Zend Server API Documentation

Release 0.0.2

Ingo Walz

November 12, 2016

Contents

The official Zend Server API documentation

CHAPTER 1

Installation

1.1 Via Composer

You can install the API with composer by adding the following lines to your `composer.json` file:

```
{
    "repositories": [
        {
            "type": "composer",
            "url": "http://packages.zendframework.com/"
        }
    ],
    "require": {
        "zendserverapi/zendserverapi": "dev-master"
    },
    "minimum-stability": "dev"
}
```

composer install
: Run the installation and you're ready to go.

Note: There are a few dependencies to the Zend Framework 2 from inside the library. The `repositories` section will allow you to install only the required components and not the whole framework. If you don't register the Zend packagist repository, the whole Zend Framework 2 will be installed by composer. I recommend to register that repository to reduce the overhead during installation.

CHAPTER 2

Configuration

This chapter explains the configuration possibilities of the Zend Server API ZendService.

2.1 Configuration File

The API is lookig for a configfile at `vendor/zendserverapi/zendserverapi/config/config.php` by default. This file is not included in the project, this will allow you to keep your project specific configuration on updates. You can find such a skeleton at `vendor/zendserverapi/zendserverapi/config/config.dist.php`:

```php
<?php

return array(
    "servers" => array (
        # Contains a valid default config
        "general" => array(
            "version" => \ZendService\ZendServerAPI\Version::ZS56,
            "apiName" => "",
            "fullApiKey" => "",
            "readApiKey" => "",
            "host" => "localhost",
            "port" => "10081"
        )
    ),
    "settings" => array (
        'loglevel' => \Zend\Log\Logger::DEBUG
    )
);
```

You can change the location of the configuration file by 2 different ways. Either statically, or on the BaseAPI object level.

```php
<?php
\ZendService\ZendServerAPI\PluginManager::setConfigFile(__DIR__.'/_files/config/config.php');
```

```php
<?php
$server = new \ZendService\ZendServerAPI\Server();
$server->setConfig(__DIR__.'/_files/config/config.php');
```

2.2 The Server configuration

This configuration file allows you to add as many (and different) Zend Servers as you want.

The `general` section in the `servers` key is the default server. Every BaseAPI implementation (e.g. Monitor, Server, Administration, ...) receives a name with the first constructor parameter. If none parameter is provided, the general section will be used.

```
<?php

    return array(
        "servers" => array (
            # Contains a valid default config
            "general" => array(
                "version" => \ZendService\ZendServerAPI\Version::ZSCM56,
                "apiName" => "admin",
                "fullApiKey" => "bee698dde6a95de71932d65cb655c31fc4ea04c1fabaf6f0a1b852617eac32ac",
                "readApiKey" => "",
                "host" => "10.0.0.1",
                "port" => "10083",
                "protocol" => "https"
            ),
            "local" => array(
                "version" => \ZendService\ZendServerAPI\Version::ZS56,
                "apiName" => "admin",
                "readApiKey" => "9dc7f8c5ac43bb2ab36120861b4aeda8f9bb6c521e124360fd5821ef279fd9c7",
                "host" => "localhost",
                "port" => "10081"
            )
        ),
        "settings" => array (
            'loglevel' => \Zend\Log\Logger::DEBUG
        )
    );
```

This configuration will give you the possibility, to connect to a remote Zend Server 5.6 Cluster Manager on port 10083 via https that way:

```
<?php
$server = new \ZendService\ZendServerAPI\Server();
$server->getSystemInfo();
```

And to do the same locally this way:

```
<?php
$server = new \ZendService\ZendServerAPI\Server('local');
$server->getSystemInfo();
```

Note: If you use port 10082, https will be selected automatically. If you want to use http and port 10082, you've to set it explicit in the config. On every other port, http will be selected by default.

2.3 Configuration settings

The settings allow you to set a proxy and to change the loglevel that is used for every request. I suggest to use `\Zend\Log\Logger::DEBUG` for development and `\Zend\Log\Logger::INFO` for production (for auditing

purposes). `\Zend\Log\Logger::DEBUG` will give you the plain HTTP requests and the responses. This is causing a huge amount of data per request. `\Zend\Log\Logger::INFO` will simply tell you, which request was performt on which server and when.

The proxy setting looks like this:

```
<?php

return array(
    "servers" => array (
        # Contains a valid default config
        "general" => array(
            "version" => \ZendService\ZendServerAPI\Version::ZS56,
                "apiName" => "",
                "fullApiKey" => "",
                "readApiKey" => "",
                "host" => "localhost",
                "port" => "10081"
        )
    ),
    "settings" => array (
        'loglevel' => \Zend\Log\Logger::DEBUG,
        'proxyHost' => 'http://internal.proxy',
        'proxyPort' => 8010
    )
);
```

Note: If you don't specify a proxy port, 8080 will be used by default.

CHAPTER 3

API Versioning

3.1 Configure the version

Reference: Zend Server 5.6 Webapi Versioning

The version needs to be specified in the *Configuration File* under the `version` key. The Web API was introduced in Zend Server 5.1 with the Web API Version 1.0. With Zend Server 5.5 the API Version 1.1 has been released. 1.2 came with Zend Server 5.6 and 1.3 with Zend Server 6. For now, the API is fully functional up to Zend Server 5.6 - Zend Server 6 implementation is in progress.

