders list.

Output is an array of structures with following scheme:

Name	Type	Short Description
autoresp_id	int	Autoresponder ID
autoresp_name	string	Autoresponder name
is_active	boolean	Whether this autoresponder is operational.

pem.cqmail.getForwardingsList

This method is supported by PEM starting from version 2.7.

This method returns the list of mailname forwarding properties.

The method has the following input parameters:

Name	Type	Short Description
mailname_id	int	Mailname ID to find out about its forwarding addresses.

Output is an array of structures with following scheme:

Name	Type	Short Description
fwd_id	int	Forwarding ID
email	string	E-mail to forward to.
is_enabled	boolean	Whether this particular forwarding is enabled

pem.cqmail.getMailboxInfo

This method is supported by PEM starting from version 2.7.

This method returns mailbox properties.

The method has the following input parameters:

Name	Type	Short Description
mailname_id	int	Mailname ID of a mailbox

Output is a structure with following scheme:

Name	Type	Short Description
state	int	0 = does not exist; 1 = On; 2 = Off.
quota	int	Disk quota for the configured mailbox.

pem.cqmail.getMailHostingInfo

This method is supported by PEM starting from version 2.7.

This method gets configuration for mail hosting of the subscription.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	id of subscription

Output is a structure with the following scheme:

Name	Type	Short Description
is_enabled	boolean	Whether mail hosting is enabled
pop3_server	string	POP3 server hostname
imap_server	string	IMAP server hostname
smtp_server	string	SMTP server hostname
webmail_url	string	WebMail site URL
relay_auth_type	string	SMTP server relay authentication type, one of {SMTP_AUTH, POP_BEFORE_SMTP}
throtling	struct	Mail delivery limit (one recipient – one delivery). 0 for unlimited

▪ <code>day_limit</code>	int	Per day
▪ <code>hour_limit</code>	int	Per hour, <code>hour_limit <= day_limit</code>
▪ <code>minute_limit</code>	int	Per minute, <code>minute_limit <= hour_limit</code>
<code>max_mailbox_quota</code>	int	Max quota per mailbox

pem.cqmail.getMailLists

This method is supported by PEM starting from version 2.7.

This method returns the list of maillists for the specified subscription.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription

Output is an array of structures with following scheme:

Name	Type	Short Description
<code>maillist_id</code>	int	Maillist ID
<code>name</code>	string	Maillist name
<code>domain_name</code>	string	Domain name this maillist belongs to (empty in case of all-domains).
<code>status</code>	struct	
▪ <code>in_sync</code>	boolean	Whether maillist is in-sync with PEM database.
▪ <code>status_name</code>	string	Symbolic status name.

List of possible statuses:

Status Name	In Sync
READY	yes
ADDING	no
UPDATING	no
DELETING	no

pem.cqmail.getMailListInfo

This method is supported by PEM starting from version 2.7.

This method returns information about a maillist.

The method has the following input parameters:

Name	Type	Short Description
maillist_id	int	Maillist ID to get info about

Output is a structure with the following scheme:

Name	Type	Short Description
info	string	General information on the maillist
intro	string	Introductory details on the maillist

pem.cqmail.getMailListOwners, pem.cqmail.getMailListModerators, pem.cqmail.getMailListMembers

This method is supported by PEM starting from version 2.7.

This method returns the list of owners, moderators, members of the specified maillist.

The method has the following input parameters:

Name	Type	Short Description
maillist_id	int	Maillist ID to get info about

Output is a structure with the following scheme:

Name	Type	Short Description
id	int	An ID of a corresponding user (owner, moderator, member).
email	string	E-mail address of a corresponding user (owner, moderator, member)
status	struct	
▪ in_sync	boolean	Whether actual user's state is in-sync with the PEM database.
▪ status_name	string	Symbolic status name

Refer to pem.cqmail.getMailLists (on page 76) for the status list.

pem.cqmail.getMailnameList

This method is supported by PEM starting from version 2.7.

This method gets the list of mail names of the subscription.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	id of subscription

Output parameters are the following:

Name	Type	Short Description
mailname_id	int	Mailname ID
mailname	string	Mailname as is. It may have domain part. No domain part means all-domains-mailname.
login	string	Mailname authorization login (for SMTP, POP3, IMAP).
password	string	Mailname authorization password (for SMTP, POP3, IMAP).
in_sync	boolean	Whether mailname in-sync with PEM database.
mailbox_status	int	0 = does not exist; 1 = On; 2 = Off.
forwarding_status	int	0 = does not exist; 1 = On; 2 = Off.
autoresponder_status	int	0 = does not exist; 1 = On; 2 = Off.

Database Management

pem.createDatabase

This method is supported by PEM starting from version 2.3.

This method creates new database for specified subscription with specific type and default user.

Name	Type	Short Description
subscription_id	int	id of subscription
name	string	database name
type	string	type of RDBMS (MYSQL or PGSQL)
user	string	database user name

password	string	password for database user
----------	--------	----------------------------

This method returns structure with following data:

Name	Type	Short Description
database_id	int	id of created database
name	string	name of created database
hostname	string	name of host where database will be located

Parameters Details

Input Parameters:

subscription_id - The ID of the subscription.

name - Name for database to create.

type - Which RDBMS use to host a database. Possible values:

- MYSQL
- PGSQL

user - Default database user.

password - Password for default database user.

Output Parameters:

database_id - An ID of created database.

name - Name of created database.

hostname - Host where database's data is located.

pem.createDatabaseUser

This method is supported by PEM starting from version 2.3.

This method creates new database user which can access specific database.

One can create only one user with particular name so the user can access only one database.

Name	Type	Short Description
db_id	int	id of database new user will be able to access
user	string	name of new user
password	string	password for new user

Method will return an ID of created user.

The method has the following output parameters:

Name	Type	Short Description
user_id	int	ID of the created database user

Parameters Details

db_id - ID of Database (received via pem.createDatabase method (on page 78)).

user - User to create.

password - User's password.

pem.getDatabaseAccessHostList

This method is supported by PEM starting from version 2.7.

The method returns a list of hosts configured to access specific database.

The method has the following input parameters:

Name	Type	Short Description
database_id	int	Database ID to gather info about

This method has the following output parameters:

Name	Type	Short Description
account_host_id	int	PEM access host ID
host_ip	string	An IP address of a host that is configured to access a database.
status	struct	
▪ in_sync	boolean	Whether the database host info in-sync with PEM database.
▪ status_name	string	Refer to pem.getDatabaseList (on page 81) for the list of possible values.

pem.getDatabaseInfo

This method is supported by PEM starting from version 2.7.

