

Software Requirements Specification:

Web API 1.2

PHP & UI Team, Zend Technologies, Inc.

Technical

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Date | Author | Description |
| 1 | 2011-09-18 | Yonni Mendes | * Initial import of WebAPI 1.2 * Basic outline for the new codetracing, monitor and studio actions * Added new data types |
| 2 | 2011-09-21 | Yonni Mendes | Added new data types  Extended all entries with examples  Corrected examples to keep to standards |
| 3 | 2011-09-27 | Yonni Mendes | Modified code tracing enable/list/disable and isEnabled actions |
| 4 | 2011-10-09 | Yonni Mendes | Added codeTracing element to eventGroupDetails  Removed traceId element from eventGroup element |
| 5 | 2011-10-26 | Yonni Mendes | Added startDebug optional parameters  Modified codeTracing element in getRequestSummary output to be an identifier only |

Table of Contents

Revision History 2

Table of Contents 3

1 Introduction 4

2 References 5

3 API Specifications 6

4 Supported Methods 21

5 Open Issues 63

# Introduction

The following document describes the requirements and high-level specifications for the Zend Server Web API version 1.2

# References

1. Web API 1.0 PRD

https://il-cms1.zend.net/svn/Zend/UI/zwas/docs/PRD/ZS-5.1-WebAPI-PRD.pdf

1. Zend Deployment PRD

https://il-cms1.zend.net/svn/Zend/UI/zwas/docs/PRD/ZS-5.2-Application-Deployment.pdf

# API Specifications

## Generic Request / Response Format

The following chapter describes the generic formatting of all Zend Server Web API requests and responses, regardless of the specific method used.

All Web API HTTP requests and responses will be encoded using UTF-8 character encoding.

### Request Format

#### Request Method, URL and Headers

Web API HTTP requests will use HTTP GET for read-only API calls, and HTTP POST for all state changing API calls.

The request URL will be different for each API version and action, and will be of the following format in Zend Server:

http://example.com:10081/ZendServer/Api/<ACTION>

Or in Zend Server Cluster Manager:

http://example.com:10081/ZendServerManager/Api/<ACTION>

Where <ACTION> is the action to perform (e.g. “disableServer”).

All HTTP requests must include the following HTTP headers:

* **Date** – should contain the current date and time in the GMT time zone, in the format specified by the HTTP RFC for date fields (e.g. “Wed, 07 Jul 2010 17:10:55 GMT”). This value will be used to verify the authenticity of the request, and is expected to be in sync with the server time up to the accuracy of ±30 seconds.
* **User-agent –** the user agent string will be logged by the server and used for message authenticity verification. It must not be empty.
* **Host** – the HTTP host header is expected to be present, and will be used for message authenticity verification
* **X-Zend-Signature** – the API key name and calculated request signature to be used for authenticating and validating the request. See section Authentication and Message Verification for additional information on calculating the signature.

In addition, clients should send the ‘Accept’ HTTP request header to designate their supported API version(s). If the ‘Accept’ header is missing, the server will fall back to the default API version. This is described in detail in the API Versioning section below.

For POST requests including any parameters or payload, clients must set the ‘**Content-type**’ header to either ‘*application/x-www-form-urlencoded’* or ‘*multipart/form-data*’, depending on payload. The size of the request body must be specified by clients [as required by the HTTP/1.1 protocol](http://tools.ietf.org/html/rfc2616) – that is by using the “Content-length” header, the “Content-transfer-encoding: chunked” header or simply by closing the connection.

#### Passing Request Parameters

For API methods that require passing parameters, these may be passed in the following forms:

For GET requests, parameters should be passed in the URL query part (following the ‘?’) in a URL-encoded format, similar to how HTML forms sent using the GET method are encoded. “Clean URL” style parameter passing (in the path part of the URL instead of the query part) is not supported.

For POST requests, parameters should be passed in the request body, encoded using either the ‘*application/x-www-form-urlencoded’* (as specified by the [HTML 4.01 standard](http://www.w3.org/TR/html401/interact/forms.html)) or ‘*multipart/form-data*’ (as specified in [RFC-2388](http://www.ietf.org/rfc/rfc2388.txt)) encoding methods.

For some methods (namely methods that may transfer large amounts of binary data), the ‘*multipart/form-data’* encoding method must be used.

Refer to specific method documentation for a list of required and optional parameters.

#### Example

The following is an example of a call to the (obviously fake) “makePizza” method (some lines are broken for readability):

POST /ZendServerManager/Api/makePizza HTTP/1.1  
Host: zscm.local  
Date: Sun, 11 Jul 2010 13:16:10 GMT  
User-agent: Zend\_Http\_Client/1.10  
Accept: application/vnd.zend.serverapi+xml;version=1.2  
X-Zend-Signature: Steve Buscemi;  
 7f0db29a3d82a81ec6f5387f5aae96e295530b4c8acf2074488185902dc900f4  
Content-type: application/x-www-form-urlencoded  
Content-length: 100  
  
style=thinCrust&extraCheese=TRUE&extras%5B0%5D=pepperoni&extras%5B1%5D=onion&extras%5B2%5D=pineapple

The request above is for the “makePizza” method, with the following parameters: style, extraCheese, extras.

The following example shows a call to a read-only “getPizzaStatus” method:

GET /ZendServerManager/Api/getPizzaStatus?pizzaId=53 HTTP/1.1  
Host: zscm.local  
Date: Sun, 11 Jul 2010 13:16:10 GMT  
User-agent: Zend\_Http\_Client/1.10  
Accept: application/vnd.zend.serverapi+xml;version=1.0  
X-Zend-Signature: Jenna Jameson;  
 02dcbf4cb338a0a8b807c83a84a7888929f5c06491105d6752f290da47a24619

Notice that the ‘pizzaId’ parameter is passed as part of the URL’s query string.

### Response Format

API HTTP response messages will use standard HTTP response codes to designate high-level status (success, failure, etc.) and simple XML payload in the response body to provide additional method specific data or specific error messages.

#### HTTP Response Codes

The following HTTP response codes will be used to indicate the overall success / failure of the request:

|  |  |
| --- | --- |
| HTTP Code | Description |
| 200 OK | The operation was completed successfully |
| 202 Accepted | The operation has been accepted and is being processed, but processing is not complete yet. |
| 400 Bad Request | The request is not understood by the server, has bad format, is missing some required parameters or is otherwise not acceptable by the server |
| 401 Unauthorized | Authentication error, the user is unauthorized to perform the action |
| 404 Not Found | Resource does not exist |
| 405 Method Not Allowed | Method is not implemented by this edition of Zend Server (e.g. cluster related methods on Zend Server) |
| 406 Not Accepted | The server does not support any of the content types or API versions specified by the client in the “Accept” header |
| 500 Internal Server Error | An error has occurred on the server while processing the request |
| 503 Service Unavailable | A temporary situation is preventing the server from fulfilling the request |

#### HTTP Response Headers

The following HTTP response headers will be included in API responses:

* **Content-type –** unless stated otherwise, will be “*application/vnd.zend.serverapi+xml; version=<API version>”*. For more information about API versions, see the API Versioning section below.

#### HTTP Response Body

The HTTP response body will contain XML of the following format, unless explicitly specified otherwise (e.g. in the case of The configurationImport method)

<?xml version=*"1.0"* encoding=*"UTF-8"*?>  
<zendServerAPIResponse   
 xmlns=“http://www.zend.com/server/api/1.2”>  
   
 <requestData>  
 <apiKeyName>angel.eyes</apiKeyName>  
 <method>clusterGetServerStatus</method>  
 </requestData>  
   
 <responseData>  
 [response data here...]  
 </responseData>  
  
</zendServerAPIResponse>

All API responses will be enclosed in <zendServerAPIResponse> tags, and will contain two sections: <requestData>, which will include some reference information about the request, and <responseData>, which will include the response data.

The <responseData> section will differ depending on the specific method called. Refer to specific method documentation for additional information.

In error responses, the <responseData> section will be replaced with an <errorData> section. See Error Responses section for additional information.

#### Error Responses

In a response representing an error, the <responseData> XML section will be replaced with the <errorData> XML section, which will have the following format:

<errorData>  
 <errorCode>serverDoesNotExist</errorCode>  
 <errorMessage>A server with the specified ID does not exist in the cluster</errorMessage>  
</errorData>

Where:

* <errorCode> is a short alphanumeric constant string that represents the specific error
* <errorMessage> is a human readable, native language explanation of the error

In addition, some error responses may include additional elements in the <errorData> container, with additional information relevant to the specific error.

The following generic error responses are possible for any operation:

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 400 | missingHttpHeader | A required HTTP header is missing |
| 400 | unexpectedHttpMethod | Unexpected HTTP method, GET used but POST expected |
| 400 | invalidParameter | One or more request parameter contains invalid data |
| 400 | missingParameter | Request is missing a required parameter |
| 400 | unknownMethod | Unknown Zend Server API method |
| 400 | malformedRequest | The server is unable to understand the request |
| 401 | authError | Authentication error, unknown key or invalid request signature |
| 401 | insufficientAccessLevel | User is not authorized to perform this action |
| 401 | timeSkewError | Request timestamp deviates too much from server time |
| 405 | notImplementedByEdition | Method is not implemented by this edition of Zend Server |
| 406 | unsupportedApiVersion | API version is not supported by this version of Zend Server |
| 500 | internalServerError | An unexpected error on the server side |
| 500 | serverNotConfigured | This Zend Server installation was not yet initialized (user did not go through the initial setup wizard) |
| 500 | serverNotLicensed | Server does not have a valid license which is required to perform this operation |

Refer to the documentation of each specific method for details about additional possible errors specific to different methods.

## API Versioning

### Versions History

|  |  |
| --- | --- |
| Zend Server Version | API Version |
| 5.1.0 | 1.0 |
| Oak | 1.1 |
| Palm | 1.2 |

### Versioning Policy

The following policy should be applied when modifying the Zend Server Web API in order to choose new API versions:

* API version numbers will be composed of two numbers separated by a decimal point, designating a major API version and a minor API version. For example “1.12” designates major API version 1 and minor API version 12.
* Each modification to the API must be designated with a new version number
* Backwards compatible changes to the API will be designated by a minor version change – for example servers supporting version 1.12 will be inherently backwards compatible with clients supporting only version 1.0.
  + Backwards compatible changes include adding new methods without modifying the signature or response format of old methods, and adding new optional parameters or optional response elements to existing methods
  + This makes servers and clients supporting API version 1.12 inherently backwards compatible with all API versions between 1.0 and 1.12.
* Changes that break backwards compatibility must be designated by a major version change – for example the meaning of a parameter in API version 2.1 may be different from the meaning of the same parameter in API version 1.2.

