Progress®
Kemp® LoadMaster®

Technical Note RESTful API Programmer Guide

Generated on: 18 December 2024

Copyright

Visit the following page online to see Progress Software Corporation's current Product Documentation Copyright Notice/Trademark Legend: Product Documentation Copyright Notice & Trademarks | Progress

Table of Contents

Chapter 1: Introduction 6

Document Purpose 6

Intended Audience 7

Prerequisites 7

Chapter 2: API Examples 8

Virtual Service Examples 9

Add a Virtual Service with Default Values 9

Add a Virtual Service and Set Values 12

List the Virtual Service Details 14

Modify a Virtual Service 19

Add a Real Server to a Virtual Service 22

Modify a Real Server 22

Delete a Real Server 23

Change Real Server Health Check Type 28

Create a SubVS 33

Modify a SubVS 34

Delete a SubVS 35

Enable the Web Application Firewall (WAF) on a Virtual Service 37

Add a WAF Rule to a Virtual Service 39

Delete a Virtual Service 40

Add Client Side Single Sign On (SSO) Configuration 43

Display a Client Side SSO Configuration 43

Modify a Client Side SSO Configuration 44

Delete a Client Side SSO Configuration 46

Add a Custom SSO Image Set 47

Delete a Custom SSO Image Set 48

List WAF Custom Rules 49
Upload a WAF Custom Rule 51
Download a WAF Custom Rule 51
Delete a WAF Custom Rule 51
Statistics 52
Show stats 52
Rules & Checking 54
Add a content rule to the system 54
Modify a Content Rule 55
Delete Rule 56
List the Content Rules Assigned to a Virtual Service 57
Add a Content Rule to a Virtual Service 60
Unassign a Content Rule from a Virtual Service 60
Set Retry Interval 62
Certificates 62
Add a Certificate 62
Delete a Certificate 63
Backup a Certificate 63
Restore Certificates 63
GEO 64
Add an FQDN 64
Modify an FQDN 65
Add an IP Address to an FQDN 66
Modify the IP Address of an FQDN 67
Delete an IP Address from an FQDN 68
Delete an FQDN 70
Add a Cluster 71
Modify a Cluster 72
Modify a Cluster Location 73
Delete a Cluster 73
Modify the GEO Miscellaneous Parameters 74
Add an IP Range 75
Modify an IP Location 76

Delete an IP Range 77

Chapter 3: Basic Scripting Examples 79

Shell Script Example to Add Multiple Virtual Services with Default Settings 80

Shell Script Example to Delete Multiple Basic Virtual Services 80

Shell Script Example to Modify the Same Parameter on Multiple Virtual Services 80

Shell Script Example to Add Multiple FQDNs 81

Shell Script Example to Add Multiple IP Addresses to an FQDN 81

Shell Script Example to Extract the SubVSs Index from one Command and Use in Another 81

Chapter 4: Microsoft Exchange 2013 HTTPS Offloaded Example 83

Chapter 5: References 92

1

Introduction

Progress Kemp leads the industry in driving the price/performance value proposition for application delivery and load balancing to levels that our customers can afford. Our products' versatile and powerful architecture provide the highest value, while enabling our customers to optimize businesses that rely on Internet-based infrastructure to conduct business with their customers, employees and partners. Progress Kemp products optimize web and application infrastructure as defined by High Availability (HA), high-performance, flexible scalability, security and ease of management. They maximize the total cost-of-ownership for web infrastructure, while enabling flexible and comprehensive deployment options.

Related Links

For details, see the following topics:

- Document Purpose
- Intended Audience
- Prerequisites

Document Purpose

This document provides examples of how some RESTful API commands (which are listed and described in the main RESTful API Interface Description document) might be used, using example data. This document is intended to complement the main RESTful API description document.

This document contains examples in APIv1 format. In LoadMaster firmware version 7.2.50, functionality was added which allows you to specify API requests as a POST of a JSON object and receive a JSON-based

API payload response. Session Management must be enabled on the LoadMaster (**Certificates & Security > Admin WUI Access > Enable Session Management**) to use the JSON-based input and output.

For further details on the JSON-based input and output, refer to the RESTful APIv2 documentation document.