The method returns information about any configured database.

The method has the following input parameters:

Name	Type	Short Description
database_id	int	Database ID to gather info about

Output is an array of structures with the following scheme:

Name	Type	Short Description
name	string	Database name
type	string	Database type; currently the one of {MYSQL, PGSQL, MSSQL}.
status	struct	
▪ in_sync	boolean	Whether the database is in-sync with PEM database.
▪ status_name	string	Refer to pem.getDatabaseList (on page 81) for the list of possible values.
host_ip	string	Host IP where database server runs
port	int	Port to connect to database server
manager_url	string	Web database manager URL

pem.getDatabaseList

This method is supported by PEM starting from version 2.7.

The method returns a list of databases that are configured for the specified subscription.

The method has the following input parameters:

Name	Type	Short Description
subscription_id	int	id of a customer's subscription

Output is an array of structures with the following scheme:

Name	Type	Short Description
name	string	database name
type	string	Database type; currently the one of {MYSQL, PGSQL, MSSQL}.

<code>status</code>	struct	
▪ <code>in_sync</code>	boolean	Whether the database is in-sync with PEM database.
▪ <code>status_name</code>	string	Symbolic status name.

List of possible statuses:

Status Name	In Sync
READY	yes
ADDING	no
UPDATING	no
DELETING	no

`pem.getDatabaseUserList`

This method is supported by PEM starting from version 2.7.

The method returns a list of DBMS-specific users that are configured for a particular database.

The method has the following input parameters:

Name	Type	Short Description
<code>database_id</code>	int	Database ID to gather info about

This method has the following output parameters:

Name	Type	Short Description
<code>user_id</code>	int	PEM user ID
<code>user_name</code>	string	DBMS login
<code>status</code>	struct	
▪ <code>in_sync</code>	boolean	Whether the database user is in-sync with PEM database.
▪ <code>status_name</code>	string	Refer to <code>pem.getDatabaseList</code> (on page 81) for the list of possible values.

Resource Accounting

pem.getResourceUsage

This method is supported by PEM starting from version 2.3.

This method returns up-to-date usage statistics for resource types within the specified subscription.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription
<code>resource_type_ids</code>	array (int)	ids of resource types in question

The method has the following output parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription
<code>resource_type_usages</code>	array (struct)	resource types usage statistics

The method has the following output parameters:

See also:

`pem.getResourceUsageForPeriod` method (on page 84).

Parameters Details

Input Parameters:

`subscription_id` - Indicates the subscription, which the resource usage is returned for.

`resource_type_ids` - Contains IDs of resource types, which the resource usage is requested for.

Output Parameters:

`subscription_id` - Indicates the subscription, which the resource usage is returned for.

`resource_type_usages` - Contains resource usages for each requested resource type. It is the array of structs, each having the following members:

- `"resource_type_id"` of type "int". Indicates the resource type the usage statistics is returned for.
- `"usage"` of type "int". Represents the absolute value of resource's usage at the moment defined by `"last_checkpoint"` parameter (see below). The usage is specified in units, which the resource type is measured in.
- `"last_checkpoint"` of type "int". Indicates the point in time when the usage of the resource was measured last time. It is specified in the number of seconds since 1970-01-01 00:00:00 UTC.

`pem.getResourceUsageForPeriod`

This method is supported by PEM starting from version 2.3.

This method returns usage statistics for resource types within the specified subscription for a given period of time. It is complimentary to the `pem.getResourceUsage` method (on page 83).

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription
<code>resource_type_ids</code>	array (int)	ids of resource types in question
<code>from_time</code>	int	start time of period (unix time)
<code>to_time</code>	int	end time of period (unix time)

The method has the following output parameters:

Name	Type	Short Description
resource_type_usages	array (struct)	usage statistics
actual_period	array (int)	actual periods

	from_time in history & to_time in history	from_time in history & to_time > history_end	from_time > history_end & to_time > history_end
actual_period			
from_time	from_time	from_time	history_end
to_time	to_time	history_end	history_end

Here “history_end” represents the point time when the given request for statistics is served (current time at callee side). “in history” means point in time that is earlier than “history_end”.

See also:

`pem.getResourceUsage` method (on page 83).

Parameters Details

Input Parameters:

`subscription_id` - Indicates the subscription, which the resource usage is returned for.

`resource_type_ids` - Contains IDs of resource types, which the resource usage is requested for.

`from_time` - Indicates the start time of the period, which the resource usage is returned for. Must be specified in the number of seconds since 1970-01-01 00:00:00 UTC.

`to_time` - Indicates the end time of the period, which the resource usage is returned for. Must be specified in the number of seconds since 1970-01-01 00:00:00 UTC.

Output Parameters:

`resource_type_usages` - Contains resource usages for each requested resource type. It is the array of structs, each having the following members:

- “`resource_type_id`” of type “int”. Indicates the resource type the usage statistics is returned for.
- “`parent_resource_type_id`” of type “int”. Indicates the parent resource type of the resource type the usage statistics is returned for. This value can be used to group usage statistics for related resource types.
- “`initial_usage`” of type “int”. Represents the absolute value of resource’s usage at the start of the requested period. The usage is specified in units, which the resource type is measured in.
- “`usage_statistics`”. The usage statistics itself. It is an array of structs. Each member of the array represents a particular change in resource usage during the requested period. Structs have the following members:
 - “`time`” of type “int”. Indicates the point in time when the usage of the resource was changed. It is specified in the number of seconds since 1970-01-01 00:00:00 UTC.
 - “`delta`” of type “int”. Contains the actual change of the resource’s usage. The value of the change may be negative depending on the resource’s type.

`actual_period` - Indicates the *actual* period, which the resource usages are returned for. For the description of what the actual period is, see the note below. This field is a struct with the following members:

- “`from_time`” of type “int”. Indicates the *actual* start time of the period, which the resource usages are returned for. It is specified in the number of seconds since 1970-01-01 00:00:00 UTC.

- “*to_time*” of type “int”. Indicates the *actual* end time of the period, which the resource usages are returned for. It is specified in the number of seconds since 1970-01-01 00:00:00 UTC.