In addition, the following policies must be maintained in order to maintain backwards compatibility between Zend Server versions:

* Zend Server releases must continue to support at least 1 major API version back
* Support for older API versions can only be dropped in major Zend Server releases

### API Version Negotiation

As a clients performs an API calls, it should specify its currently used API version as part of the Zend Server API media type in the “Accept” HTTP header. For example, a client sending a request using API version 2.1, should send the following Accept header in the request:

Accept: application/vnd.zend.serverapi+xml;version=2.1

If the server supports the specified API version, it will handle the request and respond in the appropriate format, matching the specified API version. The response format and API version will be specified using the “Content-Type” response header:

Content-type: application/vnd.zend.serverapi+xml;version=2.1

If the server is not compatible with the API version used by the client, the server will return an HTTP 406 “Not Acceptable” response, with supported version content types listed as part of the <errorData> XML. For example:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zendServerAPIResponse

xmlns=*”http://www.zend.com/server/api/1.0”*>

<requestData>

<apiKeyName>angel.eyes</apiKeyName>

<method>clusterGetServerStatus</method>

</requestData>

<errorData>

<errorCode>unsupportedApiVersion</errorCode>

<errorMessage>

Client API version not supported by this server

</errorMessage>

<supportedApiVersions>

application/vnd.zend.serverapi+xml;version=1.0,

application/vnd.zend.serverapi+xml;version=1.1,

application/vnd.zend.serverapi+xml;version=1.2,

application/vnd.zend.serverapi+xml;version=2.0

</supportedApiVersions>

</errorData>

</zendServerAPIResponse>

The client may then choose to switch to a different API version, or give up and issue a failure message to the end user.

Note that the ‘Accept’ request header, while highly recommended, is optional. If a client does not specify the ‘Accept’ header, the server must fall back to using the oldest API version supported by the server.

## Authentication and Message Verification

API requests authentication is done by creating a digital signature of some request specific parameters using an account-specific secret key, and sending this signature, as well as the key name, in the custom *X-Zend-Signature* HTTP header.

The server will compare this signature with the expected signature, calculated based on the same key and parameters as known to the server, and will only authorize the request if the signatures match.

Note that this authentication and validation method does not contradict the use of HTTPS to encrypt the communication channel, which is recommended but not required. Authentication requirements are the same, regardless of whether HTTPS or plain HTTP is used.

### Generating API keys

API keys will be generated using the ZS/ZSCM GUI, or using a command line tool. Keys can be created, accessed or managed only by users that have Administrator access to the Zend Server / ZSCM GUI, or root / Administrator access to the machine.

When generating a new key, the user will be asked to specify a name for this key. This may be a user name, or a group name, which will be used to identify the key and to tell the server which key to use when attempting to authenticate a request.

Valid key names may be composed only of [“token” characters as defined by RFC-2616](http://tools.ietf.org/html/rfc2616) with the addition of the whitespace character and the ‘@’ character. Effectively this allows all printable US-ASCII characters (ASCII characters 0x20 to 0x7e) with the exception of the following characters:

( ) < > , ; : \ " / [ ] ? = { }

In addition, key names may not begin or end with a whitespace character.

The specific API key will also determine the access level granted when using this key.

API keys will be created by generating a 256 bit random number using a cryptographic grade random number generation method, and encoding it as a 64 digit hexadecimal number with digits a-f in lower case.

Note that the API key is a **secret, pre-shared key** – the Zend Server API does not use asymmetric key pairs for authentication or encryption. For this reason each key must be kept secret and immediately revoked if there is any chance that it has been compromised.

### Signing API Requests

#### Importance of the Date header

The Value of the *Date* HTTP header is used as part of the request signing process to enforce the temporary state of signed requests. For this reason, the system clock on the client and server sides must be synchronized, up to an allowed time skew of ±30 seconds.

If the server receives an API request with a *Date* header value that represents more than 30 seconds of time difference (either before or after the server clock), the request will not be accepted.

#### The X-Zend-Signature HTTP header

In order to send authenticated API requests, the user will be required to send the *X-Zend-Signature* HTTP header with each request, with the following format:

X-Zend-Signature: <key name>; <signature>

Where <key name> is to be replaced with the key name, and <signature> is to be replaced with the calculated request signature.

Note that there can be any number of whitespace characters before or after the separating semicolon.

For example (lines are broken for readability):

X-Zend-Signature: Arch Stanton;   
 signature=b5bb9d8014a0f9b1d61e21e796d78dccdf1352f23cd328...

Note that the signature is expected to be 64 characters long, and is cut here for readability purposes.

#### Calculating the Request Signature

The request signature is a 64 digit long hexadecimal number with digits a-f in lower case, calculated using the following method:

1. Concatenate the following values in order, separated by a colon (:) into a single string:
   1. The exact value of the **Host** HTTP header – in most cases, this will be a string of the form <host>:<port>. In some cases the colon and port are omitted. In any case, if the port is included in the Host header sent in the request, it must be included in the generated string.
   2. The **Request URI** – that is the path part of the full request URL, without the query string or host name
   3. The exact value of the **User-Agent** request header
   4. The exact value of the **Date** request header
2. Hash the generated string with the HMAC/SHA-256 function using the secret API key to obtain the request signature

#### Example

For example, when sending the following API request:

POST /ZendServer/Api/findTheFish HTTP/1.1  
Host: zscm.local:10081  
User-agent: Zend\_Http\_Client/1.10  
Accept: application/vnd.zend.serverapi+xml;version=1.0  
Date: Sun, 11 Jul 2010 13:16:10 GMT  
Content-type: application/x-www-form-urlencoded  
Content-length: 19  
  
lookInCupboard=TRUE

Using a key named “angel.eyes” with the following value:

9dc7f8c5ac43bb2ab36120861b4aeda8f9bb6c521e124360fd5821ef279fd9c7

The request parameters to be signed, concatenated into a string will be:

zscm.local:10081:/ZendServer/Api/findTheFish:Zend\_Http\_Client/1.10: Sun, 11 Jul 2010 13:16:10 GMT

From this value, using the API key an HMAC/SHA-256 signature will be calculated, for example using the *hash\_hmac()* PHP function:

142a2d33c76eae62d2234f18694c1867a2a491fe3ce78dcdf4979df84629d877

The final request, including the added *X-Zend-Signature* header, will be (lines are broken for readability):

POST /ZendServer/Api/findTheFish HTTP/1.1  
Host: zscm.local:10081  
User-agent: Zend\_Http\_Client/1.10  
Accept: application/vnd.zend.serverapi+xml;version=1.0  
Date: Sun, 11 Jul 2010 13:16:10 GMT  
Content-type: application/x-www-form-urlencoded  
Content-length: 19  
X-Zend-Signature: angel.eyes;   
 142a2d33c76eae62d2234f18694c1867a2a491fe3ce78dcdf4979df84629d877  
  
lookInCupboard=TRUE

The server will then proceed to generate the same signature, based on the same data and same secret key. If the two signatures match, the request will be accepted.

## Data Types

### Request Data Types

Request data may be encoded into several primitive types. Since all data is eventually represented as UTF-8 strings, these types mostly define what characters are considered valid for data of a specific type. Additional validation rules may apply for specific parameters

* Boolean – a Boolean value, represented as either “TRUE” or “FALSE”, case insensitive
* Integer – an integer number
* String – a string of characters
* TimeStamp – Date and time represented in [RFC-882/RFC-1123 format](http://tools.ietf.org/html/rfc822) (e.g. “Sun, 06 Nov 1994 08:49:37 GMT“); Dates and times must always be represented in the GMT time zone, even if the server or client use a different default time zone.
* Array – an array of values. Arrays are to be encoded by adding square brackets with an incrementing 0-based index number to the parameter name. For example, the array parameter *fruits = (“apple”, “orange”, “banana”)* is to be represented as follows:

fruits[0]=apple&fruits[1]=orange&fruits[2]=banana

Since request parameter names must be URL-encoded the above parameter will actually be sent as:

fruits%5B0%5D=apple&fruits%5B1%5D=orange&fruits%5B2%5D=banana

* Hashmap – a hash map (associative array) of values; Hashmaps are encoded using square brackets after the parameter name, with a key name inside the square brackets (unlike the Array type in which a number based index is used). For example, the hash map UserInfo = { name: Tuco, lastname: Ramirez } will be represented as follows:

UserInfo[name]=Tuco&UserInfo[lastname]=Ramirez

Since request parameter names must be URL-encoded the above parameter will actually be sent as:

UserInfo%5Bname%5D=Tuco&UserInfo%5Blastname%5D=Ramirez

### Response Data Types

Response data types are represented in XML. This allows for more complex object types to be represented in response data.

The following response types are possible in addition to the scalar types defined for request data types (Boolean, Integer, String, TimeStamp).

Each complex type is represented as an XML element, with properties represented as sub-elements. XML element names will always use camelCase notation (first character is lower case).

#### messageList

A list of 0 of more messages

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| info | String | 0+ | Info-level message (may appear 0 or more times) |
| warning | String | 0+ | Warning-level message (may appear 0 or more times) |
| error | String | 0+ | Error-level message (may appear 0 or more times) |

#### serverInfo

An object representing a single server with information about the server

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| id | Integer | 1 | Server ID |
| name | String | 1 | Server name |
| address | String | 1 | Server address as HTTP URL |
| status | String | 1 | Server Status. May be one of the following values:   * OK * shuttingDown * startingUp * pendingRestart * restarting * misconfigured * extensionMismatch * daemonMismatch * applicationMismatch (added in 1.1) * notResponding * disabled * removed * unknown |
| messageList | messageList | 1 | List of messages reported by this server. Can be empty if there are no messages to show. |

#### serversList

A list of servers

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| serversList | serverInfo | 0+ | Server information (may appear more than once) |

#### systemInfo

Generic information about the system being accessed

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| status | String | 1 | Global status information. Can be one of:   * OK – system is operational * notLicensed – system is not licensed. In ZSCM, this means the ZSCM is not licensed. The nodes may be licensed and operating. * pendingRestart – system is pending a PHP restart. In ZSCM this will never be set. |
| edition | String | 1 | Zend Server Edition. Can be one of:   * ZendServer * ZendServerClusterManager * ZendServerCommunityEdition |
| zendServerVersion | String | 1 | Full version of Zend Server (e.g. “5.0.4”) |
| supportedApiVersions | String | 1 | Comma-separated list of supported content types / versions of the Zend Server Web API |
| phpVersion | String | 1 | Full PHP version (e.g. “5.3.3”) |
| operatingSystem | String | 1 | A string identifying the operating system |
| deploymentVersion | String | 1 | A String representing the schema version of the Zend Deployment |
| serverLincenseInfo | licenseInfo | 1 | Information about the Zend Server license. If running in cluster, will contain the node license information |
| managerLicenseInfo | licenseInfo | 1 | Information about the Zend Server Cluster Manager license |
| messageList | messageList | 1 | List of messages reported by this server. Can be empty if there are no messages to show. |