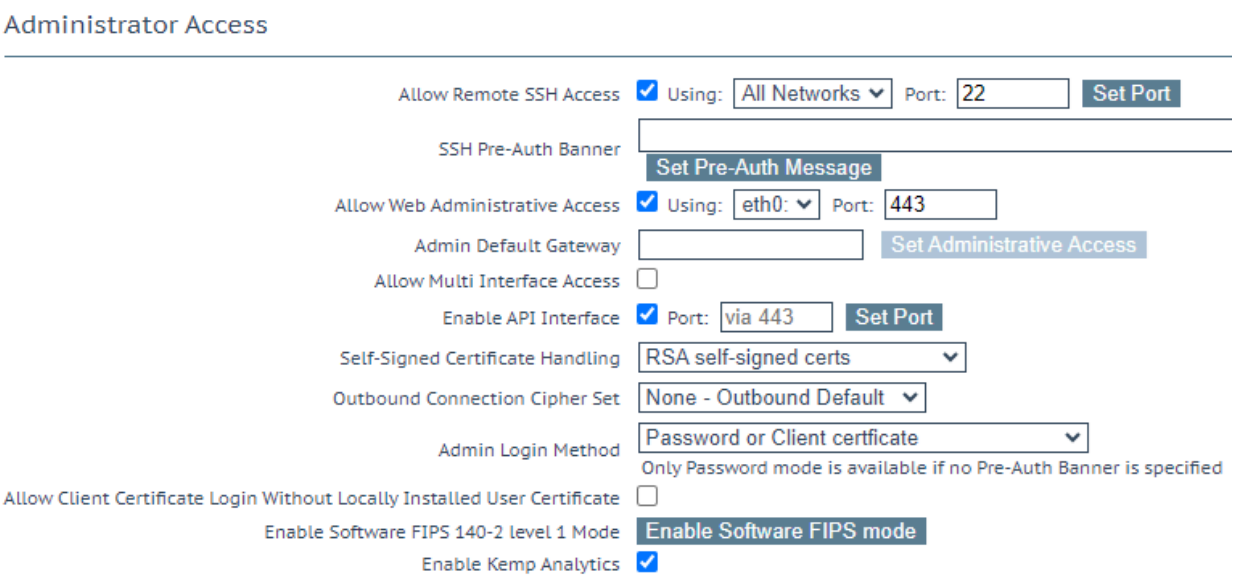
Intended Audience

This document is intended to help anyone who wishes to configure the LoadMaster using the RESTful API.

Prerequisites

Before attempting any of the examples in this document, ensure that the API interface is enabled on the LoadMaster. To do this, follow the steps below:

1. In the main menu of the LoadMaster Web User Interface (WUI), go to **Certificates & Security > Remote Access**.



2. Select the **Enable API Interface** check box.

2

API Examples

The LoadMaster RESTful API can be used in conjunction with many scripting methods and applications to allow users and applications to directly access the LoadMaster.

The examples provided in this document use the cURL tool (a Linux command line tool for transferring data with URL syntax) to demonstrate the structure of the API URL.

The syntax for using cURL is as follows:

```
curl -o <output XML> -k -silent -u <user>:<Password> <URL>
```

Note: If the username or password has a space in them, please surround <user> and <password> with double quotes, for example “<user>”:”<password>”.

- -o: output to the following file rather than just showing the results on screen
- <output XML>: the filename to store the response XML from the LoadMaster.
- -k: ignore the SSL certification on the LoadMaster
- -silent: do not output anything on screen
- -u: use the following login information
- <user>: the username
- <Password>: the user password
- <URL>: the query to send

The syntax for xpath is as follows:

```
xpath -q -e <XML path> <XML file>
```

- -q do not output anything informational, only output the raw data
- <XML path> the path inside the XML structure to look for the raw data
- <XML file> the XML file to parse

Instead of printing the data output from the XML parsing on the screen, the output can be assigned to a variable as follows:

```
var=“expr xpath -q -e <XML path> <XML file>”
```

It is possible to pass parameters in a cURL command via the HTTP POST method. This is a more secure method because the parameters are not passed in the URL.

Examples of a get and set command using this method are below:

```
curl --data "param=motd" -k
https://<user>:<password>@<LoadMasterIPAddress>/access/get

curl --data "param=motd&value=<value>" -k https://
<user>:<password>@<LoadMasterIPAddress>/access/set
```

The examples in this document include a typical XML response which is generated by the command.

Some examples show a “before” situation and an “after” situation where the object has been changed.

Items in the response which have changed as a result of the command are highlighted in yellow.

Related Links

For details, see the following topics:

- Virtual Service Examples
- Statistics
- Rules & Checking
- Certificates
- GEO

Virtual Service Examples

Refer to the following sections for examples on running the LoadMaster RESTful API commands relating to Virtual Services.