Note: the caller may request usage statistics for a period of time for which there are no statistics in the system at all or there exist statistics for a part of the requested period only. The “*actual_period*” returned parameter serves the purposes to address such situations. Depending on the values of input “*from_time*” and “*to_time*” parameters and the period of time for which the statistics exist in the system, the “*actual_period*” returned parameter can take values described in the table below.

pem.resetResourceUsage

This method is supported by PEM starting from version 2.3.

This method resets the usage counter in PEM for specified resource types under the given subscription.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription
<code>resource_type_ids</code>	array (int)	ids of resource types

The method has no output parameters.

Parameters Details

`subscription_id` - Indicates the subscription, which the resource usage is reset for.

`resource_type_ids` - Contains IDs of resource types, which the resource usage is reset for.

pem.setResourceTypeLimit

This method is supported by PEM starting from version 2.6.1

This method sets limits on a resource type within given subscription.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	ID of the subscription
<code>resource_type_id</code>	int	ID of the resource type
<code>[limit]</code>	int	new resource type limit
<code>[limit64]</code>	bigint	64bit analog of limit parameter; has higher priority.
<code>[autoprovide]</code>	boolean	If the actual provisioning/un-provisioning should proceed

If *autoprovide* flag is set to true, than the method performs the following additional tasks:

- If a particular resource either added or changed its limit from zero to any non-zero value, then it will be auto-provided.
- If limit of a particular resource is changed from non-zero value to zero, then it will be automatically un-provided.

Note: PEM resource type structure has a tree-like form, so auto-provisioning of leaf resources is impossible without provisioning of root ones. Be aware of what resources to send with this method to get predictable results.

pem.setResourceTypeLimits

This method is supported by PEM starting from version 2.6.1

The `pem.setResourceTypeLimits` method is responsible for changing limits of set of resource types in a specific subscription. This method also can control resource provisioning status according to their new limits.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	Subscription ID
<code>[autoprovide]</code>	boolean	If the actual provisioning/un-provisioning should proceed
<code>resource_limits</code>	array of structs	

▪ <code>resource_type_id</code>	int	Resource Type ID
▪ <code>limit</code>	bigint	New limit for specific Resource Type

This method has no output parameters.

The method can include new resource types into current subscription's resource type tree.

If *autoprove* flag is set to true, than the method performs the following additional tasks:

- If a particular resource either added or changed its limit from zero to any non-zero value, then it will be auto-provided.
- If limit of a particular resource is changed from non-zero value to zero, then it will be automatically un-provided.

Note: PEM resource type structure has a tree-like form, so auto-provisioning of leaf resources is impossible without provisioning of root ones. Be aware of what resources to send with this method to get predictable results.

Applications Management

`pem.installWebApplication`

This method is supported by PEM starting from version 2.7.

This method installs Web application on specified domain using default configuration settings.

The method has the following input parameters:

Name	Type	Short Description
<code>subscription_id</code>	int	id of subscription
<code>domain_name</code>	string	name of domain
<code>app_rc_name</code>	string	name of application resource class

The method has the following output parameters:

Name	Type	Short Description
<code>application_id</code>	int	id of new application

Parameters Details

Input Parameters:

subscription_id - The ID of the subscription to provision the Web application under.

domain_name - Name of the domain to install Web application on.

app_rc_name - Name of the PEM™'s resource class identifying the Web application to install.

Output Parameters:

application_id - ID of the installed Web application.

pem.removeWebSpaceBackup

This method is supported by PEM starting from version 2.7.

This method removes backup of Web Space from PEM.

The method has the following input parameters:

Name	Type	Short Description
backup_id	int	id of backup to remove

The method has no output parameters.

Parameters Details

backup_id - The ID of the backup to remove.

pem.restoreWebSpaceFromBackup

This method is supported by PEM starting from version 2.7.

This method restores selected Web Space's items from previously created backup.

The method has the following input parameters:

Name	Type	Short Description
backup_id	int	id of backup to restore
[backup_items_ids]	array (int)	ids of concrete items to be restored

The method has no output parameters.

Parameters Details

`backup_id` - The ID of the backup to restore Web Space's items from.

`backup_items_ids` - Contains IDs of items in the Web Space to restore from backup. This parameter is optional. If it is omitted, then all items contained in the backup are restored.

`pem.uninstallWebApplication`

This method is supported by PEM starting from version 2.7.

This method uninstalls previously installed Web application.

The method has the following input parameters:

Name	Type	Short Description
<code>application_id</code>	int	id of application to uninstall

Parameters Details

`application_id` - ID of the Web application to uninstall.

Security Management

`pem.checkPassword`

This method is supported by PEM starting from version 2.4

This method checks password quality.

Name	Type	Short Description
<code>account_id</code>	int	account ID
<code>password</code>	string	password to check
<code>[name]</code>	string	Name to check that password doesn't contain personal information.
<code>[disabled_chars]</code>	string	Disabled characters.

This method has no output parameters.

Parameters Details

`account_id` - The ID of an account to check password for.

`password` - password to check.

`name` - name to check that password doesn't contain any personal information.

`disabled_chars` - disabled characters. The list of characters that can occur in password.

Native Package Management

`pem.packaging.native_repository.createRepository`

This method is supported by PEM starting from version 2.6.

This method allows to create new native repository.

Name	Type	Short Description
<code>repo_type</code>	struct	The structure that defines the type of packages that will be stored within repository.
▪ <code>arch</code>	string	Hardware architecture, eg. <i>i386</i>
▪ <code>opsys</code>	string	Name of operating system
▪ <code>osrel</code>	string	Version of operating system
▪ <code>pkg_manager</code>	int	Native package manager's ID: 0-YUM
<code>[host_id]</code>	int	If this parameter is specified, the <i>manageable</i> native repository will be created on particular PEM host.
<code>[is_local]</code>	boolean	This parameter determines whether <i>manageable</i> repository is the local one or is hosted by PPM Mirror.
<code>[url]</code>	string	This parameter specifies either the URI of <i>external</i> repository or the path for local repository.

The method has the following output parameters:

Name	Type	Short Description
<code>repo_id</code>	int	ID of created native repository
<code>url</code>	string	The URL of the repository
<code>is_created</code>	boolean	Defines whether all background work related to creating native repository is finished.
<code>is_ready</code>	boolean	Defines whether native repository has been ever re-indexed or it is external one.

pem.packaging.native_repository.getRepository

This method is supported by PEM starting from version 2.6.

This method returns all the information necessary to connect to specific native repository.