#### licenseInfo

Information about a Zend Server or Zend Server Cluster Manager license

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| status | String | 1 | Licensing status. Can be one of:   * notRequired – this edition does not require this license type * OK – licensed and working * invalid – license is invalid * expired – license has expired * serverLimitExceeded – ZSCM server limit exceeded |
| orderNumber | String | 1 | License order number. Empty if no license. |
| validUntil | TimeStamp | 1 | License expiration date. Empty if no license. |
| serverLimit | Integer | 1 | If this is a ZSCM license, number of servers allowed by the license. If not a ZSCM license, value is always 0 |

#### deployedVersions

A list of deployed versions and rollback versions, contained in an applicationInfo object. May be empty. The list should be in descending order.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| deployedVersion | string | 0+ | Version identifier, such as 6.17 |
| applicationRollbackVersion | String | 0+ | Version identifier, such as 6.17 |

#### applicationServer

Information about an application status on a specific server

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| id | integer | 1 | Server ID |
| deployedVersion | String | 1 | The latest version of the application identified on the server |
| status | string | 1 | The deployedVersion's status, same list as in applicationInfo:status |

#### applicationInfo

Information about a specific deployed application

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| id | Integer | 1 | Application ID |
| baseUrl | String | 1 | Application base URL |
| appName | String | 1 | Application name |
| userAppName | String | 1 | Free text for user defined application identifier |
| installedLocation | String | 1 | The location on the filesytem where the application's source code is located |
| status | String | 1 | Application status; may be one of the following codes:   * uploadError * staging * stageError * activating * deployed * activateError * deactivating * deactivateError * unstaging * unstageError * rollingBack * rollbackError * unknown * partiallyDeployed (available only in ZSCM mode) * notExists |
| servers | applicationServer | 0+ | Breakdown of the the application status and version per server |
| deployedVersions | deployedVersions | 1 | A list of messages related to this application |

#### applicationsList

A list of 0 or more applications

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| applicationsList | applicationInfo | 0+ | Application information |

#### requestSummary

A list of 0 or more events

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| eventsCount | Integer | 1 | Number of events in the events element |
| events | event | 0+ | List of event elements |
| codeTracing | String | 1 | Trace-file identifier |

#### issue

List of basic issue properties

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| id | Integer | 1 | Issue identifier |
| rule | String | 1 | Issue's rule name |
| lastOccurance | Integer | 1 | Issue's last time of manifestation |
| severity | String | 1 | Issue's severity (Warning|Error) |
| status | String | 1 | Issue's current status |
| url | String | 1 | Issue's creating URL string |
| sourceFile | String | 1 | Path to the file where the issue manifested |
| sourceLine | Integer | 1 | Line number where the issue manifests within the sourceFile |
| function | String | 1 | Name of the function that caused the issue to manifest |
| aggregationHint | String | 1 | A unique identifier that groups all events under this issue |
| errorString | String | 1 | The error string generated for the event |
| errorType | String | 1 | PHP Error type created for the event |
| routeDetails | List of routeDetail | 0..\* | Route details for the issue and the request that created it |

#### issueDetails

Detailed view of a single issue

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| issue | Element of type issue | 1 | Issue identifier |
| eventsGroups | List of eventsGroup | 1..\* | Details about event groups in the current issue |

#### routeDetail

Issue Route details – hints provided by the monitor to indicate where or how the issue was produced in a more modular and application-aware display

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| key | String | 1 | Route detail piece's key name |
| value | String | 1 | Route detail piece's value |

#### eventsGroup

Details about an issue's evensGroup. This even describes general details about groups of events, unlike the “event” element which provides in-depth details

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| eventsGroupId | String | 1 | Event Group's identifier |
| eventsCount | Integer | 1 | The number of events in the current event-group |
| startTime | Integer | 1 | Timestamp for the first event in the current event-group |
| serverId | Integer | 1 | Identifier of the cluster-member where the event took place. This field will be empty if no serverId is applicable |
| class | String | 1 |  |
| userData | String | 1 |  |
| javaBacktrace | String | 1 |  |
| execTime | Integer | 1 |  |
| avgExecTime | Integer | 1 |  |
| memUsage | Integer | 1 |  |
| avgMemUsage | Integer | 1 |  |
| avgOutputSize | Integer | 1 |  |
| load | String | 1 |  |

#### eventsGroupDetails

Details about an issue's evensGroup include the actual event data

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| issueId | Integer | 1 | The group's Issue identifier |
| eventsGroup | eventsGroup element | 1 | Basic details about the eventGroup |
| event | event element | 1 | Event with common data for the events group |
| codeTracing | String | 1 | Associated code tracing identifier |

#### event

List of event properties with metadata and backtrace information

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| type | String | 1 | Issue type name |
| description | String | 1 | Free form text field with details about the Issue |
| superGlobals | superGlobals element | 1 | Super global arrays and their values: get, post, cookie, session, server |
| debugUrl | String | 1 | URL for debugging the event |
| severity | String | 1 | Severity indicator for the event: Info, Warning, Critical |
| backtraces | List of step elements | 1 | A list of backtrace step elements |

#### parameter

Name & value pair for parameters exposed to the script's environment

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| name | String | 1 | Parameter name or identifier |
| value | String | 1 | String value of the parameter |

#### superGlobals

List of parameter elements grouped by source – get, post, cookie, session and server

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| get | List of parameter elements | 1 | Available GET parameters |
| post | List of parameter elements | 1 | Available POST parameters |
| cookie | List of parameter elements | 1 | Available COOKIE values |
| session | List of parameter elements | 1 | Available SESSION values |
| server | List of parameter elements | 1 | Available SERVER environment parameters |

#### step

List of backtrace entry properties. Backtrace elements show up in a list of backtraces in which order is important

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| number | Integer | 1 | Sequential numbering of the backtrace steps |
| object | String | 1 | Object identifier |
| class | String | 1 | Object class identifer |
| function | String | 1 | Function or method name |
| file | String | 1 | Filepath |
| line | Integer | 1 | Line number in the file |

#### codeTracingStatus

A list of indicators for code tracing activity and operations

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| componentStatus | String | 1 | Current activity status of the component: Active | Inactive |
| alwaysDump | String | 1 | Current always\_dump directive value (On|Off) |
| traceEnabled | String | 1 | Current trace\_enabled directive value (On|Off) |
| awaitsRestart | Integer | 1 | If true, ZendServer is waiting for a restart which may affect these settings |

#### codeTrace

A single Code trace file's set of properties

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **#** | **Description** |
| id | Integer | 1 | Sequential numbering of the backtrace steps |
| date | Integer | 1 | Creation timestamp |
| url | String | 1 | URL string that created the trace |
| createdBy | String | 1 | Method of creation (Code Request, Manual Request, Monitor Event) |
| filesize | Integer | 1 | File size in bytes |
| applicationId | Integer | 1 | Application context for the trace-file |

# Supported Methods

## Server and Cluster Management Methods

### getSystemInfo

**Description:** get information about the system, including Zend Server edition and version, PHP version, licensing information etc. In general this method should be available and produce similar output on all Zend Server systems, and be as future compatible as possible

DeploymentVersion will show the current deploment version for supported Zend Server versions only. If Zend Server version does not supported deployment 0 (zero) will be used.

**Required Permissions:** read

**HTTP method:** GET

**Supported by Editions:** All

**Request Parameters:** This method has no request parameters

**Expected Response Code:** 200 OK

**Response Type:** systemInfo

**Possible Action Specific Error Codes:** This method has no action-specific error codes

**Example**

Request:

GET /ZendServerManager/Api/getSystemInfo

Response:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.0"*>

<requestData>

<apiKeyName>angel.eyes</apiKeyName>

<method>getSystemInfo</method>

</requestData>

<responseData>

<systemInfo>

<status>OK</status>

<edition>

ZendServerClusterManager

</edition>

<zendServerVersion>6.0.1</zendServerVersion>

<supportedApiVersions>

application/vnd.zend.serverapi+xml;version=1.0,

application/vnd.zend.serverapi+xml;version=1.1,

application/vnd.zend.serverapi+xml;version=2.0

</supportedApiVersions>

<phpVersion>5.4.1</phpVersion>

<operatingSystem>Linux</operatingSystem>

<deploymentVersion>1.0</deploymentVersion>

<serverLicenseInfo>

<status>OK</status>

<orderNumber>ZEND-ORDER-66</orderNumber>

<validUntil>Sat, 31 Mar 2012 00:00:00 GMT</validUntil>

<serverLimit>0</serverLimit>

</serverLicenseInfo>

<managerLicenseInfo>

<status>serverLimitExceeded</status>

<orderNumber>ZEND-ORDER-66</orderNumber>

<validUntil>Sat, 31 Mar 2012 00:00:00 GMT</validUntil>

<serverLimit>10</serverLimit>

</managerLicenseInfo>

</systemInfo>

</responseData>

</zendServerAPIResponse>

An example response for the same request sent to a Zend Server Community Edition Machine would be:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.0"*>

<requestData>

<apiKeyName>angel.eyes</apiKeyName>

<method>getSystemInfo</method>

</requestData>

<responseData>

<systemInfo>

<status>OK</status>

<edition>

ZendServerClusterCommunityEdition

</edition>

<zendServerVersion>6.0.1</zendServerVersion>

<supportedApiVersions>

application/vnd.zend.serverapi+xml;version=1.0,

application/vnd.zend.serverapi+xml;version=1.1,

application/vnd.zend.serverapi+xml;version=2.0

</supportedApiVersions>

<phpVersion>5.4.1</phpVersion>

<operatingSystem>Linux</operatingSystem>

<serverLicenseInfo>

<status>notRequired</status>

<orderNumber />  
 <validUntil />

<serverLimit>0</serverLimit>

</serverLicenseInfo>

<managerLicenseInfo>

<status>notRequired</status>

<orderNumber />  
 <validUntil />

<serverLimit>0</serverLimit>

</managerLicenseInfo>

</systemInfo>

</responseData>

</zendServerAPIResponse>

### clusterGetServerStatus

**Description:** get the list of servers in the cluster and the status of each one. On a ZSCM with no valid license, this operation will fail. Note that this operation will cause 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 in their own code.