The following class constants are available:

```
<?php
namespace ZendService\ZendServerAPI;

class Version
{
    /**
     * Zend Server Cluster Manager 5.1
     * API Version 1.0
     * @var string
     */
    const ZSCM51 = "1.0";
    /**
     * Zend Server Cluster Manager 5.5
     * API Version 1.1
     * @var string
     */
    const ZSCM55 = "1.1";
    /**
     * Zend Server Cluster Manager 5.6
     * API Version 1.2
     * @var string
     */
    const ZSCM56 = "1.2";
    /**
     * Zend Server 5.1
     * API Version 1.0
     * @var string
     */
    const ZS51 = "1.0";
    /**
     * Zend Server 5.5
```

```
     * API Version 1.1
     * @var string
     */
    const ZS55 = "1.1";
    /**
     * Zend Server 5.6
     * API Version 1.2
     * @var string
     */
    const ZS56 = "1.2";
}
```

This is configured on the Server level and is the basis for the web api factories. Every API method is fetched from specific version factories. If you specifiy e.g. ZS55 on a Zend Server 5.6, you're not able to perform Web API 1.2 actions! So take care, that you configure the correct API Version for your servers.

3.2 Web API Version methods

Web API 1.0

- clusterGetServerStatus
- clusterAddServer
- clusterRemoveServer
- clusterEnableServer
- clusterDisableServer
- restartPHP
- *The getSystemInfo Method*
- configurationImport
- configurationExport

Web API 1.1

- clusterReconfigureServer
- applicationGetStatus
- applicationDeploy
- applicationRemove
- applicationRollback
- applicationSynchronize
- applicationUpdate

Web API 1.2

- codetracingDisable
- codetracingEnable
- codetracingIsEnabled
- codetracingCreate

- codetracingDelete
- codetracingList
- codetracingDownloadTraceFile
- monitorGetRequestSummary
- monitorGetIssuesListByPredefinedFilter
- monitorGetIssuesDetails
- monitorGetEventGroupDetails
- monitorChangeIssueStatus
- monitorExportIssueByEventsGroup
- studioStartDebug
- studioStartProfile

CHAPTER 4

Server and Cluster Management methods

The following is a list and documentation of the available methods used to manage your server and/or cluster.

- getSystemInfo
- clusterGetServerStatus
- clusterAddServer
- clusterRemoveServer
- clusterDisableServer
- clusterEnableServer
- clusterReconfigureServer
- restartPHP

4.1 The getSystemInfo Method

Use this method to get information about the system, including the Zend Server edition and version, PHP version, licensing information, etc. This method produces similar output on all Zend Server systems, and is future compatible.

4.1.1 Method definition

```
<?php
public function getSystemInfo() { }
```

4.1.2 getSystemInfo information

Return value	\ZendService\ZendServerAPI\DataTypes\SystemInfo (SystemInfo api doc)
Online reference	getSystemInfo online reference
Available in Version	• 1.0 • 1.1 • 1.2 • 1.3

4.1.3 Example

```php
<?php
use ZendService\ZendServerAPI\Server;

$server = new Server();
$systemInfo = $server->getSystemInfo();
```

4.2 The clusterGetServerStatus Method

Use this method to get the list of servers in the cluster and the status of each one. On a Zend Server Cluster Manager with no valid license, this operation fails. This operation causes Zend Server Cluster Manager to check the status of servers and return fresh, non-cached information. This is different from the Servers List tab in the GUI, which may present cached information. Users interested in reducing load by caching this information should do it in their own code.

4.2.1 Method definition

```php
<?php
public function clusterGetServerStatus(array $parameters = array()) { }
```

Table 4.1: **Parameter**

Parameter	Data Type	Default value	Required	Description
$parameter	array	array()	no	A list of server IDs. If specified, the status is returned for these servers only. If not specified, the status of all the servers is returned.

4.2.2 clusterGetServerStatus information

Return value	\ZendService\ZendServerAPI\DataTypes\ServersList (ServersList api doc)
Online reference	clusterGetServerStatus online reference
Available in Version	• 1.0 • 1.1 • 1.2 • 1.3

4.2.3 Example

```php
<?php
use ZendService\ZendServerAPI\Server;

$server = new Server();
$serversList = $server->clusterGetServerStatus(array($serverId1, $serverId2));
```

```
foreach ($serversList as $server) {
    /** @var $server \ZendService\ZendServerAPI\DataTypes\ServersList */
    echo "Id: " . $server->getId() . PHP_EOL;
    echo "Name: " . $server->getName() . PHP_EOL;
 }
```

4.3 The clusterAddServer Method

Add a new server to the cluster. On a Zend Server Cluster Manager with no valid license, this operation fails.

4.3.1 Method clusterAddServer definition

```
<?php
public function clusterAddServer($serverName, $serverUrl, $guiPassword, $propagateSettings = false)
```

Table 4.2: **Parameter**

Parameter	Data Type	Default value	Required	Description
$serverName	string		yes	The server name.
$serverUrl	string		yes	The server address as a full HTTP/HTTPS URL.
$guiPassword	string		yes	The server GUI password.
$propagateSettings	boolean	false	no	Propagate this server's current settings to the rest of the cluster.

4.3.2 clusterAddServer information

Return value:	\ZendService\ZendServerAPI\DataTypes\ServerInfo (ServerInfo api doc)
Online reference:	clusterAddServer online reference
Available in Version:	• 1.0 • 1.1 • 1.2 • 1.3

4.3.3 Example

```
<?php
use ZendService\ZendServerAPI\Server;

$server = new Server();
$serverName = "server10";
$serverUrl = "https://10.0.0.10:10082/ZendServer";
$guiPassword = "test";
$propagateSettings = false;
$serverInfo = $server->clusterAddServer($serverName, $serverUrl, $guiPassword, $propagateSettings);
```

4.4 The clusterRemoveServer Method

This method removes a server from the cluster. The removal process may be asynchronous if Session Clustering is used. If this is the case, the initial operation will return an HTTP 202 response. As long as the server is not fully removed, further calls to remove the same server should be idempotent. On a Zend Server Cluster Manager with no valid license, this operation fails.