Name	Type	Short Description
repo_id	int	ID of particular native repository

This method has the following output parameters:

Name	Type	Short Description
repo_id	int	ID of particular native repository
url	string	The URL of the requested repository
is_created	boolean	Defines wether all background work related to creating native repository is finished.
is_ready	boolean	Defines whether native repository has been ever re-indexed or it is external one.

pem.packaging.native_repository.reindex

This method is supported by PEM starting from version 2.6.

This method re-creates package manager's specific database within the repository. In case of YUM, this method causes the calling of "yum-arch" or "createrepo" utility depending on YUM version installed on host. One must call this method (or its UI analog) each time when the actual content of repository is changed.

Name	Type	Short Description
repo_id	int	ID of particular native repository

This method has no output parameters.

pem.packaging.native_repository.removeRepository

This method is supported by PEM starting from version 2.6.

This method causes the removal of native repository.

Name	Type	Short Description
repo_id	int	ID of particular native repository

This method has no output parameters.

Branding Management

pem.brandDomain

This method is supported by PEM starting from version 2.6.

This method allows to brand an existing domain.

Name	Type	Short Description
domain_name	string	Name of the existing domain with apache server hosting to create branding for.
[brand_name]	string	Name of the brand to be created. If not issued, brand name will be the same as the name of the domain.
[path_prefix]	string	This prefix will be added to the branded URL path. For example, http://mydomain.com/path_prefix/ . If not issued, no prefix will be added.
[logo_url]	string	The URL of the logo to be inserted to the control panel. If not issued, the default logo will be used.
[skin]	string	Name of the skin to be installed to control panels. If not issued, the default skin will be used.
[access_type]	string	Access type to the branded domain. "s" stands for ssl access, "h" - non-ssl, "b" will provide both access types.
[skin_editable]	boolean	The parameter determines whether the skin can be customized. If not issued, skin editing will not be allowed to reseller.
[has_bm]	boolean	The parameter determines whether Billing Manager is installed.
[shop_id]	int	The ID of the reseller's online shop. If not issued, the shop will not be created.

This method has no output parameters.

pem.unbrandDomain

This method is supported by PEM starting from version 2.6.

This method removes the branding from the existing domain.

Name	Type	Short Description
domain_name	string	Name of the existing domain to remove the branding from.

Plesk Management

pem.installPleskLicense

This method is supported by PEM starting from version 2.6.

This method allows to install Plesk license.

Name	Type	Short Description
subscription_id	int	The ID of the subscription, which the Plesk in VPS is provisioned under.
license	base64	The content of the Plesk license.

This method has no output parameters.

pem.revokePleskLicense

This method is supported by PEM starting from version 2.6.

This method allows to revoke Plesk license.

Name	Type	Short Description
subscription_id	int	The ID of the subscription, which the Plesk in VPS is provisioned under.

This method has no output parameters.

Transactional Extensions

pem.batchRequest

This method is supported by PEM starting from version 2.3.

This method executes the set of specified Open API methods in a single PEM transaction. This method is introduced for external system that does not have the own transaction mechanism or mechanism which handles PEM interaction statuses. The methods go through XML-RPC “params” array and for each parameter execute corresponding operation which described in value. All operation are executed in a single PEM transaction. Requested operations are executed in order they are specified. On first operation failure PEM rollbacks transaction and returns the result of the first failed operation.

Each XML-RPC “param” has the following input parameters:

Name	Type	Short Description
operation	string	Indicates name of OpenAPI method that should be executed.
parameters	array	Parameters that should be passed to requested method as for usual XML-RPC call.

OR

[request_id]	string	the identifier of request, to be further used by the pem.getRequestStatus() (on page 96)
--------------	--------	--

This method has no output parameters.

For the XML-RPC sample of this method refer to the pem.batchRequest sample topic (on page 109).

pem.getRequestStatus

This method is supported by PEM starting from version 2.6.2

This method allows to get the information on the result of any request previously issued to PEM via Open API.

The method has the following input parameters:

Name	Type	Short Description
request_id	string	the identifier of request (that was previously set by pem.batchRequest (on page 96))

Output parameters:

Name	Type	Short Description
<code>request_status</code>	int	The status of the request. 0 - succeeded 1 - running 2 - failed
<code>status_messages</code>	array of strings	The reason of failure.

CHAPTER 5

XML-RPC Samples

This section provides examples of using PEM's *Public API* for several most important methods. The samples are presented in terms of XML-RPC requests and responses.

Important: Use only UTF-8 encoding in your xml requests!

In This Chapter

Common Response Codes	99
pem.activateSubscription Sample	100
pem.addAccount Sample	103
pem.addAccountMember Sample	106
pem.addDomain Sample	109
pem.batchRequest Sample	109

Common Response Codes

Each XML-RPC call contains of a request and a response. Structures of requests depend on the particular method and described in corresponding sections below. As to responses, there can be two types of them: response indicating successful completion of the call, and response indicating failure. The following subsections depict the structure of these types of responses for XML-RPC methods constituting PEM's *Public API*.

Success Response

The structure of success response for a method depends on whether this method has output parameters or not. The following XML snippet shows the structure of success response for a method that does not have output parameters:

```
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>status</name>
            <value><int>0</int></value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

The following XML snippet shows the structure of success response for a method that has one or more output parameters:

```
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>status</name>
            <value><int>0</int></value>
          </member>
          <member>
            <name>result</name>
            <value>
              <struct>
                ...
              </struct>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

In this case the response contains additional member “result”, which has a value of “struct” type. The structure contains members, each representing a single output parameter for the method. The exact type of each output parameter depends on the particular method and is described in corresponding sections below in this document.