**Required Permissions:** read

**HTTP method:** GET

**Supported by Editions:** ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| servers | Array | No | List of server IDs. If specified, status will be returned for these servers only. If not specified, the status of all servers will be returned. |

**Expected Response Code:** 200 OK

**Response Type:** serversList

**Possible Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchServer | One or more of the provided server IDs do not exist in the cluster |
| 405 | notImplementedByEdition | This method is only available on Zend Server Cluster Manager |
| 500 | serverNotLicensed | Zend Server Cluster Manager is not licensed |

**Example**

Request (URI broken for readability):

GET /ZendServerManager/Api/clusterGetServerStatus?   
 servers%5B0%5D=12&servers%5B1%5D=15

Response:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>  
<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.0"*>  
   
 <requestData>  
 <apiKeyName>angel.eyes</apiKeyName>  
 <method>clusterGetServersStatus</method>  
 </requestData>  
   
 <responseData>  
 <serversList>  
 <serverInfo>  
 <id>12</id>  
 <name>www-01</name>  
 <address>https://www-01.local:10082/ZendServer</address>  
 <status>OK</status>  
 <messageList />  
 </serverInfo>  
 <serverInfo>  
 <id>15</id>  
 <name>www-02</name>  
 <address>https://www-02.local:10082/ZendServer</address>  
 <status>pendingRestart</status>  
 <messageList>  
 <warning>This server is waiting a PHP restart</warning>  
 </messageList>  
 </serverInfo>  
 </serversList>  
 </responseData>  
  
</zendServerAPIResponse>

### clusterAddServer

**Description:** Add a new server to the cluster. On a ZSCM with no valid license, this operation will fail.

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| serverName | String | Yes | Server Name |
| serverUrl | String | Yes | Server address, as a full HTTP / HTTPS URL |
| guiPassword | String | Yes | Server GUI password |
| propagateSettings | Boolean | No | Propagate this server’s current settings to the rest of the cluster. Default is FALSE |
| ~~doRestart~~ | ~~Boolean~~ | ~~No~~ | ~~Initiate a PHP restart on the cluster after adding the server; Default is FALSE~~  *Deprecated as of 1.1 in order to support automatic deployment. The system will restart the added server automatically and will ignore this parameter if passed* |

**Expected Response Code:**

* 200 OK – operation was successful
* 202 Accepted – server was added successfully, but setting propagation failed (can only happen if propagateSettings was set to TRUE)

**Response Type:** serverInfo with information about the just-added server

**Possible Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 500 | cantConnectToServer | ZSCM is unable to connect to the specified Server URL |
| 500 | invalidServerResponse | Invalid or unexpected response from new server |
| 400 | wrongPassword | Provided GUI password is incorrect |
| 400 | alreadyConnected | The server is already a member of a cluster (not necessarily the current cluster) |
| 503 | temporarilyLocked | The server cannot be added because a cluster member is in graceful startup / shutdown mode |
| 500 | noActiveServers | The server cannot be added because all servers in the cluster are disabled or unreachable |
| 500 | serverNotLicensed | Zend Server Cluster Manager is not licensed |
| 405 | notImplementedByEdition | This method is only available on Zend Server Cluster Manager |

**Example**

Request (headers removed for clarity):

POST /ZendServerManager/Api/clusterAddServer  
  
serverName=www-05&serverURL=https://www-05.local:10081/ZendServer&  
guiPassword=somepassword&doRestart=TRUE

Response:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>  
<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.0"*>  
   
 <requestData>  
 <apiKeyName>angel.eyes</apiKeyName>  
 <method>clusterAddServer</method>  
 </requestData>  
   
 <responseData>  
 <serverInfo>  
 <id>25</id>  
 <name>www-05</name>  
 <address>https://www-05.local:10082/ZendServer</address>  
 <status>OK</status>  
 <messageList />  
 </serverInfo>  
 </responseData>  
  
</zendServerAPIResponse>

### clusterRemoveServer

**Description:** Remove 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 ZSCM with no valid license, this operation will fail.

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| serverId | String | Yes | Server ID |
| force | Boolean | No | Force-remove the server, skipping graceful shutdown process. Default is FALSE |

**Expected Response Code:**

* 200 OK – the server removal process was completed successfully. This status is expected if there is no need to perform a Graceful Shutdown process, or if the *force* option was set to TRUE.
* 202 Accepted – the removal process has started but was not completed yet. The user may want to check the server status within a few seconds using the *clusterGetServerStatus* method to verify that the operation was complete.

**Response Type:** serverInfo with the status of the server being removed. *Status* is expected to be either *shuttingDown* or *removed*.

**Possible Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchServer | No server with the provided server ID |
| 500 | cantConnectToServer | ZSCM is unable to connect to the specified server |
| 500 | invalidServerResponse | Invalid or unexpected response from the server |
| 503 | temporarilyLocked | The server cannot be removed because another server in the cluster is in graceful startup / shutdown process |
| 405 | notImplementedByEdition | Method is not implemented by this edition of Zend Server |
| 500 | serverNotLicensed | Zend Server Cluster Manager is not licensed |

**Example**

Request (headers removed for clarity):

POST /ZendServerManager/Api/clusterRemoveServer  
  
serverId=5

Response:

HTTP/1.0 202 Accepted  
  
<?xml version=*"1.0"* encoding=*"UTF-8"*?>  
<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.0"*>  
   
 <requestData>  
 <apiKeyName>angel.eyes</apiKeyName>  
 <method>clusterRemoveServer</method>  
 </requestData>  
   
 <responseData>  
 <serverInfo>  
 <id>5</id>  
 <name>www-02</name>  
 <address>https://www-02.local:10082/ZendServer</address>  
 <status>shuttingDown</status>  
 <messageList />  
 </serverInfo>  
 </responseData>  
  
</zendServerAPIResponse>

### clusterDisableServer

**Description:** Disable 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. As long as the server is not fully disabled, further calls to this method should be idempotent. On a ZSCM with no valid license, this operation will fail.

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| serverId | String | Yes | Server ID |

**Expected Response Code:**

* 200 OK – the process was completed successfully. This status is expected if there is no need to perform a Graceful Shutdown process.
* 202 Accepted – the disabling process has started but was not completed yet. The user may want to check the server status within a few seconds using the *clusterGetServerStatus* method to verify that the operation was complete.

**Response Type:** serverInfo with the status of the server being disabled. *status* is expected to be either *shuttingDown* or *disabled*. On a ZSCM with no valid license, this operation will fail.

**Possible Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchServer | No server with the provided server ID |
| 500 | cantConnectToServer | ZSCM is unable to connect to the specified server |
| 500 | invalidServerResponse | Invalid or unexpected response from the server |
| 503 | temporarilyLocked | The server cannot be disabled because another server in the cluster is in graceful startup / shutdown process |
| 405 | notImplementedByEdition | Method is not implemented by this edition of Zend Server |
| 500 | serverNotLicensed | Zend Server Cluster Manager is not licensed |

**Example**

Request (headers removed for clarity):

POST /ZendServerManager/Api/clusterDisableServer  
  
serverID=5

Response:

HTTP/1.0 200 OK  
  
<?xml version=*"1.0"* encoding=*"UTF-8"*?>  
<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.0"*>  
   
 <requestData>  
 <apiKeyName>angel.eyes</apiKeyName>  
 <method>clusterDisableServer</method>  
 </requestData>  
   
 <responseData>  
 <serverInfo>  
 <id>5</id>  
 <name>www-02</name>  
 <address>https://www-02.local:10082/ZendServer</address>  
 <status>disabled</status>  
 <messageList />  
 </serverInfo>  
 </responseData>  
  
</zendServerAPIResponse>

### clusterEnableServer

**Description:** 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. Running it on an enabled server will result in a “200 OK” response with no consequences. On a ZSCM with no valid license, this operation will fail.

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| serverId | String | Yes | Server ID |

**Expected Response Code:**

* 200 OK – the server was enabled successfully. This status is expected if the server is not re-joining the cluster after a Graceful Shutdown process and has no sessions to reclaim
* 202 Accepted – the process has started but was not completed yet. The user may want to check the server status within a few seconds using the *clusterGetServerStatus* method to verify that the operation was complete.

**Response Type:** serverInfo with the status of the server being enabled. *Status* is expected to be either *startingUp* if the server is in the process of re-joining the cluster, or any other active status (*OK, pendingRestart, misconfigured, extensionMismatch, daemonMismatch, notResponding*) if the process was completed.

**Possible Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchServer | No server with the provided server ID |
| 500 | cantConnectToServer | ZSCM is unable to connect to the specified server |
| 500 | invalidServerResponse | Invalid or unexpected response from the server |
| 503 | temporarilyLocked | The server cannot be disabled because another server in the cluster is in graceful startup / shutdown process |
| 405 | notImplementedByEdition | Method is not implemented by this edition of Zend Server |
| 500 | serverNotLicensed | Zend Server Cluster Manager is not licensed |

**Example**

Request (headers removed for clarity):

POST /ZendServerManager/Api/clusterEnableServer  
  
serverID=5

Response:

HTTP/1.0 200 OK  
  
<?xml version=*"1.0"* encoding=*"UTF-8"*?>  
<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.0"*>  
   
 <requestData>  
 <apiKeyName>angel.eyes</apiKeyName>  
 <method>clusterEnableServer</method>  
 </requestData>  
   
 <responseData>  
 <serverInfo>  
 <id>5</id>  
 <name>www-02</name>  
 <address>https://www-02.local:10082/ZendServer</address>  
 <status>pendingRestart</status>  
 <messageList />  
 </serverInfo>  
 </responseData>  
  
</zendServerAPIResponse>

### clusterReconfigureServer

**Description:** Reconfigure a cluster member to match the cluster's profile. On a ZSCM with no valid license, this operation will fail.

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| serverId | String | Yes | Server ID |
| doRestart | Boolean | No | Should the reconfigured server be restarted after the reconfigure action. Default is FALSE |

**Expected Response Code:**

* 200 OK – the server were reconfigured successfully
* 202 Accepted – the server were reconfigured successfully and is now in the process of being restarted

**Response Type:** serverInfo with the status of the server being reconfigured. *Status* is expected to be either *OK* if the server was restarted successfully or if not restart is needed;  *pendingRestart* if the server was reconfigured but not restarted (doRestart was FALSE); *restarting* if the server was reconfigured and is in the process of restarting

**Possible Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchServer | No server with the provided server ID |
| 500 | cantConnectToServer | ZSCM is unable to connect to the specified server |
| 500 | invalidServerResponse | Invalid or unexpected response from the server |
| 503 | temporarilyLocked | The server cannot be reconfigured because it is currently in the middle of another operation (e.g. being disabled) |
| 405 | notImplementedByEdition | Method is not implemented by this edition of Zend Server |
| 500 | serverNotLicensed | Zend Server Cluster Manager is not licensed |
| 500 | restartFailed | Restarting the server failed |

**Example**

Request (headers removed for clarity):

POST /ZendServerManager/Api/clusterReconfigureServer  
  
serverID=5

Response:

HTTP/1.0 200 OK  
  
<?xml version=*"1.0"* encoding=*"UTF-8"*?>  
<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>  
   
 <requestData>  
 <apiKeyName>angel.eyes</apiKeyName>  
 <method>clusterReconfigureServer</method>  
 </requestData>  
   
 <responseData>  
 <serverInfo>  
 <id>5</id>  
 <name>www-02</name>  
 <address>https://www-02.local:10082/ZendServer</address>  
 <status>pendingRestart</status>  
 <messageList />  
 </serverInfo>  
 </responseData>  
  
</zendServerAPIResponse>

### restartPhp

**Description:** Restart 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, and the user is advised to check the server(s) status again after a few seconds using the clusterGetServerStatus command.

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** All

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| servers | Array | No | List of server IDs to restart. If not specified, all servers in the cluster will be restarted. In single Zend Server context this parameter is ignored. |
| parallelRestart | Boolean | No | Send the restart command to all servers at the same time. Default is FALSE |

**Expected Response Code:** 202 Accepted

**Response Type:** serversList with the status of all servers to which the restart command was requested (i.e. the servers provided in the *servers* parameter or all servers if no servers where specified).