Add a Virtual Service with Default Values

The example below shows a command to the LoadMaster at the IP address **20.200.25.100** to create a Virtual

Service with the IP address **20.200.25.250** on **Port 80** for the **TCP** protocol. Creating a Virtual Service will enable a default set of options for the Virtual Service, as can be seen in the response below. For example, the Edge Security Pack (ESP) is disabled and the Web Application Firewall (WAF) (**Intercept**) is disabled.

```
curl -k "https://bal:1fourall@20.200.25.100/access/
addvs?vs=20.200.25.250&port=80&prot=tcp"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Status>Down</Status>

<Index>15</Index>

<VSAddress>20.200.25.250</VSAddress>

<VSPort>80</VSPort>

<Enable>Y</Enable>

<SSLReverse>N</SSLReverse>

<SSLReencrypt>N</SSLReencrypt>

<Intercept>N</Intercept>

<InterceptOpts><Opt>opnormal</Opt>

<Opt>auditnone</Opt>

<Opt>reqdatadisable</Opt>

<Opt>resdatadisable</Opt>

</InterceptOpts><AlertThreshold>0</AlertThreshold>

<Transactionlimit>0</Transactionlimit>

<Transparent>Y</Transparent>

<ServerInit>0</ServerInit>

<StartTLSMode>0</StartTLSMode>

<Idletime>0</Idletime>

<Cache>N</Cache>
```

```
<Compress>N</Compress>

<Verify>0</Verify>

<UseforSnat>N</UseforSnat>

<ForceL7>Y</ForceL7>

<ClientCert>0</ClientCert>

<ErrorCode>0</ErrorCode>

<CheckUse1.1>N</CheckUse1.1>

<MatchLen>0</MatchLen>

<CheckUseGet>0</CheckUseGet>

<SSLRewrite>0</SSLRewrite>

<VStype>http</VStype>

<FollowVSID>0</FollowVSID>

<Protocol>tcp</Protocol>

<Schedule>rr</Schedule>

<CheckType>http</CheckType>

<PersistTimeout>0</PersistTimeout>

<CheckPort>0</CheckPort>

<NRules>0</NRules>

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>0</NPreProcessRules>

<EspEnabled>N</EspEnabled>

<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

<MasterVS>0</MasterVS>
```

```
<MasterVSID>0</MasterVSID>

<AddVia>0</AddVia>

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<OCSPVerify>N</OCSPVerify>

<NumberOfRSs>0</NumberOfRSs>

</Data>

</Success>

</Response>
```

Add a Virtual Service and Set Values

The example below shows a command which adds a Virtual Service with the IP address **20.200.25.250** on **Port 80** for the **TCP** protocol. This command also sets the **nickname** of the new Virtual Service to **testvs** and disabled the **Transparency** option.:

```
 curl -k "https://bal:1fourall@20.200.25.100/access/
addvs?vs=20.200.25.250&port=80&prot=tcp&nickname=testvs&transparent=N"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Status>Down</Status>

<Index>15</Index>

<VSAddress>20.200.25.250</VSAddress>

<VSPort>80</VSPort>

<NickName>testvs</NickName>

<Enable>Y</Enable>

<SSLReverse>N</SSLReverse>

<SSLReencrypt>N</SSLReencrypt>
```

<Intercept>N</Intercept>

<InterceptOpts><Opt>opnormal</Opt>

<Opt>auditnone</Opt>

<Opt>reqdatadisable</Opt>

<Opt>resdatadisable</Opt>

</InterceptOpts><AlertThreshold>0</AlertThreshold>

<Transactionlimit>0</Transactionlimit>

<Transparent>N</Transparent>

<ServerInit>0</ServerInit>

<StartTLSMode>0</StartTLSMode>

<Idletime>0</Idletime>

<Cache>N</Cache>

<Compress>N</Compress>

<Verify>0</Verify>

<UseforSnat>N</UseforSnat>

<ForceL7>Y</ForceL7>

<ClientCert>0</ClientCert>

<ErrorCode>0</ErrorCode>

<CheckUse1.1>N</CheckUse1.1>

<MatchLen>0</MatchLen>

<CheckUseGet>0</CheckUseGet>

<SSLRewrite>0</SSLRewrite>

<VStype>http</VStype>

<FollowVSID>0</FollowVSID>

<Protocol>tcp</Protocol>

```
<Schedule>rr</Schedule>

<CheckType>http</CheckType>

<PersistTimeout>0</PersistTimeout>

<CheckPort>0</CheckPort>

<NRules>0</NRules>

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>0</NPreProcessRules>

<EspEnabled>N</EspEnabled>

<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

<MasterVS>0</MasterVS>

<MasterVSID>0</MasterVSID>

<AddVia>0</AddVia>

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<OCSPVerify>N</OCSPVerify>

<NumberOfRSs>0</NumberOfRSs>

</Data>

</Success>

</Response>
```

List the Virtual Service Details

When referencing a Virtual Service, the IP address, port and protocol can be used. The VS index can also be used, which replaces those three parameters.