Failure Response

The structure of failure response for a method has the same form independently on whether this method has output parameters or not.

```
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>status</name>
            <value><int>-1</int></value>
          </member>
          <member>
            <name>module_id</name>
            <value>${errorDomain}</value>
          </member>
          <member>
            <name>extype_id</name>
            <value><int>${exceptionType}</int></value>
          </member>
          <member>
            <name>error_message</name>
            <value>${theErrorMessage}</value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

Negative status field indicates fault response. Fields `module_id`, `extype_id` identify actual error condition:

`module_id` - identifies some logical domain an error condition belongs to;

`extype_id` - uniquely identifies concrete error condition within its domain.

Field `error_message` contains descriptive message for specific error condition. This message is in English only and can not be localized.

Deprecated field `error_code` contains old-style error condition code (it is not the same as `extype_id`). This field will be removed from fault responses in PEM 2.6.

pem.activateSubscription Sample

The following XML snippet shows the structure of sample request for XML-RPC call for this method:

```

<methodCall>
  <methodName>pem.activateSubscription</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>subscription_id</name>
            <value><int>${theSubscriptionID}</int></value>
          </member>
          <member>
            <name>service_template_id</name>
            <value><int>${theServiceTemplateID}</int></value>
          </member>
          <member>
            <name>account_id</name>
            <value><int>${theAccountID}</int></value>
          </member>
          <member>
            <name>resource_limits</name>
            <value>
              <array>
                <data>
                  <value>
                    <struct>
                      <member>
                        <name>resource_id</name>
                        <value>${rt_id1}</value>
                      </member>
                      <member>
                        <name>resource_limit</name>
                        <value>${Limit_1}</value>
                      </member>
                    </struct>
                  </value>
                  ...
                  <value>
                    <struct>
                      <member>
                        <name>resource_id</name>
                        <value>${rt_idN}</value>
                      </member>
                      <member>
                        <name>var_value</name>
                        <value>${Limit_N}</value>
                      </member>
                    </struct>
                  </value>
                </data>
              </array>
            </value>
          </member>
          <member>
            <name>parameters</name>
            <value>

```

```

<array>
  <data>
    <value>
      <struct>
        <member>
          <name>var_name</name>
          <value>${Name_1}</value>
        </member>
        <member>
          <name>var_value</name>
          <value>${Value_1}</value>
        </member>
      </struct>
    </value>
    ...
    <value>
      <struct>
        <member>
          <name>var_name</name>
          <value>${Name_N}</value>
        </member>
        <member>
          <name>var_value</name>
          <value>${Value_N}</value>
        </member>
      </struct>
    </value>
  </data>
</array>
</value>
</member>
<member>
  <name>provisioned_resource_types</name>
  <value>
    <array>
      <data>
        <value>
          <struct>
            <member>
              <name>resource_type_id</name>
              <value>
                <int>${theResourceTypeID_1}</int>
              </value>
            </member>
            <member>
              <name>amount</name>
              <value><int>${theAmount_1}</int></value>
            </member>
          </struct>
        </value>
        <value>
          <struct>
            <member>
              <name>resource_type_id</name>
              <value>
                <int>${theResourceTypeID_2}</int>
              </value>
            </member>
          </struct>
        </value>
      </data>
    </array>
  </value>
</member>

```

```

        <name>amount</name>
        <value><int>${theAmount_2}</int></value>
    </member>
</struct>
</value>
...
<value>
    <struct>
        <member>
            <name>resource_type_id</name>
            <value>
                <int>${theResourceTypeID_N}</int>
            </value>
        </member>
        <member>
            <name>amount</name>
            <value><int>${theAmount_N}</int></value>
        </member>
    </struct>
</value>
</data>
</array>
</value>
</member>
</struct>
</value>
</param>
</params>
</methodCall>

```

pem.addAccount Sample

The following XML snippet shows the structure of sample request for XML-RPC call for this method:

```

<methodCall>
  <methodName>pem.addAccount</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>account_id</name>
            <value><int>${account_id}</int></value>
          </member>
          <member>
            <name>account_type</name>
            <value>${account_type}</value>
          </member>
          <member>
            <name>parent_account_id</name>
            <value><int>${parent_account_id}</int></value>
          </member>
          <member>
            <name>external_info</name>
            <value>

```

```
<struct>
  <member>
    <name>system_id</name>
    <value><string>${system_id}</string></value>
  </member>
  <member>
    <name>external_account_id</name>
    <value><string>${external_account_id}</string></value>
  </member>
</struct>
</value>
</member>
<member>
  <name>person</name>
  <value>
    <struct>
      <member>
        <name>title</name>
        <value>${title}</value>
      </member>
      <member>
        <name>first_name</name>
        <value>${first_name}</value>
      </member>
      <member>
        <name>middle_name</name>
        <value>${middle_name}</value>
      </member>
      <member>
        <name>last_name</name>
        <value>${last_name}</value>
      </member>
      <member>
        <name>company_name</name>
        <value>${company_name}</value>
      </member>
    </struct>
  </value>
</member>
<member>
  <name>address</name>
  <value>
    <struct>
      <member>
        <name>street_name</name>
        <value>${street_name}</value>
      </member>
      <member>
        <name>house_num</name>
        <value>${house_num}</value>
      </member>
      <member>
        <name>address2</name>
        <value>${address2}</value>
      </member>
      <member>
        <name>zipcode</name>
        <value>${zipcode}</value>
      </member>
    </struct>
  </value>
</member>
```

```
</member>
<member>
  <name>city</name>
  <value>${city}</value>
</member>
<member>
  <name>country</name>
  <value>${country}</value>
</member>
<member>
  <name>state</name>
  <value>${state}</value>
</member>
</struct>
</value>
</member>
<member>
  <name>phone</name>
  <value>
    <struct>
      <member>
        <name>country_code</name>
        <value>${p_country_code}</value>
      </member>
      <member>
        <name>area_code</name>
        <value>${p_area_code}</value>
      </member>
      <member>
        <name>phone_num</name>
        <value>${p_phone_num}</value>
      </member>
      <member>
        <name>ext_num</name>
        <value>${p_ext_num}</value>
      </member>
    </struct>
  </value>
</member>
<member>
  <name>fax</name>
  <value>
    <struct>
      <member>
        <name>country_code</name>
        <value>${f_country_code}</value>
      </member>
      <member>
        <name>area_code</name>
        <value>${f_area_code}</value>
      </member>
      <member>
        <name>phone_num</name>
        <value>${f_phone_num}</value>
      </member>
      <member>
        <name>ext_num</name>
        <value>${f_ext_num}</value>
      </member>
    </struct>
  </value>
</member>
```

```

        </member>
      </struct>
    </value>
  </member>
  <member>
    <name>email</name>
    <value>${email}</value>
  </member>
</struct>
</value>
</param>
</params>
</methodCall>

```

pem.addAccountMember Sample

The following XML snippet shows the structure of sample request for XML-RPC call for this method:

```

<methodCall>
  <methodName>pem.addAccountMember</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>account_id</name>
            <value><int>${theAccountID}</int></value>
          </member>
          <member>
            <name>user_id</name>
            <value><int>${theMemberID}</int></value>
          </member>
          <member>
            <name>login</name>
            <value>${theLoginName}</value>
          </member>
          <member>
            <name>auth</name>
            <value>
              <struct>
                <member>
                  <name>login</name>
                  <value>${theLoginName}</value>
                </member>
                <member>
                  <name>password</name>
                  <value>${thePassword}</value>
                </member>
              </struct>
            </value>
          </member>
          <member>
            <name>person</name>
            <value>
              <struct>

```

```
<member>
  <name>title</name>
  <value>${theTitle}</value>
</member>
<member>
  <name>first_name</name>
  <value>${theFirstName}</value>
</member>
<member>
  <name>middle_name</name>
  <value>${theMiddleName}</value>
</member>
<member>
  <name>last_name</name>
  <value>${theLastName}</value>
</member>
<member>
  <name>company_name</name>
  <value>${theCompanyName}</value>
</member>
</struct>
</value>
</member>
<member>
  <name>address</name>
  <value>
    <struct>
      <member>
        <name>street_name</name>
        <value>${theStreetName}</value>
      </member>
      <member>
        <name>house_num</name>
        <value>${theHouseNumber}</value>
      </member>
      <member>
        <name>address2</name>
        <value>${theSecondAddress}</value>
      </member>
      <member>
        <name>zipcode</name>
        <value>${thePostalCode}</value>
      </member>
      <member>
        <name>city</name>
        <value>${theCityName}</value>
      </member>
      <member>
        <name>country</name>
        <value>${theCountryCode}</value>
      </member>
      <member>
        <name>state</name>
        <value>${theStateOrProvinceName}</value>
      </member>
    </struct>
  </value>
</member>
```

```
<member>
  <name>phone</name>
  <value>
    <struct>
      <member>
        <name>country_code</name>
        <value>${theCountryCode}</value>
      </member>
      <member>
        <name>area_code</name>
        <value>${theAreaCode}</value>
      </member>
      <member>
        <name>phone_num</name>
        <value>${thePhoneNumber}</value>
      </member>
      <member>
        <name>ext_num</name>
        <value>${theExtensionNumber}</value>
      </member>
    </struct>
  </value>
</member>
<member>
  <name>fax</name>
  <value>
    <struct>
      <member>
        <name>country_code</name>
        <value>${theCountryCode}</value>
      </member>
      <member>
        <name>area_code</name>
        <value>${theAreaCode}</value>
      </member>
      <member>
        <name>phone_num</name>
        <value>${theFaxNumber}</value>
      </member>
      <member>
        <name>ext_num</name>
        <value>${theExtensionNumber}</value>
      </member>
    </struct>
  </value>
</member>
<member>
  <name>email</name>
  <value>${theEmailAddress}</value>
</member>
</struct>
</value>
</param>
</params>
</methodCall>
```

pem.addDomain Sample

The following XML snippet shows the structure of sample request for XML-RPC call for this method:

```
<methodCall>
  <methodName>pem.addDomain</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>subscription_id</name>
            <value><int>${theSubscriptionID}</int></value>
          </member>
          <member>
            <name>domain_name</name>
            <value><string>${theDomainName}</string></value>
          </member>
          <member>
            <name>registrar_status</name>
            <value><int>${theRegistrarStatus}</int></value>
          </member>
          <member>
            <name>hosting_type</name>
            <value><int>${theHostingType}</int></value>
          </member>
          <member>
            <name>path</name>
            <value><string>${thePath}</string></value>
          </member>
          <member>
            <name>tld_name</name>
            <value><string>${theTLDName}</string></value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