**Possible Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchServer | One or more of the provided server IDs do not exist. In this case no servers are restarted. |
| 500 | restartFailed | Restarting at least some of the servers failed. This response is only possible when working with a cluster. |

**Example**

Request (headers removed for clarity):

POST /ZendServerManager/Api/restartPhp  
  
servers%5B0%5D=1&servers%5B1%5D=2

Response:

HTTP/1.0 200 OK  
  
<?xml version=*"1.0"* encoding=*"UTF-8"*?>  
<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.0"*>  
   
 <requestData>  
 <apiKeyName>angel.eyes</apiKeyName>  
 <method>restartPhp</method>  
 </requestData>  
   
 <responseData>  
 <serversList>  
 <serverInfo>  
 <id>1</id>  
 <name>www-01</name>  
 <address>https://www-01.local:10082/ZendServer</address>  
 <status>restarting</status>  
 <messageList />  
 </serverInfo>  
 <serverInfo>  
 <id>2</id>  
 <name>www-02</name>  
 <address>https://www-02.local:10082/ZendServer</address>  
 <status>restarting</status>  
 <messageList />  
 </serverInfo>  
 <serverInfo>  
 <id>3</id>  
 <name>www-03</name>  
 <address>https://www-03.local:10082/ZendServer</address>  
 <status>OK</status>  
 <messageList />  
 </serverInfo>  
 </serversList>  
 </responseData>  
  
</zendServerAPIResponse>

## Configuration Management Methods

### The configurationExport method

**Description:** export the current server / cluster configuration into a file

**Required Permissions:** full

**HTTP method:** GET

**Supported by Editions:** All

**Request Parameters:** this method has no request parameters

**Expected Response Code:** 200 OK

**Response Format:** a successful call to the configurationExport method will result in an HTTP response with the configuration snapshot file in the response body.

The content type for the configuration snapshot file is “application/vnd.zend.serverconfig”. In addition, the response will include a “Content-disposition” header specifying a suggested file name for the configuration snapshot file.

This is different from most Web API calls where the content type is expected to be “application/vnd.zend.serverpi+xml; version=…” and the response body payload is expected to be in XML format.

In case of error, a regular error response will be returned, containing an <errorData> element as defined for other Web API methods.

**Possible Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 500 | exportFailed | Creating a configuration snapshot failed |

**Example**

Request (headers removed for clarity):

GET /ZendServerManager/Api/configurationExport

Response (not all headers are shown):

HTTP/1.0 200 OK  
Content-type: application/vnd.zend.serverconfig  
Content-disposition: attachment;   
 filename=”ZendServerConfig-20101123.zcfg”  
  
[...binary data follows...]

### The configurationImport method

**Description:** import a saved configuration snapshot into the server

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** All

**Request Parameters:**Because this method is expected to contain a file upload, parameters are expected to be encoded using the ‘multipart/form-data’ content type.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| configFile | File | Yes | Configuration snapshot file to import. Content-type for the file must be ‘application/vnd.zend.serverconfig’ |
| ignoreSystemMismatch | Boolean | No | If set to TRUE, configuration must be applied even if it was exported from a different system (other major PHP version, ZS version or operating system). Default is FALSE. |

**Expected Response Code:** 200 OK

**Response Type:** serversList with information about affected servers (one server in Zend Server, all cluster members in Zend Server Cluster Manager)

**Possible Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 500 | importFailed | Importing the configuration snapshot failed |
| 409 | systemMismatch | The system type, PHP version or ZS version from which the configuration snapshot was exported does not match the current system. This error can be overridden if the ignoreSystemMismatch parameter is set to TRUE. |

**Example**

Request (some headers removed for clarity):

POST /ZendServerManager/Api/configurationImport  
Content-type: multipart/form-data, boundary=--bla-bla-bla--  
  
----bla-bla-bla--  
Content-disposition: form-data; name=ignoreSystemMismatch  
  
TRUE  
----bla-bla-bla--  
Content-disposition: form-data; name=”configFile”;   
 filename=”mySavedConfig.zcfg”  
Content-type: application/vnd.zend.serverconfig  
  
[...binary data follows...]  
----bla-bla-bla----

Response:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>  
<zendServerAPIResponse  
 xmlns=*"http://www.zend.com/server/api/1.0"*>  
   
 <requestData>  
 <apiKeyName>angel.eyes</apiKeyName>  
 <method>configurationImport</method>  
 </requestData>  
   
 <responseData>  
 <serversList>  
 <serverInfo>  
 <id>12</id>  
 <name>www-01</name>  
 <address>https://www-01.local:10082/ZendServer</address>  
 <status>pendingRestart</status>  
 <messageList>  
 <warning>This server is waiting a PHP restart</warning>  
 </messageList>  
 </serverInfo>  
 <serverInfo>  
 <id>15</id>  
 <name>www-02</name>  
 <address>https://www-02.local:10082/ZendServer</address>  
 <status>pendingRestart</status>  
 <messageList>  
 <warning>This server is waiting a PHP restart</warning>  
 </messageList>  
 </serverInfo>  
 </serversList>  
 </responseData>  
  
</zendServerAPIResponse>

## Deployment Methods

Deployment methods will be available only from Api version 1.1 and will be included only in Zend Server running Apache and Zend Server Cluster Manager on linux.

**Possible Deployment Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 500 | serverVersionMismatch | One or more servers in the cluster has a Zend Server version which does not support deployment feature. |

### applicationGetStatus

**Description:** get the list of applications currently deployed (or staged) on the server or the cluster and information about each application. If application IDs are specified, will return information about the specified applications; If no IDs are specified, will return information about all applications.

**Required Permissions:** read

**HTTP method:** GET

**Supported by Editions:** ZSCM, ZS

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| applications | Array | No | List of application IDs. If specified, information will be returned about these applications only. If not specified, information about all applications will be returned. Note that if a non-existing application ID is provided, this action will not fail but instead will return no information about the specific app. |

**Expected Response Code:** 200 OK

**Response Type:** applicationsList

**Possible Action Specific Error Codes:** This action has no specific error codes

**Example**

Request:

GET /ZendServerManager/Api/applicationGetStatus?

applications%5B0%5D=1&applications%5B0%5D=2

Response Body:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName>angel.eyes</apiKeyName>

<method>applicationGetStatus</method>

</requestData>

<responseData>

<applicationsList>

<applicationInfo>

<id>1</id>

<baseUrl>http://example.com/myapp</baseUrl>

<appName>Wordpress</appName>

<userAppName>Wolfgang's Blog</userAppName>

<installedLocation>/usr/local/somewhere</installedLocation>

<status>partiallyDeployed</status>

<servers>

<applicationServer>

<id>1</id>

<deployedVersion>1.6</deployedVersion>

<status>OK</status>

</applicationServer>

<applicationServer>

<id>4</id>

<deployedVersion>1.6</deployedVersion>

<status>OK</status>

</applicationServer>

<applicationServer>

<id>8</id>

<deployedVersion>1.5</deployedVersion>

<status>OK</status>

</applicationServer>

</servers>

<deployedVersions>

<deployedVersion>1.6</deployedVersion>

<deployedVersion>1.5</deployedVersion>

<applicationRollbackVersion>1.5</applicationRollbackVersion>

</deployedVersions>

<messageList />

</applicationInfo>

<applicationInfo>

<id>2</id>

<baseUrl>http://oapp.example.com:8080/</baseUrl>

<appName>Blog 2.0</appName>

<userAppName>Wolfgang's Blog</userAppName>

<installedLocation>/usr/local/somewhere</installedLocation>

<status>staging</status>

<servers>

<applicationServer>

<id>1</id>

<deployedVersion>1.6</deployedVersion>

<status>staging</status>

</applicationServer>

<applicationServer>

<id>4</id>

<deployedVersion>1.6</deployedVersion>

<status>staging</status>

</applicationServer>

<applicationServer>

<id>8</id>

<deployedVersion>1.5</deployedVersion>

<status>staging</status>

</applicationServer>

</servers>

<deployedVersions>

<deployedVersion>1.6</deployedVersion>

<deployedVersion>1.5</deployedVersion>

</deployedVersions>

<messageList />

</applicationInfo>

</applicationsList>

</responseData>

</zendServerAPIResponse>

### applicationDeploy

**Description:** deploy a new application to the server or cluster. This process is asynchronous – the initial request will wait until the application is uploaded and verified, and the initial response will show information about the application being deployed – however the staging and activation process will proceed after the response is returned. The user is expected to continue checking the application status using the applicationGetStatus method until the deployment process is complete.

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

Because this method is expected to contain a file upload, parameters are expected to be encoded using the ‘multipart/form-data’ content type.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| appPackage | File | Yes | Application package file. Content type for the file must be ‘application/vnd.zend.applicationpackage’. |
| baseUrl | String | Yes | Base URL to deploy the application to. Must be an HTTP URL. |
| createVhost | Boolean | No | Create a virtual host based on the base URL if such a virtual host wasn't already created by Zend Server.  Default is FALSE |
| defaultServer | Boolean | No | Deploy the application on the default server; the base URL host provided will be ignored and replaced with <default-server>.  In case of a conjunction of this parameter and createVhost, the latter will be ignored.  Default is FALSE |
| userAppName | String | No | Free text for user defined application identifier; if not specified, the baseUrl parameter will be used |
| ignoreFailures | Boolean | No | Ignore failures during staging if only some servers reported failures; If all servers report failures the operation will fail in any case. The default value is FALSE – meaning any failure will return an error |
| userParams | Hashmap | No | Set values for user parameters defined in the package; Depending on package definitions, this parameter may be required; Each user parameter defined in the package must be provided as a key for this parameter |

**Expected Response Code:** 202 Accepted

**Response Type:** applicationInfo

**Possible Action Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 409 | baseUrlConflict | Provided base URL is already in use |
| 400 | missingParameter | A required parameter is missing |
| 400 | invalidParameter | A value provided for a parameter is invalid |
| 400 | missingVirtualHost | The virtual host in the baseUrl does not exist. Use createVhost flag to create it |

**Example**

Request:

POST /ZendServerManager/Api/applicationDeploy

Content-type: multipart/form-data, boundary=--bla-bla-bla--  
  
----bla-bla-bla--  
Content-disposition: form-data; name=baseUrl  
  
http://example.com/  
----bla-bla-bla--  
Content-disposition: form-data; name=”appPackage”;   
 filename=”myApplication.zpk”  
Content-type: application/vnd.zend.applicationpackage

[...binary data follows...]