4.4.1 Method clusterRemoveServer definition

```php
<?php
public function clusterRemoveServer($serverId, $force = false) { }
```

Table 4.3: **Parameter**

Parameter	Data Type	Default value	Required	Description
$serverId	int		yes	The id of the server to remove
$force	boolean	false	no	Force-remove the server, skipping graceful shutdown process.

4.4.2 clusterRemoveServer information

Return value:	\ZendService\ZendServerAPI\DataTypes\ServerInfo (ServerInfo api doc)
Online reference:	clusterRemoveServer online reference
Available in Version:	• 1.0 • 1.1 • 1.2 • 1.3

4.4.3 Example

```php
<?php
use ZendService\ZendServerAPI\Server;

$server = new Server();
$serversList = $server->clusterGetServerStatus();

// removes all server from the 'general' cluster
foreach ($serversList as $server) {
    $server->clusterRemoveServer($server->getId());
}
```

4.5 The clusterEnableServer Method

This method is used to re-enable a cluster member. This process may be asynchronous if Session Clustering is used. If this is the case, the initial operation will return an HTTP 202 response. This action is idempotent, and running it on

an enabled server will result in a 200 OK response with no consequences. On a Zend Server Cluster Manager with no valid license this operation fails.

4.5.1 Method clusterEnableServer definition

```php
<?php
public function clusterEnableServer($serverId)
```

Table 4.4: **Parameter**

Parameter	Data Type	Default value	Required	Description
$serverId	int		yes	The server id

4.5.2 clusterEnableServer information

Return value:	\ZendService\ZendServerAPI\DataTypes\ServerInfo (ServerInfo api doc)
Online reference:	clusterEnableServer online reference
Available in Version:	• 1.0 • 1.1 • 1.2 • 1.3

4.5.3 Example

```php
<?php
use ZendService\ZendServerAPI\Server;

$server = new Server();
$serversList = $server->clusterGetServerStatus();

// enables all server from the 'general' cluster
foreach ($serversList as $server) {
    $server->clusterEnableServer($server->getId());
}
```

4.6 The clusterDisableServer Method

This method disables a cluster member. This process may be asynchronous if Session Clustering is used. If this is the case, the initial operation returns an HTTP 202 response. As long as the server is not fully disabled, further calls to this method are idempotent. On a Zend Server Cluster Manager with no valid license, this operation fails.

4.6.1 Method clusterDisableServer definition

```php
<?php
public function clusterDisableServer($serverId)
```

Table 4.5: **Parameter**

Parameter	Data Type	Default value	Required	Description
$serverId	int		yes	The server id

4.6.2 clusterDisableServer information

Return value:	\ZendService\ZendServerAPI\DataTypes\ServerInfo (ServerInfo api doc)
Online reference:	clusterDisableServer online reference
Available in Version:	• 1.0 • 1.1 • 1.2 • 1.3

4.6.3 Example

```php
<?php
use ZendService\ZendServerAPI\Server;

$server = new Server();
$serversList = $server->clusterGetServerStatus();

// disables all server from the 'general' cluster
foreach ($serversList as $server) {
    $server->clusterDisableServer($server->getId());
}
```

4.7 The clusterReconfigureServer Method

Re-configure a cluster member to match the cluster's profile. This operation will fail on a Zend Server Cluster Manager with no valid license.

4.7.1 Method clusterReconfigureServer definition

```php
<?php
public function clusterReconfigureServer($serverId, $doRestart = false)
```

Table 4.6: **Parameter**

Parameter	Data Type	Default value	Required	Description
$serverId	int		yes	Specify if the re-configured server should be restarted after the re-configure action.
$doRestart	boolean	false	no	Sends the restart command to all servers at the same time

Note: Because of the Zend Deployment Deamon (zdd), since Zend Server 5.5, there is an implicit restart on this method anyways.

4.7.2 clusterReconfigureServer information

Return value:	\ZendService\ZendServerAPI\DataTypes\ServerInfo (ServerInfo api doc)
Online reference:	clusterReconfigureServer online reference
Available in Version:	• 1.1 • 1.2 • 1.3

4.7.3 Example

```php
<?php
use ZendService\ZendServerAPI\Server;

$server = new Server();
$serverInfo = $server->clusterGetServerStatus();

// reconfigure all server from the 'general' cluster
foreach ($serversList as $server) {
    $server->clusterReconfigureServer($server->getId());
}
```

4.8 The restartPhp Method

This method restarts PHP on all servers or on specified servers in the cluster. A 202 response in this case does not always indicate a successful restart of all servers. Use the clusterGetServerStatus command to check the server(s) status again after a few seconds.

4.8.1 Method restartPhp definition

```php
<?php
public function restartPhp($serverIds = array(), $parallelRestart = false) { }
```

Table 4.7: **Parameter**

Parameter	Data Type	Default value	Required	Description
$serverIds	array	array()	no	A list of server IDs to restart. If not specified, all servers in the cluster will be restarted. In a single Zend Server context this parameter is ignored.
$parallelRestart	boolean	false	no	Sends the restart command to all servers at the same time.

4.8.2 restartPhp information

Return value:	\ZendService\ZendServerAPI\DataTypes\ServersList (ServersList api doc)
Online reference:	restartPhp online reference
Available in Version:	• 1.0 • 1.1 • 1.2 • 1.3

4.8.3 Example

```php
<?php
use ZendService\ZendServerAPI\Server;

$server = new Server();
$server->restartPhp();
```

CHAPTER 5

Filter methods

Filter API actions provide external actors with ways to query and manipulate filters and their definitions.

- filterGetByType
- filterSave
- filtersDelete

5.1 The filterGetByType Method

Retrieve and display a list of filters.