pem.batchRequest Sample

The following XML snippet shows the structure of sample request for XML-RPC call for this method. It performs an account creation through pem.addAccount method (on page 24), then add user to the account by pem.addAccountMember (on page 27) and then provides a subscription for the account by pem.activateSubscription method in one XML-RPC request i.e. one PEM transaction.

```
<?xml version="1.0" ?>
  <methodCall>
    <methodName>pem.batchRequest</methodName>
    <params>
```

```
<param>
  <value>
    <struct>
      <member>
        <name>operation</name>
        <value>pem.addAccount</value>
      </member>
      <member>
        <name>parameters</name>
        <value>
          <array>
            <data>
              <value>
                <struct>
                  <member>
                    <name>account_id</name>
                    <value>
                      <int>${account_id_here}</int>
                    </value>
                  </member>
                  <member>
                    <name>account_type</name>
                    <value>C</value>
                  </member>
                  <member>
                    <name>parent_account_id</name>
                    <value>
                      <int>1</int>
                    </value>
                  </member>
                  <member>
                    <name>person</name>
                    <value>
                      <struct>
                        <member>
                          <name>title</name>
                          <value>
                            <string>Mr.</string>
                          </value>
                        </member>
                        <member>
                          <name>first_name</name>
                          <value>
                            <string>${first_name_here}</string>
                          </value>
                        </member>
                        <member>
                          <name>middle_name</name>
                          <value>
                            <string>${middle_name_here}</string>
                          </value>
                        </member>
                        <member>
                          <name>last_name</name>
                          <value>
                            <string>${last_name_here}</string>
                          </value>
                        </member>
                      </struct>
                    </value>
                  </member>
                </struct>
              </value>
            </data>
          </array>
        </value>
      </member>
    </struct>
  </value>
</param>
```

```
        <name>company_name</name>
        <value>${company_name_here}</value>
    </member>
</struct>
</value>
</member>
<member>
    <name>address</name>
    <value>
        <struct>
            <member>
                <name>street_name</name>
                <value>
                    <string>${street_name_here}</string>
                </value>
            </member>
            <member>
                <name>house_num</name>
                <value>
                    <string>${house_num_here}</string>
                </value>
            </member>
            <member>
                <name>address2</name>
                <value>
                    <string>${second_address_here}</string>
                </value>
            </member>
            <member>
                <name>zipcode</name>
                <value>
                    <string>${zipcore_here}</string>
                </value>
            </member>
            <member>
                <name>city</name>
                <value>
                    <string>${city_here}</string>
                </value>
            </member>
            <member>
                <name>country</name>
                <value>
                    <string>${two_char_code_here}</string>
                </value>
            </member>
            <member>
                <name>state</name>
                <value>
                    <string>${state_here}</string>
                </value>
            </member>
        </struct>
    </value>
</member>
<member>
    <name>phone</name>
    <value>
        <struct>
```

```
<member>
  <name>country_code</name>
  <value>
    <string>${country_code_here}</string>
  </value>
</member>
<member>
  <name>area_code</name>
  <value>
    <string>${area_core_here}</string>
  </value>
</member>
<member>
  <name>phone_num</name>
  <value>
    <string>${phone_num_here}</string>
  </value>
</member>
<member>
  <name>ext_num</name>
  <value>
    <string>${ext_num_here}</string>
  </value>
</member>
</struct>
</value>
</member>
<member>
  <name>fax</name>
  <value>
    <struct>
      <member>
        <name>country_code</name>
        <value>
          <string>${country_code_here}</string>
        </value>
      </member>
      <member>
        <name>area_code</name>
        <value>
          <string>${area_code_here}</string>
        </value>
      </member>
      <member>
        <name>phone_num</name>
        <value>
          <string>${phone_num_here}</string>
        </value>
      </member>
      <member>
        <name>ext_num</name>
        <value>
          <string>${ext_num_here}</string>
        </value>
      </member>
    </struct>
  </value>
</member>
<member>
```



```
        <value>
          <string>Mr.</string>
        </value>
      </member>
    <member>
      <name>first_name</name>
      <value>
        <string>${first_name_here}</string>
      </value>
    </member>
    <member>
      <name>middle_name</name>
      <value>
        <string>${middle_name_here}</string>
      </value>
    </member>
    <member>
      <name>last_name</name>
      <value>
        <string>${last_name_here}</string>
      </value>
    </member>
    <member>
      <name>company_name</name>
      <value>
        <string>${company_name_here}</string>
      </value>
    </member>
  </struct>
</value>
</member>
<member>
  <name>address</name>
  <value>
    <struct>
      <member>
        <name>street_name</name>
        <value>
          <string>${street_name_here}</string>
        </value>
      </member>
      <member>
        <name>house_num</name>
        <value>
          <string>${house_num_here}</string>
        </value>
      </member>
      <member>
        <name>address2</name>
        <value>
          <string>${second_address_here}</string>
        </value>
      </member>
      <member>
        <name>zipcode</name>
        <value>
          <string>${zipcode_here}</string>
        </value>
      </member>
    </struct>
  </value>
</member>
```

```
<member>
  <name>city</name>
  <value>
    <string>${city_here}</string>
  </value>
</member>
<member>
  <name>country</name>
  <value>
    <string>${two_char_code_here}</string>
  </value>
</member>
<member>
  <name>state</name>
  <value>
    <string>${state_here}</string>
  </value>
</member>
</struct>
</value>
</member>
<member>
  <name>phone</name>
  <value>
    <struct>
      <member>
        <name>country_code</name>
        <value>
          <string>${country_code_here}</string>
        </value>
      </member>
      <member>
        <name>area_code</name>
        <value>
          <string>${area_code_here}</string>
        </value>
      </member>
      <member>
        <name>phone_num</name>
        <value>
          <string>${phone_num_here}</string>
        </value>
      </member>
      <member>
        <name>ext_num</name>
        <value>
          <string>${ext_num_here}</string>
        </value>
      </member>
    </struct>
  </value>
</member>
<member>
  <name>fax</name>
  <value>
    <struct>
      <member>
        <name>country_code</name>
        <value>
```

```

        <string>${country_code_here}</string>
    </value>
</member>
<member>
    <name>area_code</name>
    <value>
        <string>${area_code_here}</string>
    </value>
</member>
<member>
    <name>phone_num</name>
    <value>
        <string>${phone_num_here}</string>
    </value>
</member>
<member>
    <name>ext_num</name>
    <value>
        <string>${ext_num_here}</string>
    </value>
</member>
</struct>
</value>
</member>
<member>
    <name>email</name>
    <value>
        <string>${email_here}</string>
    </value>
</member>
</struct>
</value>
</data>
</array>
</value>
</member>
</struct>
</value>
</param>
<param>
    <value>
        <struct>
            <member>
                <name>operation</name>
                <value>pem.activateSubscription</value>
            </member>
            <member>
                <name>parameters</name>
                <value>
                    <array>
                        <data>
                            <value>
                                <struct>
                                    <member>
                                        <name>account_id</name>
                                        <value>
                                            <int>${account_id_here}</int>
                                        </value>
                                    </member>
                                </struct>
                            </value>
                        </data>
                    </array>
                </value>
            </member>
        </struct>
    </value>
</param>

```

```
<member>
  <name>service_template_id</name>
  <value>
    <int>${service_template_id_here}</int>
  </value>
</member>
<member>
  <name>subscription_id</name>
  <value>
    <int>${subscription_id_here}</int>
  </value>
</member>
<member>
  <name>parameters</name>
  <value>
    <array>
      <data>
        <value>
          <struct>
            <member>
              <name>var_name</name>
              <value>domain_name</value>
            </member>
            <member>
              <name>var_value</name>
              <value>${domain_name_here}</value>
            </member>
          </struct>
        </value>
        <value>
          <struct>
            <member>
              <name>var_name</name>
              <value>domain_id</value>
            </member>
            <member>
              <name>var_value</name>
              <value>${domain_id_here}</value>
            </member>
          </struct>
        </value>
        <value>
          <struct>
            <member>
              <name>var_name</name>
              <value>registrar_status</value>
            </member>
            <member>
              <name>var_value</name>
              <value>${registrar_status_here}</value>
            </member>
          </struct>
        </value>
      </data>
    </array>
  </value>
</member>
</struct>
</value>
```

```
        </data>  
    </array>  
  </value>  
</member>  
</struct>  
</value>  
</param>  
</params>  
</methodCall>
```

CHAPTER 6

Diagrams

This section contains diagrams explaining complex scenarios involving several methods of PEM's Public API.