----bla-bla-bla--  
Content-disposition: form-data; name=userParams%5Bdbhost%5D

database.example.lan

----bla-bla-bla--  
Content-disposition: form-data; name=userParams%5Bdbname%5D

database\_schema

----bla-bla-bla----

Response Body:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName>angel.eyes</apiKeyName>

<method>applicationDeploy</method>

</requestData>

<responseData>

<applicationInfo>

<id>2</id>

<baseUrl>http://oapp.example.com:8080/</baseUrl>

<appName>Blog 2.0</appName>

<userAppName>Wolfgang's Blog</userAppName>

<installedLocation>/usr/local/somewhere</installedLocation>

<status>staging</status>

<servers>

<applicationServer>

<id>1</id>

<deployedVersion>1.6</deployedVersion>

<status>staging</status>

</applicationServer>

<applicationServer>

<id>4</id>

<deployedVersion>1.6</deployedVersion>

<status>staging</status>

</applicationServer>

<applicationServer>

<id>8</id>

<deployedVersion>1.6</deployedVersion>

<status>staging</status>

</applicationServer>

</servers>

<deployedVersions>

<deployedVersion>1.6</deployedVersion>

<deployedVersion>1.5</deployedVersion>

</deployedVersions>

<messageList />

</applicationInfo>

</responseData>

</zendServerAPIResponse>

### applicationUpdate

**Description:** Update an existing application. The package provided must be of the same application. Additionally any new parameters or new values to existing parameters must be provided. This process is asynchronous – the initial request will wait until the package is uploaded and verified, and the initial response will show information about the new version being deployed – however the staging and activation process will proceed after the response is returned. The user is expected to continue checking the application status using the applicationGetStatus method until the deployment process is complete.

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

Because this method is expected to contain a file upload, parameters are expected to be encoded using the ‘multipart/form-data’ content type.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| appId | Integer | Yes | Application ID to update |
| appPackage | File | Yes | Application package file. Content type for the file must be ‘application/vnd.zend.applicationpackage’. |
| ignoreFailures | Boolean | No | Ignore failures during staging if only some servers reported failures; If all servers report failures the operation will fail in any case. The default value is FALSE – meaning any failure will return an error |
| userParams | Hashmap | No | Set values for user parameters defined in the package; Any required parameters that were not defined in previous deployments of the same application will be required. Any parameters with already defined values are not required, but may be specified again if a new value is to be set. |

**Expected Response Code:** 202 Accepted

**Response Type:** applicationInfo

**Possible Action Specific Error Codes:**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchApplication | The provided application ID does not exist |
| 409 | applicationConflict | Provided application package includes a different application, and not a different version of the existing application |
| 400 | missingParameter | A required parameter is missing |
| 400 | invalidParameter | A value provided for a parameter is invalid |

**Example**

Request:

POST /ZendServerManager/Api/applicationUpdate

Content-type: multipart/form-data, boundary=--bla-bla-bla--  
  
----bla-bla-bla--  
Content-disposition: form-data; name=appId   
  
2  
----bla-bla-bla--  
Content-disposition: form-data; name=”appPackage”;   
 filename=”myApplication-v1.1.zpk”  
Content-type: application/vnd.zend.applicationpackage

[...binary data follows...]

----bla-bla-bla--  
Content-disposition: form-data; name=userParams%5Bnewparam%5D

someValueForNewParam

----bla-bla-bla--  
Content-disposition: form-data; name=userParams%5BOldParamRedone%5D

some new value for a parameter that was previously defined

----bla-bla-bla----

Response Body:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName>angel.eyes</apiKeyName>

<method>applicationUpdate</method>

</requestData>

<responseData>

<applicationInfo>

<id>2</id>

<baseUrl>http://oapp.example.com:8080/</baseUrl>

<appName>Blog 2.0</appName>

<userAppName>Wolfgang's Blog</userAppName>

<installedLocation>/usr/local/somewhere</installedLocation>

<status>staging</status>

<servers>

<applicationServer>

<id>1</id>

<deployedVersion>1.6</deployedVersion>

<status>staging</status>

</applicationServer>

<applicationServer>

<id>4</id>

<deployedVersion>1.6</deployedVersion>

<status>staging</status>

</applicationServer>

<applicationServer>

<id>8</id>

<deployedVersion>1.6</deployedVersion>

<status>staging</status>

</applicationServer>

</servers>

<deployedVersions>

<deployedVersion>1.6</deployedVersion>

<deployedVersion>1.5</deployedVersion>

</deployedVersions>

<messageList />

</applicationInfo>

</responseData>

</zendServerAPIResponse>

### applicationRemove

**Description:** Remove / undeploy an existing application. This process is asynchronous – the initial request will start the removal process and the initial response will show information about the application being removed – however the removal process will proceed after the response is returned. The user is expected to continue checking the application status using the applicationGetStatus method until the removal process is complete. Once applicationGetStatus contains no information about the specific application, it has been completely removed

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| appId | Integer | Yes | Application ID to remove |

**Expected Response Code:** 202 Accepted

**Response Type:** applicationInfo

**Possible Action Specific Error Codes:**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchApplication | One or more of the provided application IDs do not exist |

**Example**

Request:

POST /ZendServerManager/Api/applicationRemove

Content-type: application/x-www-form-urlencoded

appId=5

Response Body:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName>angel.eyes</apiKeyName>

<method>applicationRemove</method>

</requestData>

<responseData>

<applicationInfo>

<id>2</id>

<baseUrl>http://oapp.example.com:8080/</baseUrl>

<appName>Blog 2.0</appName>

<userAppName>Wolfgang's Blog</userAppName>

<installedLocation>/usr/local/somewhere</installedLocation>

<status>removing</status>

<servers>

<applicationServer>

<id>1</id>

<deployedVersion>1.6</deployedVersion>

<status>removing</status>

</applicationServer>

<applicationServer>

<id>4</id>

<deployedVersion>1.6</deployedVersion>

<status>removing</status>

</applicationServer>

<applicationServer>

<id>8</id>

<deployedVersion>1.6</deployedVersion>

<status>removing</status>

</applicationServer>

</servers>

<deployedVersions>

<deployedVersion>1.6</deployedVersion>

</deployedVersions>

<messageList />

</applicationInfo>

</responseData>

</zendServerAPIResponse>

### applicationSynchronize

**Description**: Synchronize an existing application, whether in order to fix a problem or to reset an installation. This process is asynchronous. The initial request will start the Synchronize process and the initial response will show information about the application being Synchronized – however the synchronization process will proceed after the response is returned. The user is expected to continue checking the application status using the applicationGetStatus method until the process is complete.

**Required Permissions**: full

**HTTP method**: POST

**Supported by Editions**: ZS, ZSCM

**Request Parameters**:

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| appId | string | Yes | Application ID to Synchronize |
| servers | Array | No | List of server IDs. If specified, action will be done only on the subset of servers which are currently members of the cluster |
| ignoreFailures | Boolean | No | Ignore failures during staging or activation if only some servers reported failures; If all servers report failures the operation will fail in any case. The default value is FALSE – meaning any failure will return an error |

**Expected Response Code**: 202 Accepted

**Response Type**: applicationsList

**Possible Action Specific Error Codes**:

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchApplication | The provided application ID do not exist |

**Example**

Request:

POST /ZendServerManager/Api/applicationSynchronize

Content-type: application/x-www-form-urlencoded

appId=5&servers=...

Response body:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName>angel.eyes</apiKeyName>

<method>applicationSynchronize</method>

</requestData>

<responseData>

<applicationsList>

<applicationInfo>

<id>1</id>

<baseUrl>http://example.com/myapp</baseUrl>

<appName>Wordpress</appName>

<userAppName>Wolfgang's Blog</userAppName>

<installedLocation>/usr/local/somewhere</installedLocation>

<status>activating</status>

<servers>

<applicationServer>

<id>1</id>

<deployedVersion>1.6</deployedVersion>

<status>activating</status>

</applicationServer>

<applicationServer>

<id>4</id>

<deployedVersion>1.6</deployedVersion>

<status>activating</status>

</applicationServer>

<applicationServer>

<id>8</id>

<deployedVersion>1.5</deployedVersion>

<status>activating</status>

</applicationServer>

</servers>

<deployedVersions>

<deployedVersion>1.6</deployedVersion>

<deployedVersion>1.5</deployedVersion>

</deployedVersions>

<messageList />

</applicationInfo>

</applicationsList>

</responseData>

</zendServerAPIResponse>

### applicationRollback

**Description**: Rollback an existing application to its previous version. This process is asynchronous – the initial request will start the rollback process and the initial response will show information about the application being rolled back. The user is expected to continue checking the application status using the applicationGetStatus method until the process is complete.

**Required Permissions**: full

**HTTP method**: POST

**Supported by Editions**: ZS, ZSCM

**Request Parameters**:

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| appId | string | Yes | Application ID to rollback |

**Expected Response Code**: 202 Accepted

**Response Type**: applicationsInfo

**Possible Action Specific Error Codes**:

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchApplication | The provided application ID do not exist |
| 404 | noRollbackAvailable | The application does not have a version to rollback to |

**Example**

Request:

POST /ZendServerManager/Api/applicationRollback

Content-type: application/x-www-form-urlencoded

appId=5

Response body:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName>angel.eyes</apiKeyName>

<method>applicationRollback</method>

</requestData>

<responseData>

<applicationInfo>

<id>1</id>

<baseUrl>http://example.com/myapp</baseUrl>

<appName>Wordpress</appName>

<userAppName>Wolfgang's Blog</userAppName>

<installedLocation>/usr/local/somewhere</installedLocation>

<status>rollingBack</status>

<servers>

<applicationServer>

<id>1</id>

<deployedVersion>1.6</deployedVersion>

<status>rollingBack</status>

</applicationServer>

<applicationServer>

<id>4</id>

<deployedVersion>1.6</deployedVersion>

<status>rollingBack</status>

</applicationServer>

<applicationServer>

<id>8</id>

<deployedVersion>1.5</deployedVersion>

<status>deployed</status>

</applicationServer>

</servers>

<deployedVersions>

<deployedVersion>1.6</deployedVersion>

<deployedVersion>1.5</deployedVersion>

</deployedVersions>

<messageList />

</applicationInfo>

</responseData>

</zendServerAPIResponse>

## Code-tracing Methods

Code Tracing methods provide a programmatic interface for managing, listing and using codetracing amf files generated during run-time or otherwise.