To get the VS index (**Index** parameter), and to list the settings for the existing Virtual Services, run the **ListVS** command:

```
curl -k https://bal:1fourall@20.200.25.100/access/listvs
```

Response:

```
<Response stat="200" code="ok">
<Success>
<Data>
<VS>
<Status>Down</Status>
<Index>1</Index>
<VSAddress>20.200.25.250</VSAddress>
<VSPort>80</VSPort>
<NickName>testvs</NickName>
<Enable>Y</Enable>
<SSLReverse>N</SSLReverse>
<SSLReencrypt>N</SSLReencrypt>
<Intercept>N</Intercept>
<InterceptOpts>
<Opt>opnormal</Opt>
<Opt>auditnone</Opt>
<Opt>reqdatadisable</Opt>
<Opt>resdatadisable</Opt>
</InterceptOpts>
<AlertThreshold>0</AlertThreshold>
<Transactionlimit>0</Transactionlimit>
<Transparent>N</Transparent>
<ServerInit>0</ServerInit>
```

```
<StartTLSMode>0</StartTLSMode>

<Idletime>0</Idletime>

<Cache>N</Cache>

<Compress>N</Compress>

<Verify>0</Verify>

<UseforSnat>N</UseforSnat>

<ForceL7>Y</ForceL7>

<ClientCert>0</ClientCert>

<ErrorCode>0</ErrorCode>

<CheckUse1.1>N</CheckUse1.1>

<MatchLen>0</MatchLen>

<CheckUseGet>0</CheckUseGet>

<SSLRewrite>0</SSLRewrite>

<VStype>http</VStype>

<FollowVSID>0</FollowVSID>

<Protocol>tcp</Protocol>

<Schedule>rr</Schedule>

<CheckType>http</CheckType>

<PersistTimeout>0</PersistTimeout>

<CheckPort>0</CheckPort>

<NRules>0</NRules>

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>0</NPreProcessRules>

<EspEnabled>N</EspEnabled>
```

<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

<MasterVS>0</MasterVS>

<MasterVSID>0</MasterVSID>

<AddVia>0</AddVia>

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<OCSPVerify>N</OCSPVerify>

<NumberOfRSs>0</NumberOfRSs>

</VS>

<VS>

<Status>Down</Status>

<Index>2</Index>

<VSAddress>20.200.25.250</VSAddress>

<VSPort>80</VSPort>

<NickName>newname</NickName>

<Enable>Y</Enable>

<SSLReverse>N</SSLReverse>

<SSLReencrypt>N</SSLReencrypt>

<Intercept>N</Intercept>

<InterceptOpts>

<Opt>opnormal</Opt>

<Opt>auditnone</Opt>

<Opt>reqdatadisable</Opt>

<Opt>resdatadisable</Opt>

```
</InterceptOpts>

<AlertThreshold>0</AlertThreshold>

<Transactionlimit>0</Transactionlimit>

<Transparent>Y</Transparent>

<ServerInit>0</ServerInit>

<StartTLSMode>0</StartTLSMode>

<Idletime>0</Idletime>

<Cache>N</Cache>

<Compress>N</Compress>

<Verify>0</Verify>

<UseforSnat>N</UseforSnat>

<ForceL7>Y</ForceL7>

<ClientCert>0</ClientCert>

<ErrorCode>0</ErrorCode>

<CheckUse1.1>N</CheckUse1.1>

<MatchLen>0</MatchLen>

<CheckUseGet>0</CheckUseGet>

<SSLRewrite>0</SSLRewrite>

<VStype>http</VStype>

<FollowVSID>0</FollowVSID>

<Protocol>tcp</Protocol>

<Schedule>rr</Schedule>

<CheckType>http</CheckType>

<PersistTimeout>0</PersistTimeout>

<CheckPort>0</CheckPort>
```

```
<NRules>0</NRules>

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>0</NPreProcessRules>

<EspEnabled>N</EspEnabled>

<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

<MasterVS>0</MasterVS>

<MasterVSID>0</MasterVSID>

<AddVia>0</AddVia>

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<OCSPVerify>N</OCSPVerify>

<NumberOfRSs>0</NumberOfRSs>

</VS>

</Data>

</Success>

</Response>
```

Modify a Virtual Service

When referencing a Virtual Service, the IP address, port and protocol can be used. The VS index can also be used, which replaces those three parameters. To find out how to get the VS index, refer to the List the Virtual Service Details section.