In This Chapter

Obsolete Ordering a Subscription Model 119

Obsolete Ordering a Subscription Model

The following diagram shows the sequence of calls, which external billing system shall perform to add a shared hosting subscription and provision it in PEM:

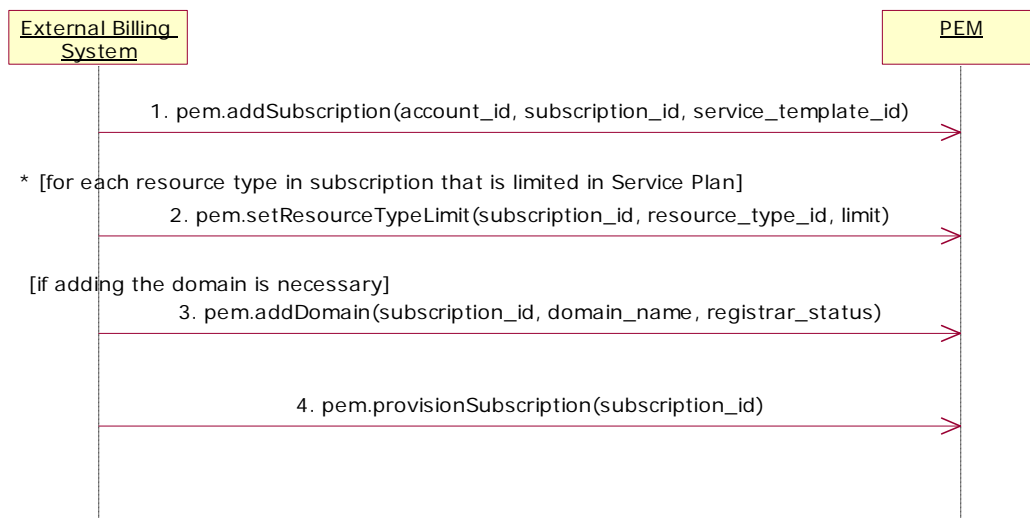


Figure 1: Ordering a Subscription Model

Index

A

Applications Management • 17, 89

B

Branding Management • 18, 94

C

Common Response Codes • 99

D

Database management • 16

Database Management • 78

Definitions, Acronyms, and Abbreviations • 10

Diagrams • 119

Domains Management • 14, 52

F

Failure Response • 100

Feedback • 7

I

Introduction • 8

M

Mail management • 16

Mail Management • 71

**Management of Accounts and
Account's Staff Members** • 12, 24

N

Native Package Management • 18, 92

O

**Obsolete Ordering a Subscription
Model** • 119

Overall Description • 11

Overview • 10

P

Parameters Details • 26, 29, 31, 34, 35, 36, 37,
38, 39, 41, 43, 44, 47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 59, 60, 61, 62, 63, 70, 71, 72,
79, 80, 84, 86, 87, 90, 91, 92

pem.activateSubscription • 42

pem.activateSubscription Sample • 100

pem.addAccount • 24

pem.addAccount Sample • 103

pem.addAccountMember • 27

pem.addAccountMember Sample • 106

pem.addDomain • 52

pem.addDomain Sample • 109

pem.addSubdomain • 54

pem.addSubscription • 43

pem.addWebSpaceBackup • 69

pem.batchRequest • 96

pem.batchRequest Sample • 109

pem.brandDomain • 94

pem.checkPassword • 91

pem.cqmail.addMailbox • 71

pem.cqmail.addMailForwarding • 73

pem.cqmail.getAutoresponderInfo • 73

pem.cqmail.getAutoresponderList • 74

pem.cqmail.getForwardingsList • 74

pem.cqmail.getMailboxInfo • 75

pem.cqmail.getMailHostingInfo • 75

pem.cqmail.getMailListInfo • 77

pem.cqmail.getMailListOwners,

 pem.cqmail.getMailListModerators,

 pem.cqmail.getMailListMembers • 77

pem.cqmail.getMailLists • 76

pem.cqmail.getMailnameList • 78

pem.createDatabase • 78

pem.createDatabaseUser • 79

pem.disableAccount • 30

pem.disableDomain • 55

pem.disableSubscription • 46

pem.doLogin • 31

pem.enableAccount • 33

pem.enableDomain • 55

pem.enableSubscription • 47

pem.getAccountInfo • 31

pem.getAccountMemberByLogin • 32

pem.getAccountMemberInfo • 32

pem.getAccountSubscriptions • 33

pem.getDatabaseAccessHostList • 80

pem.getDatabaseInfo • 81

pem.getDatabaseList • 81

pem.getDatabaseUserList • 82

pem.getDomainInfo • 56

pem.getDomainList • 57

pem.getDomainSubscription • 58

pem.getFTPUser • 64

pem.getNameServers • 59
 pem.getRequestStatus • 96
 pem.getResourceUsage • 83
 pem.getResourceUsageForPeriod • 84
 pem.getServiceTemplate • 44
 pem.getSubscription • 45
 pem.getWebspacesList • 58
 pem.importCertificate • 59
 pem.installPleskLicense • 95
 pem.installWebApplication • 89
 pem.migrateSubscription • 47
 pem.orderSubscription (deprecated) • 48
 pem.packaging.native_repository.createRepository • 92
 pem.packaging.native_repository.getRepository • 93
 pem.packaging.native_repository.reindex • 93
 pem.packaging.native_repository.removeRepository • 93
 pem.provisionSubscription (deprecated) • 49
 pem.removeAccount • 34
 pem.removeAccountMember • 34
 pem.removeDomain • 60
 pem.removeSubdomain • 60
 pem.removeSubscription • 50
 pem.removeWebSpaceBackup • 90
 pem.resetResourceUsage • 87
 pem.restoreWebSpaceFromBackup • 90
 pem.revokePleskLicense • 95
 pem.setAccountAuthData • 35
 pem.setAccountInfo • 36
 pem.setAccountStatus • 38
 pem.setDomainHostingType • 61
 pem.setDomainRegistrarStatus • 62
 pem.setFrontPageParameters • 70
 pem.setFTPPassword • 64
 pem.setMemberInfo • 38
 pem.setMemberPassword • 40
 pem.setResourceTypeLimit • 87
 pem.setResourceTypeLimits • 88
 pem.setSubdomainsHostingType • 63
 pem.setSubscriptionName • 51
 pem.unbrandDomain • 95
 pem.uninstallWebApplication • 91
 pem.updateAccountAndAccountMember • 40
 pem.upgradeSubscription (deprecated) • 51
 pem.web.getFileManagerInfo • 65
 pem.web.getFrontPageInfo • 66
 pem.web.getFTPConf • 66
 pem.web.getHttpErrorDocs • 67
 pem.web.getMimeTypes • 67
 pem.web.getSiteProps • 68
 pem.web.getSSLInfo • 69
 Plesk Management • 18, 95

Preface • 6

Provisioning Models • 42

Public API Reference • 19

Purpose • 9

R

References • 9

Resource Accounting • 17, 83

S

Scope • 9

Security Management • 18, 91

Subscription Management • 13

Subscriptions Management • 42

Success Response • 99

T

Transactional Extensions • 18, 96

Typographical Conventions • 6

W

WebSpace Management • 15, 64

X

XML-RPC Samples • 98