**Possible Code-tracing Actions Specific Error Codes**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchTrace | The requested trace could not be found |

### codetracingDisable

**Description:** Disable the code-tracing directive two directives necessary for creating tracing dumps, this action does not disable the code-tracing component

**Required Permissions:** Full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| restartNow | Integer | No | Perform a php restart as part of the action to apply the new settings, defaults to true |

**Expected Response Code:**

200 OK – Returned for performing the disable action and completing immediately. This can happen either because no restart is performed or because no action is needed

202 Accepted – Returned for performing the disable action and performing a restart

**Response Format:** The new code tracing directive's state

**Possible Action Specific Error Codes:**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 500 | internalServerError | The code-tracing action failed |

**Example**

POST /ZendServerManager/Api/codetracingDisable

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>codetracingDisable</method>

</requestData>

<responseData>

<codeTracingStatus>

<componentStatus>Inactive</componentStatus>

<alwaysDump>Off</alwaysDump>

<traceEnabled>Off</traceEnabled>

<awaitsRestart>0</awaitsRestart>

</codeTracingStatus>

</responseData>

<zendServerAPIResponse>

### codetracingEnable

**Description:** Enable code-tracing component and two directives necessary for creating tracing dumps

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| restartNow | Integer | No | Perform a php restart as part of the action to apply the new settings, defaults to true |

**Expected Response Code:**

200 OK – Returned for performing the enable action and completing immediately. This can happen either because no restart is performed or because no action is needed

202 Accepted – Returned for performing the enable action and performing a restart

**Response Format:** The new code tracing directive's state

**Possible Action Specific Error Codes:**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 500 | internalServerError | The code-tracing action failed |

**Example**

POST /ZendServerManager/Api/codetracingEnable

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>codetracingEnable</method>

</requestData>

<responseData>

<codeTracingStatus>

<componentStatus>Active</componentStatus>

<alwaysDump>On</alwaysDump>

<traceEnabled>On</traceEnabled>

<awaitsRestart>0</awaitsRestart>

</codeTracingStatus>

</responseData>

<zendServerAPIResponse>

### codetracingIsEnabled

**Description:** check if the directives zend\_codetracing.always\_dump, zend\_codetracing.trace\_enabled are set and if the code-tracing component is active

**Required Permissions:** read-only

**HTTP method:** GET

**Supported by Editions:** ZS, ZSCM

**Request Parameters: None**

**Expected Response Code:** 200 OK

**Response Format:** Indication for the component's current status and

**Possible Action Specific Error Codes:**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 500 | internalServerError | The code-tracing action failed |

**Example**

GET /ZendServerManager/Api/codetracingIsEnabled

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>codetracingIsEnabled</method>

</requestData>

<responseData>

<codeTracingStatus>

<componentStatus>Active</componentStatus>

<alwaysDump>On</alwaysDump>

<traceEnabled>On</traceEnabled>

<awaitsRestart>1</awaitsRestart>

</codeTracingStatus>

</responseData>

<zendServerAPIResponse>

### codetracingCreate

**Description:** Create a new code-tracing entry

**Required Permissions:** Full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| url | string | Yes | URLencoded URL to call the code tracing request from |

**Expected Response Code:** 200 OK

**Response Format:** A code-tracing entry with full details or an error message explaining the failure

**Possible Action Specific Error Codes:**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 500 | internalServerError | The code-tracing action failed |

**Example**

POST /ZendServerManager/Api/codetracingCreate

url=http%3A%2F%2Flocalhost%2Ftest.php

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>codetracingCreate</method>

</requestData>

<responseData>

<codeTrace>

<id>1.123.5</id>

<date>123412341234</date>

<url>http://localhost/test.php</url>

<createdBy>Monitor Event</createdBy>

<fileSize>12341234</fileSize>

<applicationId>1</applicationId>

</codeTrace>

</responseData>

<zendServerAPIResponse>

### codetracingDelete

**Description:** Delete a code-tracing file entry

**Required Permissions:** Full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| traceFile | string | Yes | Trace file identifier |

**Expected Response Code:** 200 OK

**Response Format:** Details of the trace-file entry that was deleted or an error message if the operation failed

**Possible Action Specific Error Codes:**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchTrace | The requested trace could not be found |
| 500 | internalServerError | The code-tracing delete action failed |

**Example**

Request:

POST /ZendServerManager/Api/codetracingDelete

traceFile=1.123.5

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>codetracingDelete</method>

</requestData>

<responseData>

<codeTrace>

<id>1.123.5</id>

<date>123412341234</date>

<url>http://localhost/test.php</url>

<createdBy>Monitor Event</createdBy>

<fileSize>12341234</fileSize>

<applicationId>1</applicationId>

</codeTrace>

</responseData>

<zendServerAPIResponse>

### codetracingList

**Description:** Retrieve a list of code-tracing files available for download using codetracingDownloadTraceFile

**Required Permissions:** full

**HTTP method:** GET

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| applications | Array | No | List of application IDs. If specified, code-tracing entries will be returned for these applications only. Default: all applications |
| limit | Integer | No | Row limit to retrieve, defaults to value defined in zend-user-user.ini |
| offset | Integer | No | The page offset to be displayed, defaults to 0 |
| orderBy | String | No | Column to sort the result by (Id, Date, Url, CreatedBy, Filesize), defaults to Date |
| direction | String | No | Sorting direction , defaults to Desc |

**Expected Response Code:** 200 OK

**Response Format:** Response successful message or error message

**Possible Action Specific Error Codes:** This action has no specific error codes

**Example**

Request:

GET /ZendServerManager/Api/codetracingList?applications[]=1&applications[]=2&limit=10&offset=0&orderBy=Date&direction=Desc

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>codetracingList</method>

</requestData>

<responseData>

<codeTracingList>

<codeTrace>

<id>1.256.3</id>

<date>123412341234</date>

<url>http://localhost/test.php</url>

<createdBy>Monitor Event</createdBy>

<fileSize>12341234</fileSize>

<applicationId>1</applicationId>

</codeTrace>

<codeTrace>

<id>1.256.3</id>

<date>123412341234</date>

<url>http://localhost/test.php</url>

<createdBy>Monitor Event</createdBy>

<fileSize>12341234</fileSize>

<applicationId>2</applicationId>

</codeTrace>

</codeTracingList>

</responseData>

<zendServerAPIResponse>

### codetracingDownloadTraceFile

**Description:** Download the amf file specified by codetracing identifier

**Required Permissions:** full

**HTTP method:** GET

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| traceFile | string | Yes | Trace file identifier. Note that a codetracing identifier is provided as part of the monitorGetRequestSummary xml response |

**Expected Response Code:** 200 OK

**Response Format:** a successful call to the monitorDownloadAmf method will result in an HTTP response with the amf file in the response body.

The content type for the amf file is “application/x-amf”. In addition, the response will include a “Content-disposition” header specifying a suggested file name for the file.

This is different from most Web API calls where the content type is expected to be “application/vnd.zend.serverpi+xml; version=…” and the response body payload is expected to be in XML format.

In case of error, a regular error response will be returned, containing an <errorData> element as defined for other Web API methods.

**Possible Action Specific Error Codes:**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchTrace | The requested trace could not be found |

**Example**

Request (headers removed for clarity):

GET /ZendServerManager/Api/monitorDownloadAmf?amf=10.123.4

Response (not all headers are shown):

HTTP/1.0 200 OK  
Content-type: application/x-amf  
Content-disposition: attachment;   
 filename=”10.123.4.amf”  
  
[...binary data follows...]

## Monitor Methods

Monitor methods will be available only from Api version 1.2 and will be included only in Zend Server running Apache and Zend Server Cluster Manager on linux.

Note that some of the Monitor Methods were stealthily added in Api version 1.1 however these were not officially supported in version 1.1

### monitorGetRequestSummary

**Description:** Retrieve information about a particular request's events and code tracing. The requestUid identifier is provided in a cookie that is set in the response to the particular request.

This API action is designed to be used with the new Studio browser toolbar.

**Required Permissions:** Read-Only

**HTTP method:** GET

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| requestUid | string | Yes | The request identifier, obtained from response cookie |

**Expected Response Code:** 200 OK

**Response Format:** Response successful message or error message

**Possible Action Specific Error Codes:** This action has no specific error codes

**Example**

Request:

GET /ZendServerManager/Api/monitorGetRequestSummary?requestUid=3AFD7433445593C54177E2A6BA60933B

Content-type: application/x-www-form-urlencoded

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>monitorGetRequestSummary</method>

</requestData>

<responseData>

<requestSummary>

<eventsCount>1</eventsCount>

<codeTracing>5.7002.1</codeTracing>  
 <events>

<event>

<type>PHP Error</type>

<description>.....</description>

<superGlobals>

<get>

<parameter>

<name>all</name>

<value>.....</value>

</parameter>

<parameter>

<name>php\_warn</name>

<value>1</value>

</parameter>

.

.

.

</get>

<post></post>

<cookie>

<parameter>

<name>ZDEDebuggerPresent</name>

<value>.....</value>

</parameter>

.

.

</cookie>

<session></session>

<server>

<parameter>

<name>HTTP\_USER\_AGENT</name>

<value>Wget/1.12 (linux-gnu)</value>

</parameter>

<parameter>

<name>HTTP\_ACCEPT</name>

<value>\*/\*</value>

</parameter>

.

.

.

.

<parameter>

<name>REQUEST\_TIME</name>

<value>1315396868</value>

</parameter>

</server>

</superGlobals>

<debugUrl>...</debugUrl>

<severity>normal</severity>

<backtrace>

<step>

<number>0</number>

<object></object>

<class></class>

<function>bt\_generator</function>

<file>/var/www/test.php</file>

<line>293</line>

</step>

</backtrace>

</event>

</events>

</requestSummary>

</responseData>

</zendServerAPIResponse>

### monitorDownloadTraceFile (codetracingDownloadTraceFile)

### MonitorStartDebug (studioStartDebug)

### monitorIssuesListByFilter

**Description:** Retrieve a list of monitor issues according to a set of filter options. This WebAPI method may also accept ordering details and paging limits

**Required Permissions:** read-only

**HTTP method:** GET

**Supported by Editions:** ZS, ZSCM

### monitorIssuesListByPredefinedFilter

**Description:** Retrieve a list of monitor issues according to a preset filter identifier. The filter identifier is shared with the UI's predefined filters. This WebAPI method may also accept ordering details and paging limits

The response is a list of issue elements with their general details and event-groups identifiers

**Required Permissions:** read-only

**HTTP method:** GET

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| filterId | string | Yes | The predefined filter's id. Can be the filter's “name” or the actual identifier randomly created by the system. This parameter is case-sensitive |
| limit | integer | No | The number of rows to retrieve. Default lists all events up to an arbitrary limit set by the system |
| offset | integer | No | A paging offset to begin the issues list from. Default is 0 |
| order | string | No | Column identifier for sorting the result set (id, repeats, date, eventType, fullUrl, severity, status). Default is date |
| direction | string | No | Sorting direction: Ascending or Descending. Default is Descending |

**Expected Response Code:** 200 OK

**Response Format:** Response successful message or error message

**Possible Action Specific Error Codes:**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchFilterId | The requested filter does not exist |

**Example**

Request:

GET /ZendServerManager/Api/monitorIssuesListByPredefinedFilter?filterId=1&limit=20&offset=0&order=lastOccurance&direction=DESC

Content-type: application/x-www-form-urlencoded

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>monitorIssuesListByPredefinedFilter</method>

</requestData>

<responseData>

<issues>

<issue>

<id>11</id>

<rule>PHP Error</rule>

<count>1</count>

<lastOccurance>123412341234</lastOccurance>

<severity>Warning</severity>

<status>Open</status>

<generalDetails>

<url>http://localhost/test.php</url>

<sourceFile>/var/www/test.php</sourceFile>

<sourceLine>302</sourceLine>

<function>fopen</function>

<aggregationHint>123412341234</aggregationHint>

<errorString>Permission Denied</errorString>

<errorType>E\_WARNING</errorType>

</generalDetails>

</issue>

</issues>

</responseData>

</zendServerAPIResponse>

### monitorGetIssueDetails

**Description:** Retrieve an issue's details according to the issueId passed as a parameter. Additional information about event groups is also displayed