The following example command modifies the existing **20.200.25.250** Virtual Service to set the **nickname** to **newname** and enable transparency:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
modvs?vs=20.200.25.250&port=80&prot=tcp&nickname=newname&transparent=Y"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Response stat="200" code="ok">
<Success><Data><Status>Down</Status>
<Index>15</Index>
<VSAddress>20.200.25.250</VSAddress>
<VSPort>80</VSPort>
<NickName>newname</NickName>
<Enable>Y</Enable>
<SSLReverse>N</SSLReverse>
<SSLReencrypt>N</SSLReencrypt>
<Intercept>N</Intercept>
<InterceptOpts><Opt>opnormal</Opt>
<Opt>auditnone</Opt>
<Opt>reqdatadisable</Opt>
<Opt>resdatadisable</Opt>
</InterceptOpts><AlertThreshold>0</AlertThreshold>
<Transactionlimit>0</Transactionlimit>
<Transparent>Y</Transparent>
<ServerInit>0</ServerInit>
<StartTLSMode>0</StartTLSMode>
<Idletime>0</Idletime>
<Cache>N</Cache>
<Compress>N</Compress>
<Verify>0</Verify>
<UseforSnat>N</UseforSnat>
```

```
<ForceL7>Y</ForceL7>

<ClientCert>0</ClientCert>

<ErrorCode>0</ErrorCode>

<CheckUse1.1>N</CheckUse1.1>

<MatchLen>0</MatchLen>

<CheckUseGet>0</CheckUseGet>

<SSLRewrite>0</SSLRewrite>

<VStype>http</VStype>

<FollowVSID>0</FollowVSID>

<Protocol>tcp</Protocol>

<Schedule>rr</Schedule>

<CheckType>http</CheckType>

<PersistTimeout>0</PersistTimeout>

<CheckPort>0</CheckPort>

<NRules>0</NRules>

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>0</NPreProcessRules>

<EspEnabled>N</EspEnabled>

<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

<MasterVS>0</MasterVS>

<MasterVSID>0</MasterVSID>

<AddVia>0</AddVia>

<TlsType>N</TlsType>
```

```
<NeedHostName>N</NeedHostName>

<OCSPVerify>N</OCSPVerify>

<NumberOfRSs>0</NumberOfRSs>

</Data>

</Success>

</Response>
```

Add a Real Server to a Virtual Service

The following command adds a Real Server with the IP address **20.200.25.21** on port **80** to the existing **20.200.25.100** Virtual Service:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
addrs?vs=20.200.25.202&port=80&prot=tcp&rs=20.200.25.21&rsport=80"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Command completed ok</Success>

</Response>
```

Modify a Real Server

The following command modifies the existing **20.200.25.21** Real Server to set the **weight** to **500**:

```
https://bal:1fourall@20.200.25.100/access/
modrs?vs=20.200.25.202&port=80&prot=tcp&rs=20.200.25.21&rsport=80&weight=500
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Command completed ok</Success>

</Response>
```

Delete a Real Server

The following example command deletes the **20.200.25.21** Real Server from the **20.200.25.202** Virtual Service.