5.1.1 Method filterGetByType definition

```
<?php
public function filterGetByType($type) { }
```

Table 5.1: **Parameter**

Parameter	Data Type	Default value	Required	Description
$type	string		yes	Type of a filter (issue,job)

5.1.2 filterGetByType information

Return value	\ZendService\ZendServerAPI\DataTypes\Filters (Filters api doc)
Online reference	filterGetByType online reference
Available in Version	• 1.3

5.1.3 Example

```
<?php
use ZendService\ZendServerAPI\Filter;

$server = new Filter();
```

```
$filters = $server->filterGetByType("issue");

foreach($filters as $filter) {
    echo $filter->getName() . PHP_EOL;
}
```

5.2 The filterSave Method

Save a filter.

5.2.1 Method filterSave definition

```
<?php
public function filterSave($type, $name, $id = null, $data = array()) { }
```

Table 5.2: **Parameter**

Parameter	Data Type	Default value	Required	Description
$type	string		yes	Type of a filter (issue,job)
$name	string		yes	Name of filter.
$id	int		no	ID of a filter.
$data	array	array()	no	Array of parameters to be saved.

5.2.2 filterSave information

Return value	\ZendService\ZendServerAPI\DataTypes\Filter (Filter api doc)
Online reference	filterSave online reference
Available in Version	• 1.3

5.2.3 Example

```
<?php
use ZendService\ZendServerAPI\Filter;

$server = new Filter();
$filter = $server->filterSave("issue", "foo", array("eventTypes" => array("function-slow-exec")));

echo $filter->getName() . " successfully added with id " . $filter->getId() . PHP_EOL;
```

5.3 The filterDelete Method

Deletes a filter.

5.3.1 Method filterDelete definition

```
<?php
public function filterDelete($name) { }
```

Table 5.3: **Parameter**

Parameter	Data Type	Default value	Required	Description
$name	string		yes	Name of filter.

5.3.2 filterDelete information

Return value	\ZendService\ZendServerAPI\DataTypes\Filter (Filter api doc)
Online reference	filterDelete online reference
Available in Version	• 1.3

5.3.3 Example

```
<?php
use ZendService\ZendServerAPI\Filter;

$server = new Filter();
$filter = $server->filterDelete("foo");

echo $filter->getName() . " successfully removed" . PHP_EOL;
```

CHAPTER 6

Audit methods

Save settings of audit history and triggers.

- auditGetList
- auditGetDetails
- auditSetSettings

6.1 The auditGetList Method

Get a list of audit entries.

6.1.1 Method auditGetList definition

```
<?php
public function auditGetList($limit = null, $offset = null, $order = null, $direction = null, $filte
```

Table 6.1: **Parameter**

Parameter	Data Type	Default value	Required	Description
$limit	int		no	The number of rows to retrieve. Default lists all audit entries up to an arbitrary limit set by the system
$offset	int	0	no	A paging offset to begin the list from. Default: 0
$order	string	audit_id	no	Column identifier for sorting the result set (audit_id, node_id, time).
$direction	string	DESC	no	Sorting direction: ASC or DESC.
$filters	array	array()	no	Add filter parameters in an ad-hoc manner. These filters will be added to the predefined filter that was passed. This parameter is an array with a predefined set of parameters that accept strings or arrays to hold multiple values: from: string, a timestamp to use for retrieving audit rows to: string, a timestamp to use for retrieving audit rows freeText: string auditTypes: array, a list of auditTypes- AUDIT_APPLICATION_DEPLOY, AUDIT_APPLICATION_REMOVE, AUDIT_APPLICATION_UPGRADE, AUDIT_APPLICATION_ROLLBACK, AUDIT_APPLICATION_REDEPLOY, AUDIT_APPLICATION_REDEPLOY_ALL, AUDIT_APPLICATION_DEFINE, AUDIT_DIRECTIVES_MODIFIED, AUDIT_EXTENSION_ENABLED, AUDIT_EXTENSION_DISABLED, AUDIT_RESTART_DAEMON, AUDIT_RESTART_PHP, AUDIT_GUI_AUTHENTICATION, AUDIT_GUI_CHANGE_PASSWORD, AUDIT_GUI_AUTHORIZATION, AUDIT_GUI_AUTHENTICATION_LOGOUT, AUDIT_GUI_AUDIT_SETTINGS_SAVE, AUDIT_GUI_BOOTSTRAP_CREATEDB, AUDIT_GUI_BOOTSTRAP_SAVELICENSE, AUDIT_SERVER_JOIN, AUDIT_SERVER_ADD, AUDIT_SERVER_ENABLE, AUDIT_SERVER_DISABLE, AUDIT_SERVER_REMOVE, AUDIT_SERVER_REMOVE_FORCE, AUDIT_SERVER_RENAME, AUDIT_SERVER_SETPASSWORD, AUDIT_CODETRACING_CREATE, AUDIT_CODETRACING_DELETE, AUDIT_CODETRACING_DEVELOPER_ENABLE, AUDIT_CODETRACING_DEVELOPER_DISABLE, AUDIT_JOBQUEUE_REQUEUE, AUDIT_JOBQUEUE_DELETE, AUDIT_MONITOR_RULES_ENABLE, AUDIT_MONITOR_RULES_DISABLE, AUDIT_MONITOR_RULES_SAVE, AUDIT_MONITOR_RULES_REMOVE, AUDIT_STUDIO_DEBUG, AUDIT_STUDIO_PROFILE, AUDIT_STUDIO_SOURCE, AUDIT_CLEAR_OPTIMIZER_PLUS_CACHE, AUDIT_CLEAR_DATA_CACHE_CACHE, AUDIT_CLEAR_PAGE_CACHE_CACHE, AUDIT_PAGE_CACHE_SAVE_RULE, AUDIT_PAGE_CACHE_DELETE_RULES, AUDIT_JOB_QUEUE_SAVE_RULE, AUDIT_JOB_QUEUE_DELETE_RULES, AUDIT_JOB_QUEUE_DELETE_JOBS, AUDIT_JOB_QUEUE_REQUEUE_JOBS, AUDIT_JOB_QUEUE_RESUME_RULES,AUDIT_JOB_QUEUE_DISABLE_RULES, AUDIT_JOB_QUEUE_RUN_NOW_RULE