The response is a list of issue elements with their general details and event-groups identifiers

**Required Permissions:** Read-Only

**HTTP method:** GET

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| issueId | string | Yes | The predefined filter's id. Can be the filter's “name” or the actual identifier randomly created by the system. This parameter is case-sensitive |

**Expected Response Code:** 200 OK

**Response Format:** Response successful message or error message

**Possible Action Specific Error Codes:**

|  |  |  |
| --- | --- | --- |
| HTTP Code | Error Code | Description |
| 404 | noSuchIssue | The requested issue does not exist |

**Example**

Request:

GET /ZendServerManager/Api/monitorIssueDetails?issueId=1

Content-type: application/x-www-form-urlencoded

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>monitorIssueDetails</method>

</requestData>

<responseData>

<issueDetails>

<issue>

<id>11</id>

<rule>PHP Error</rule>

<count>1</count>

<lastOccurance>123412341234</lastOccurance>

<severity>Warning</severity>

<status>Open</status>

<generalDetails>

<url>http://localhost/test.php</url>

<sourceFile>/var/www/test.php</sourceFile>

<sourceLine>302</sourceLine>

<function>fopen</function>

<aggregationHint>123412341234</aggregationHint>

<errorString>Permission Denied</errorString>

<errorType>E\_WARNING</errorType>

</generalDetails>

<routeDetails>

<routeDetail>

<key>controller</key>

<value>test</value>

<routeDetail>

</routeDetails>

</issue>

<eventsGroups>

<eventsGroup>

<eventsGroupId>134</eventsGroupId>

<eventsCount>1</eventsCount>

<startTime>20-Sep-2011 18:45</startTime>

<serverId>0</serverId>

<class></class>

<userData><![CDATA[]]></class>

<javaBacktrace><![CDATA[]]><javaBacktrace>

<execTime>0</execTime>

<avgExecTime>0</avgExecTime>

<memUsage>0</memUsage>

<avgMemUsage>0</avgMemUsage>

<avgOutputSize>0</avgOutputSize>

<load>0</load>

</eventsGroup>

</eventsGroups>

</issueDetails>

</responseData>

</zendServerAPIResponse>

### monitorGetEventGroupDetails

**Description:** Retrieve an events list object identified by an events-group identifier. The events-group identifier is retrieved from an Issue element's data

The response is a list of all event elements in the group and their full details

**Required Permissions:** Read-Only

**HTTP method:** GET

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| issueId | Integer | Yes | Issue identifier, provided in the issue element |
| eventsGroupId | Integer | Yes | Event group identifier, provided in the issue element |

**Expected Response Code:** 200 OK

**Response Format:** Response successful message or error message

**Possible Action Specific Error Codes:** This action has no specific error codes

**Example**

Request:

GET /ZendServerManager/Api/monitorGetEventGroupDetails?issueId=1&eventsGroupId=2

Content-type: application/x-www-form-urlencoded

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>monitorGetEventsDetails</method>

</requestData>

<responseData>

<eventsGroupDetails>

<issueId>1</issueId>

<eventsGroup>

<eventsGroupId>134</eventsGroupId>

<eventsCount>1</eventsCount>

<startTime>20-Sep-2011 18:45</startTime>

<serverId>0</serverId>

<class></class>

<userData><![CDATA[]]></class>

<javaBacktrace><![CDATA[]]><javaBacktrace>

<execTime>0</execTime>

<avgExecTime>0</avgExecTime>

<memUsage>0</memUsage>

<avgMemUsage>0</avgMemUsage>

<avgOutputSize>0</avgOutputSize>

<load>0</load>

</eventsGroup>

<event>

<type>PHP Error</type>

<description>.....</description>

<superGlobals>

<get>

<parameter>

<name>all</name>

<value>.....</value>

</parameter>

<parameter>

<name>php\_warn</name>

<value>1</value>

</parameter>

.

.

.

</get>

<post></post>

<cookie>

<parameter>

<name>ZDEDebuggerPresent</name>

<value>.....</value>

</parameter>

.

.

</cookie>

<session></session>

<server>

<parameter>

<name>HTTP\_USER\_AGENT</name>

<value>Wget/1.12 (linux-gnu)</value>

</parameter>

<parameter>

<name>HTTP\_ACCEPT</name>

<value>\*/\*</value>

</parameter>

.

.

.

.

<parameter>

<name>REQUEST\_TIME</name>

<value>1315396868</value>

</parameter>

</server>

</superGlobals>

<debugUrl>...</debugUrl>

<severity>normal</severity>

<backtrace>

<step>

<number>0</number>

<object></object>

<class></class>

<function>bt\_generator</function>

<file>/var/www/test.php</file> <line>293</line>

</step>

</backtrace>

</event>

<codeTracing><![CDATA[10.2.555]]></codeTracing>

</eventsGroupDetails>

</responseData>

</zendServerAPIResponse>

### monitorExportIssueByEventsGroup

**Description:** Export an archive containing all of the issue's information, event groups and code tracing if available, ready for consumption by Zend Studio

The response is a binary payload

**Required Permissions:** Read-Only

**HTTP method:** GET

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| eventsGroupId | string | Yes | The issue event group identifier |

Request (headers removed for clarity):

GET /ZendServerManager/Api/monitorExportIssueByEventsGroup?eventsGroupId=2

Response (not all headers are shown):

HTTP/1.0 200 OK  
Content-type: application/vnd.zend.eventexport  
Content-disposition: attachment;   
 filename=”Severe\_Slow\_Function\_Execution-1-2-20110921.zsf”  
  
[...binary data follows...]

### monitorChangeIssueStatus

**Description:** Modify an Issue's status code based on an Issue's Id and a status code

Response is an issue element's updated details

**Required Permissions:** Full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

**Response Format:** Up-to-date issue element or an error message explaining the problem

**Possible Action Specific Error Codes:** This action has no specific error codes

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| issueId | string | Yes | The issue identifier |
| newStatus | String | Yes | The new status to set: Open | Closed | Ignored |

Request (headers removed for clarity):

POST /ZendServerManager/Api/monitorChangeIssueStatus

Content-type: application/x-www-form-urlencoded

issueId=1&newStatus=Closed

Response (not all headers are shown):

HTTP/1.0 200 OK

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>monitorChangeIssueStatus</method>

</requestData>

<responseData>

<issue>

<id>11</id>

<rule>PHP Error</rule>

<count>1</count>

<lastOccurance>123412341234</lastOccurance>

<severity>Warning</severity>

<status>Closed</status>

<generalDetails>

<url>http://localhost/test.php</url>

<sourceFile>/var/www/test.php</sourceFile>

<sourceLine>302</sourceLine>

<function>fopen</function>

<aggregationHint>123412341234</aggregationHint>

<errorString>Permission Denied</errorString>

<errorType>E\_WARNING</errorType>

</generalDetails>

<routeDetails>

<routeDetail>

<key>controller</key>

<value>test</value>

<routeDetail>

</routeDetails>

<eventsCount>1</eventsCount>

<eventsGroups>

<eventsGroup>

<eventsGroupId>4</eventsGroupId>

<eventsCount>1</eventsCount>

<startTime>123412341234</startTime>

<serverId>1</serverId>

</eventsGroup>

</eventsGroups>

</issue>

</responseData>

</zendServerAPIResponse>

## Studio Integration Methods

### StudioStartDebug

**Description:** Start a debug session for specific issue

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Req. | Description |
| eventsGroupId | string | Yes | The issue event group identifier |
| noRemote | boolean | No | Use server's own local files for debug display. Default: true. Setting to false will use local files from studio if available |

**Expected Response Code:**

200 OK – For execution as a synchronous debug request

**Response Format:** Response successful message or error message

**Possible Action Specific Error Codes:** This action has no specific error codes

**Example**

Request:

POST /ZendServerManager/Api/studioStartDebug

Content-type: application/x-www-form-urlencoded

eventsGroupId=36

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>studioStartDebug</method>

</requestData>

<responseData>

<debugRequest>

<success>1</success>

<message>Debug session completed successfully</message>

</debugRequest>

</responseData>

</zendServerAPIResponse>

### studioStartProfile

**Description:** Start a profiling session with Zend Studio's integration using an event-group's identifier

This action has the peculiar behavior of being synchronous and hanging until the profiling session is completed.

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

**Request Parameters:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Type | Req. | Description |  |
| eventsGroupId | string | Yes | The issue event group identifier | |

**Expected Response Code:**

200 OK – For execution as a synchronous profile request

**Response Format:** Response successful message or error message

**Possible Action Specific Error Codes:** This action has no specific error codes

**Example**

Request:

POST /ZendServerManager/Api/studioStartProfile

Content-type: application/x-www-form-urlencoded

eventsGroupId=36

Response body:

<?xml version="1.0" encoding="UTF-8"?>

<zendServerAPIResponse xmlns=*"http://www.zend.com/server/api/1.1"*>

<requestData>

<apiKeyName><![CDATA[Admin]]></apiKeyName>

<method>studioStartProfile</method>

</requestData>

<responseData>

<profileRequest>

<success>1</success>

<message>Debug session completed successfully</message>

</profileRequest>

</responseData>

</zendServerAPIResponse>

### studioShowFile

**Description:** Display a file from an event-group's backtrace, according to event-group's identifier and the backtrace row needed

**Required Permissions:** full

**HTTP method:** POST

**Supported by Editions:** ZS, ZSCM

# Open Issues

Is studioShowFile action actually needed?

Should monitorIssuesListByFilter be implemented? Avoided for now.

Studio integration actions in asynchronous mode by using job queue?

Corporate Headquarters: Zend Technologies, Inc. 19200 Stevens Creek Blvd. Cupertino, CA 95014, USA · Tel +1-408-253-8800 · Fax +1-408-253-8801

Central Europe: (Germany, Austria, Switzerland) Zend Technologies GmbH Bayerstrasse 83, 80335 Munich, Germany · Tel +49-89-516199-0 · Fax +49-89-516199-20

International: Zend Technologies Ltd. 12 Abba Hillel Street, Ramat Gan, Israel 52506 · Tel +972-3-753-9500 · Fax +972-3-613-9671

France: Zend Technologies SARL, 5 Rue de Rome, ZAC de Nanteuil, 93110 Rosny-sous-Bois, France · Tel +33 1 4855 0200 · Fax +33 1 4812 3132

Italy: Zend Technologies, Largo Richini 6, 20122 Milano, Italy · Tel +39 02 5821 5832 · Fax +39 02 5821 5400

© 2009 Zend Corporation. Zend is a registered trademark of Zend Technologies Ltd.   
All other trademarks are the property of their respective owners. [www.zend.com](http://www.zend.com/)

0106-SDD-T-20111128-R5-EN