6.1.2 auditGetList information

Return value	\ZendService\ZendServerAPI\DataTypes\AuditMessages (AuditMessages api doc)
Online reference	auditGetList online reference
Available in Version	• 1.3

6.1.3 Example

```
<?php
use ZendService\ZendServerAPI\Audit;

$audit = new Audit();
$auditMesssages = $audit->auditGetList();

foreach($auditMessages as $auditMessage) {
    echo $auditMessage->getAuditTypeTranslated() . " by " . $auditMessage->getUsername() . PHP_EOL;
}
```

6.2 The auditGetDetails Method

Get all details available on a particular audit item.

6.2.1 Method auditGetDetails definition

```
<?php
public function auditGetDetails($auditId) { }
```

Table 6.2: **Parameter**

Parameter	Data Type	Default value	Required	Description
$auditId	int		yes	Audit ID to get all details for

6.2.2 auditGetDetails information

Return value	\ZendService\ZendServerAPI\DataTypes\AuditMessageDetails (AuditMessageDetails api doc)
Online reference	auditGetDetails online reference
Available in Version	• 1.3

6.2.3 Example

```
<?php
use ZendService\ZendServerAPI\Audit;
```

```
$audit = new Audit();
$auditMesssages = $audit->auditGetList();

foreach($auditMessages as $auditMessage) {
    $details = $audit->auditGetDetails($auditMessage->getId());
    echo $details->getAuditProgress()->getServerName() . PHP_EOL
}
```

6.3 The auditSetSettings Method

Get all details available on a particular audit item.

6.3.1 Method auditSetSettings definition

```
<?php
public function auditSetSettings($history, $email = null, $callbackUrl = null) { }
```

Table 6.3: **Parameter**

Parameter	Data Type	Default value	Required	Description
$history	int		yes	Number of saved days in history
$email	string		no	Email to send notifications to
$callbackUrl	string		no	URL to send notification to

6.3.2 auditSetSettings information

Return value	\ZendService\ZendServerAPI\DataTypes\AuditSettings (AuditSettings api doc)
Online reference	auditSetSettings online reference
Available in Version	• 1.3

6.3.3 Example

```
<?php
use ZendService\ZendServerAPI\Audit;

$audit = new Audit();
$audit->auditSetSettings(20, "a@b.com", "http://www.test.com");
```

CHAPTER 7

Administration methods

The following is a list of methods available for the Administration feature:

- userAuthenticateSettings
- userSetPassword
- setPassword
- apiKeysGetList
- apiKeysAddKey
- apiKeyRemove
- serverValidateLicense
- aclSetGroups
- bootstrapSingleServer

7.1 The userAuthenticateSettings Method

Modify current authentication settings, allowing the user to switch between simple and extended authentication and authorization schemes.

7.1.1 Method userAuthenticateSettings definition

```
<?php
public function userAuthenticateSettings($type, $password, $confirmNewPassword, $ldap = array()) { }
```

Table 7.1: **Parameter**

Parameter	Data Type	Default value	Required	Description
$type	string		yes	One of : simple, extended
$password	string		yes	Current user's password for authentication
$confirmNewPassword	string		yes	Confirmation of new password
$ldap	array		no	Array of ldap properties: host: host, ip or location of the active directory port: port part of the URL above encryption: ssl: use SSL to secure communications tls: start TLS to secure communications none: no encryption is used username: directory username, broken to CN and DC parts for use in querying the active directory password: matching password for the above username baseDn: DN broken down to CN and DC parts for using during user authentication

7.1.2 userAuthenticateSettings information

Return value	\ZendService\ZendServerAPI\DataTypes\AuthenticationType (AuthenticationType api doc)
Online reference	userAuthenticateSettings online reference
Available in Version	• 1.3

7.1.3 Example

```
<?php
use ZendService\ZendServerAPI\Administration;

$administration = new Administration();
$administration->userAuthenticateSettings("extended", "currentPassword", "newPassword",
```

```
    array(
        "host" => "internalldap",
        "port" => 636,
        "encryption" => "ssl",
        "username" => "admin",
        "password" => "currentPassword",
        "baseDn" => "internal"
    )
);
```

7.2 The userSetPassword Method

Modify a specific user password. This action changes any user password and is an administrative action. Note that a separate action exists for the user to modify his own password and has a lower permission level.

7.2.1 Method userSetPassword definition

```
<?php
public function userSetPassword($username, $password, $newPassword, $confirmNewPassword) { }
```

Table 7.2: **Parameter**

Parameter	Data Type	Default value	Required	Description
$username	string		yes	admin (for Administrator) testuser (for Developer)
$password	string		yes	Current password
$newPassword	string		yes	New password
$confirmNewPassword	string		yes	Confirmation of new password

7.2.2 userSetPassword information

Return value	\ZendService\ZendServerAPI\DataTypes\UserInfo (UserInfo api doc)
Online reference	userSetPassword online reference
Available in Version	• 1.3

7.2.3 Example

```
<?php
use ZendService\ZendServerAPI\Administration;

$administration = new Administration();
$administration->userSetPassword("admin", "oldpassword", "newpassword", "newpassword");
```

7.3 The setPassword Method

Modify a current user password.

7.3.1 Method setPassword definition

```php
<?php
public function setPassword($password, $newPassword, $confirmNewPassword) { }
```

Table 7.3: **Parameter**

Parameter	Data Type	Default value	Required	Description
$password	string		yes	Current password
$newPassword	string		yes	New password
$confirmNewPassword	string		yes	Confirmation of new password

7.3.2 setPassword information

Return value	\ZendService\ZendServerAPI\DataTypes\UserInfo (UserInfo api doc)
Online reference	setPassword online reference
Available in Version	• 1.3

7.3.3 Example

```php
<?php
use ZendService\ZendServerAPI\Administration;

$administration = new Administration();
$administration->setPassword("oldpassword", "newpassword", "newpassword");
```

7.4 The ApiKeysGetList Method

Get a list of api keys.

7.4.1 Method apiKeysGetList definition

```php
<?php
public function apiKeysGetList() { }
```

7.4.2 apiKeysGetList information

Return value	\ZendService\ZendServerAPI\DataTypes\UserInfo (ApiKeys api doc)
Online reference	apiKeysGetList online reference
Available in Version	• 1.3

7.4.3 Example

```
<?php
use ZendService\ZendServerAPI\Administration;

$administration = new Administration();
$apiKeys = $administration->apiKeysGetList();
```

7.5 The apiKeysAddKey Method

Add a WebAPI Key.

7.5.1 Method apiKeysAddKey definition

```
<?php
public function apiKeysAddKey($name, $username) { }
```

Table 7.4: **Parameter**

Parameter	Data Type	Default value	Required	Description
$name	string		yes	The name of the key
$username	string		yes	Any username supplied for retrieving ACL information

7.5.2 apiKeysAddKey information

Return value	\ZendService\ZendServerAPI\DataTypes\ApiKeys (ApiKeys api doc)
Online reference	apiKeysAddKey online reference
Available in Version	• 1.3

7.5.3 Example

```
<?php
use ZendService\ZendServerAPI\Administration;

$administration = new Administration();
$apiKeys = $administration->apiKeysAddKey("foo", "admin");
```

7.6 The apiKeysRemoveKey Method

Remove a WebAPI Key.

7.6.1 Method apiKeysRemoveKey definition

```
<?php
public function apiKeysRemoveKey($ids) { }
```

Table 7.5: **Parameter**

Parameter	Data Type	Default value	Required	Description
$ids	array		yes	array of api key ids to remove

7.6.2 apiKeysRemoveKey information

Return value	\ZendService\ZendServerAPI\DataTypes\ApiKeys (ApiKeys api doc)
Online reference	apiKeysRemoveKey online reference
Available in Version	• 1.3

7.6.3 Example

```
<?php
use ZendService\ZendServerAPI\Administration;

$administration = new Administration();
$apiKeys = $administration->apiKeysRemoveKey(array(5, 6, 7));
```

7.7 The serverValidateLicense Method

Validate a Zend Server license.

7.7.1 Method serverValidateLicense definition

```
<?php
public function serverValidateLicense($licenseName, $licenseValue) { }
```

Table 7.6: **Parameter**

Parameter	Data Type	Default value	Required	Description
$licenseName	string		yes	The name of the license
$licenseValue	string		yes	The value of the license

7.7.2 serverValidateLicense information

Return value	\ZendService\ZendServerAPI\DataTypes\LicenseValidated (LicenseValidated api doc)
Online reference	serverValidateLicense online reference
Available in Version	• 1.3

7.7.3 Example

```
<?php
use ZendService\ZendServerAPI\Administration;

$administration = new Administration();
$licenseValidated = $administration->serverValidateLicense("TRIAL-1795-69", "V1Q26G1VUK185031C5ACF716
```

7.8 The serverStoreLicense Method

Stores a Zend Server license.

7.8.1 Method serverStoreLicense definition

```
<?php
public function serverStoreLicense($licenseName, $licenseValue) { }
```

Table 7.7: **Parameter**

Parameter	Data Type	Default value	Required	Description
$licenseName	string		yes	The name of the license
$licenseValue	string		yes	The value of the license

7.8.2 serverStoreLicense information

Return value	\ZendService\ZendServerAPI\DataTypes\LicenseValidated (LicenseValidated api doc)
Online reference	serverValidateLicense online reference
Available in Version	• 1.3

7.8.3 Example

```
<?php
use ZendService\ZendServerAPI\Administration;

$administration = new Administration();
$licenseValidated = $administration->serverStoreLicense("TRIAL-1795-69", "V1Q26G1VUK185031C5ACF71656
```

7.9 The aclSetGroups Method

Store a set of group mappings for resolving user roles during authentication. These groups correspond to roles within the system or to applications that implicitly grant the developerLimited role to the user.

7.9.1 Method aclSetGroups definition

```
<?php
public function aclSetGroups($roleGroups, $appGroups = null) { }
```

Table 7.8: **Parameter**

Parameter	Data Type	Default value	Required	Description
$roleGroups	array		yes	An associative list of role names and their corresponding group.
$appGroups	array		no	An associative list of application IDs (numbers) and their corresponding group.

7.9.2 aclSetGroups information

Return value	\ZendService\ZendServerAPI\DataTypes\LicenseValidated (LicenseValidated api doc)
Online reference	aclSetGroups online reference
Available in Version	• 1.3

7.9.3 Example

```
<?php
use ZendService\ZendServerAPI\Administration;

$administration = new Administration();
$roleGroups = $administration->->aclSetGroups(array("developer" => "bar"));
```

7.10 The bootstrapSingleServer Method

Bootstrap a server for standalone usage in production or development environment. This action is designed to give an automated process the option to bootstrap a server with particular settings. Note that once a server has been bootstrapped, it may not be added passively into a cluster using clusterAddServer. It may still join a cluster using a direct WebAPI -serverAddToCluster, or a UI call. This WebAPI action is explicitly accessible without a WebAPI Key, but only during the bootstrap stage. Unlike the UI bootstrap/launching process, this bootstrap action does not restart Zend Server nor perform any authentication. A WebAPI key with administrative permissions is created as part of the bootstrap process so that you may immediately continue working. It is up to the user to decide what to do with this key once the bootstrap is completed. Read a certain number of log lines from the end of the file log. If serverId is passed, then the request will be performed against that cluster member, otherwise it is performed locally.

7.10.1 Method bootstrapSingleServer definition

```php
<?php
public function bootstrapSingleServer(
        $adminPassword,
        $orderNumber,
        $licenseKey,
        $acceptEula,
        $production = null,
        $applicationUrl = null,
        $adminEmail = null,
        $developerPassword = null
    ) { }
```

Table 7.9: **Parameter**

Parameter	Data Type	Default value	Required	Description
$adminPassword	string		yes	The new administrator password to store for authentication
$orderNumber	string		yes	License order number to store in the server's configuration. This license can be obtained from zend.com
$licenseKey	string		yes	License key to store in the server's configuration. This license can be obtained from zend.com
$acceptEula	bool		yes	Must be set to true to accept ZS6's EULA
$production	bool		no	Bootstrap this server using the factory "production" usage profile. Default value: true
$applicationUrl	string		no	The default application URL to use when displaying and handling deployed application URLs in the UI. Default: empty
$adminEmail	string		no	The default Email to use when sending notifications about events, audit entries and other features
$developerPassword	string		no	The new developer user password to be stored for authentication. If no password is supplied, the developer user will not be created

7.10.2 bootstrapSingleServer information

Return value	\ZendService\ZendServerAPI\DataTypes\Bootstrap (Bootstrap api doc)
Online reference	bootstrapSingleServer online reference
Available in Version	• 1.3

7.10.3 Example

```php
<?php
use ZendService\ZendServerAPI\Administration;

$administration = new Administration();
$bootstrap = $administration->bootstrapSingleServer("test", "ON", "LC", true);
```

CHAPTER 8

Jobqueue methods

Job Queue API actions provide external actors with ways to query and manipulate jobs and their recurring definitions. The following is a list of methods available for the Job Queue feature:

 The jobqueueListJobs Method The jobqueueJobInfo Method The jobqueueDeleteJob Method The jobqueueRequeueJob Method The jobqueueListRules Method The jobqueueRuleInfo Method The jobqueueSaveRule Method The jobqueueDisableRules Method The jobqueueResumeRules Method The jobqueueDeleteRules Method The jobqueueRunNowRule Method

8.1 The jobqueueJobsList Method

Job Queue API actions provide external actors with ways to query and manipulate jobs and their recurring definitions.

8.1.1 Method jobqueueJobsList definition

```
<?php
public function jobqueueJobsList($limit = null, $offset = null, $orderBy = null, $direction = null,
```

Table 8.1: **Parameter**

Parameter	Data Type	Default value	Required	Description
$limit	int		no	Row limit to retrieve, defaults to value defined in zend-user-user.ini
$offset	int	0	no	The page offset to be displayed, defaults to 0
$orderBy	string	Date	no	Column to sort the result by (), defaults to Date
$direction	string	DESC	no	Sorting direction: ASC or DESC.
$filters	array	array()	no	Associative array, accteps any of the following keys: app_id, name, script, priority, status, rule_id, scheduled_before, scheduled_after, executed_before, executed_after, freeText The priority key, accepts the following values: low, normal, high, urgent. The status key, accepts the following values: Active, Waiting, Running, Completed, Failed, Timeout, Removed, Scheduled, Suspende

8.1.2 jobqueueJobsList information

Return value	\ZendService\ZendServerAPI\DataTypes\Jobs (Jobs api doc)
Online reference	jobqueueJobsList online reference
Available in Version	• 1.3

8.1.3 Example

```
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$jobs = $jobqueue->jobqueueJobsList();
```

8.2 The jobqueueJobInfo Method

Retrieve and display details of a job.

8.2.1 Method jobqueueJobInfo definition

```
<?php
public function jobqueueJobInfo($id) { }
```

Table 8.2: **Parameter**

Parameter	Data Type	Default value	Required	Description
$id	int		yes	job id

8.2.2 jobqueueJobInfo information

Return value	\ZendService\ZendServerAPI\DataTypes\JobInfo (JobInfo api doc)
Online reference	jobqueueJobInfo online reference
Available in Version	• 1.3

8.2.3 Example

```
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$jobInfo = $jobqueue->jobqueueJobInfo(1);
```

8.3 The jobqueueDeleteJobs Method

Delete job queue.

8.3.1 Method jobqueueDeleteJobs definition

```php
<?php
public function jobqueueDeleteJobs(array $ids) { }
```

Table 8.3: **Parameter**

Parameter	Data Type	Default value	Required	Description
$ids	array		yes	job ids

8.3.2 jobqueueDeleteJobs information

Return value	\ZendService\ZendServerAPI\DataTypes\Jobs (Jobs api doc)
Online reference	jobqueueDeleteJobs online reference
Available in Version	• 1.3

8.3.3 Example

```php
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$jobs = $jobqueue->jobqueueDeleteJobs(array(1));
```

8.4 The jobqueueRequeueJobs Method

Requeue a job.

8.4.1 Method jobqueueRequeueJobs definition

```php
<?php
public function jobqueueRequeueJobs(array $ids) { }
```

Table 8.4: **Parameter**

Parameter	Data Type	Default value	Required	Description
$ids	array		yes	job ids

8.4.2 jobqueueRequeueJobs information

Return value	\ZendService\ZendServerAPI\DataTypes\Jobs (Jobs api doc)
Online reference	jobqueueRequeueJobs online reference
Available in Version	• 1.3

8.4.3 Example

```
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$jobs = $jobqueue->jobqueueRequeueJobs(array(1));
```

8.5 The jobqueueListRules Method

Retrieve and display a list of jobs rules.

8.5.1 Method jobqueueListRules definition

```
<?php
public function jobqueueListRules($limit = null, $offset = null, $orderBy = null, $direction = null)
```

Table 8.5: **Parameter**

Parameter	Data Type	Default value	Required	Description
$limit	int		no	Row limit to retrieve, defaults to value defined in zend-user-user.ini
$offset	int	0	no	The page offset to be displayed, defaults to 0
$orderBy	string	Date	no	Column to sort the result by (), defaults to Date
$direction	string	DESC	no	Sorting direction: ASC or DESC.

8.5.2 jobqueueListRules information

Return value	\ZendService\ZendServerAPI\DataTypes\Jobs (Rules api doc)
Online reference	jobqueueListRules online reference
Available in Version	• 1.3

8.5.3 Example

```php
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$rules = $jobqueue->jobqueueListRules();
```

8.6 The jobqueueRuleInfo Method

Retrieve and display a job rule information.

8.6.1 Method jobqueueRuleInfo definition

```php
<?php
public function jobqueueRuleInfo($id) { }
```

Table 8.6: **Parameter**

Parameter	Data Type	Default value	Required	Description
$id	int		yes	job id

8.6.2 jobqueueRuleInfo information

Return value	\ZendService\ZendServerAPI\DataTypes\RuleInfo (RuleInfo api doc)
Online reference	jobqueueRuleInfo online reference
Available in Version	• 1.3

8.6.3 Example

```php
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$ruleInfo = $jobqueue->jobqueueRuleInfo(1);
```

8.7 The jobqueueSaveRule Method

Create a job queue rule.

8.7.1 Method jobqueueSaveRule definition

```php
<?php
public function jobqueueSaveRule($url, $options, $vars = array()) { }
```

Table 8.7: **Parameter**

Parameter	Data Type	Default value	Required	Description
$url	string		yes	A URL for the job.
$options	string		yes	Rule options. (schedule pattern)
$vars	array		no	Variables for the rule.

8.7.2 jobqueueSaveRule information

Return value	\ZendService\ZendServerAPI\DataTypes\RuleInfo (RuleInfo api doc)
Online reference	jobqueueSaveRule online reference
Available in Version	• 1.3

8.7.3 Example

```
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$ruleInfo = $jobqueue->jobqueueSaveRule("http://www.example.com/foo", "1 */10");
```

8.8 The jobqueueDisableRules Method

Suspend a job queue rule.

8.8.1 Method jobqueueDisableRules definition

```
<?php
public function jobqueueDisableRules(array $ruleIds) { }
```

Table 8.8: **Parameter**

Parameter	Data Type	Default value	Required	Description
$ruleIds	array		yes	Array of rule ids

8.8.2 jobqueueDisableRules information

Return value	\ZendService\ZendServerAPI\DataTypes\Rules (Rules api doc)
Online reference	jobqueueDisableRules online reference
Available in Version	• 1.3

8.8.3 Example

```
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$rules = $jobqueue->jobqueueDisableRules(array(1));
```

8.9 The jobqueueResumeRules Method

Resume a suspended job queue rule.

8.9.1 Method jobqueueResumeRules definition

```
<?php
public function jobqueueResumeRules(array $ruleIds) { }
```

Table 8.9: **Parameter**

Parameter	Data Type	Default value	Required	Description
$ruleIds	array		yes	Array of rule ids

8.9.2 jobqueueResumeRules information

Return value	\ZendService\ZendServerAPI\DataTypes\Rules (Rules api doc)
Online reference	jobqueueResumeRules online reference
Available in Version	• 1.3

8.9.3 Example

```
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$rules = $jobqueue->jobqueueResumeRules(array(1));
```

8.10 The jobqueueDeleteRules Method

Delete a job queue rule.

8.10.1 Method jobqueueDeleteRules definition

```
<?php
public function jobqueueDeleteRules(array $ruleIds) { }
```

Table 8.10: **Parameter**

Parameter	Data Type	Default value	Required	Description
$ruleIds	array		yes	Array of rule ids

8.10.2 jobqueueDeleteRules information

Return value	\ZendService\ZendServerAPI\DataTypes\Rules (Rules api doc)
Online reference	jobqueueDeleteRules online reference
Available in Version	• 1.3

8.10.3 Example

```
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$rules = $jobqueue->jobqueueDeleteRules(array(1));
```

8.11 The jobqueueRunNowRule Method

Run a scheduled job that was scheduled for a later time.

8.11.1 Method jobqueueRunNowRule definition

```
<?php
public function jobqueueRunNowRule($ruleId) { }
```

Table 8.11: **Parameter**

Parameter	Data Type	Default value	Required	Description
$ruleId	int		yes	Rule id

8.11.2 jobqueueRunNowRule information

Return value	\ZendService\ZendServerAPI\DataTypes\RuleInfo (RuleInfo api doc)
Online reference	jobqueueRunNowRule online reference
Available in Version	• 1.3

8.11.3 Example

```php
<?php
use ZendService\ZendServerAPI\Jobqueue;

$jobqueue = new Jobqueue();
$ruleInfo = $jobqueue->jobqueueRunNowRule(1);
```

search

C