Situation before delete:

```
<Response stat="200" code="ok">

<Success>

<Data>

<VS>

<Status>Down</Status>

<Index>2</Index>

<VSAddress>20.200.25.202</VSAddress>

<VSPort>80</VSPort>

<NickName>newname</NickName>

<Enable>Y</Enable>

<SSLReverse>N</SSLReverse>

<SSLReencrypt>N</SSLReencrypt>

<Intercept>N</Intercept>

<InterceptOpts>

<Opt>opnormal</Opt>

<Opt>auditnone</Opt>

<Opt>reqdatadisable</Opt>

<Opt>resdatadisable</Opt>

</InterceptOpts>

<AlertThreshold>0</AlertThreshold>

<Transactionlimit>0</Transactionlimit>
```

```
<Transparent>Y</Transparent>

<ServerInit>0</ServerInit>

<StartTLSMode>0</StartTLSMode>

<Idletime>0</Idletime>

<Cache>N</Cache>

<Compress>N</Compress>

<Verify>0</Verify>

<UseforSnat>N</UseforSnat>

<ForceL7>Y</ForceL7>

<ClientCert>0</ClientCert>

<ErrorCode>0</ErrorCode>

<CheckUse1.1>N</CheckUse1.1>

<MatchLen>0</MatchLen>

<CheckUseGet>0</CheckUseGet>

<SSLRewrite>0</SSLRewrite>

<VStype>http</VStype>

<FollowVSID>0</FollowVSID>

<Protocol>tcp</Protocol>

<Schedule>rr</Schedule>

<CheckType>http</CheckType>

<PersistTimeout>0</PersistTimeout>

<CheckPort>0</CheckPort>

<NRules>0</NRules>

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>
```

```
<NPreProcessRules>0</NPreProcessRules>
<EspEnabled>N</EspEnabled>
<InputAuthMode>0</InputAuthMode>
<OutputAuthMode>0</OutputAuthMode>
<MasterVS>0</MasterVS>
<MasterVSID>0</MasterVSID>
<AddVia>0</AddVia>
<TlsType>N</TlsType>
<NeedHostName>N</NeedHostName>
<OCSPVerify>N</OCSPVerify>
<NumberOfRSs>1</NumberOfRSs>
<Rs>
<Status>Down</Status>
<VSIndex>2</VSIndex>
<RsIndex>1</RsIndex>
<Addr>20.200.25.21</Addr>
<Port>80</Port>
<Forward>nat</Forward>
<Weight>1000</Weight>
<Limit>0</Limit>
<Enable>Y</Enable>
</Rs>
</VS>
</Data>
</Success>
</Response>
```

```
curl -k "https://bal:1fourall@20.200.25.100/access/
delrs?vs=20.200.25.202&port=80&prot=tcp&rs=20.200.25.21&rsport=80"
```

Situation after delete:

```
<Response stat="200" code="ok">

<Success>

<Data>

<VS>

<Status>Down</Status>

<Index>2</Index>

<VSAddress>20.200.25.202</VSAddress>

<VSPort>80</VSPort>

<NickName>newname</NickName>

<Enable>Y</Enable>

<SSLReverse>N</SSLReverse>

<SSLReencrypt>N</SSLReencrypt>

<Intercept>N</Intercept>

<InterceptOpts>

<Opt>opnormal</Opt>

<Opt>auditnone</Opt>

<Opt>reqdatadisable</Opt>

<Opt>resdatadisable</Opt>

</InterceptOpts>

<AlertThreshold>0</AlertThreshold>

<Transactionlimit>0</Transactionlimit>

<Transparent>Y</Transparent>
```

```
<ServerInit>0</ServerInit>

<StartTLSMode>0</StartTLSMode>

<Idletime>0</Idletime>

<Cache>N</Cache>

<Compress>N</Compress>

<Verify>0</Verify>

<UseforSnat>N</UseforSnat>

<ForceL7>Y</ForceL7>

<ClientCert>0</ClientCert>

<ErrorCode>0</ErrorCode>

<CheckUse1.1>N</CheckUse1.1>

<MatchLen>0</MatchLen>

<CheckUseGet>0</CheckUseGet>

<SSLRewrite>0</SSLRewrite>

<VStype>http</VStype>

<FollowVSID>0</FollowVSID>

<Protocol>tcp</Protocol>

<Schedule>rr</Schedule>

<CheckType>http</CheckType>

<PersistTimeout>0</PersistTimeout>

<CheckPort>0</CheckPort>

<NRules>0</NRules>

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>0</NPreProcessRules>
```

```
<EspEnabled>N</EspEnabled>

<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

<MasterVS>0</MasterVS>

<MasterVSID>0</MasterVSID>

<AddVia>0</AddVia>

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<OCSPVerify>N</OCSPVerify>

<NumberOfRSs>0</NumberOfRSs>

</VS>

</Data>

</Success>

</Response>
```

Change Real Server Health Check Type

The following example changes the health check of the **20.200.25.200** Real Server to **none**:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
modvs?vs=20.200.25.200&port=80&prot=tcp&checktype=none"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Status>Unchecked</Status>

<Index>1</Index>

<VSAddress>20.200.25.200</VSAddress>

<VSPort>80</VSPort>
```

```
<NickName>newname</NickName>

<Enable>Y</Enable>

<SSLReverse>N</SSLReverse>

<SSLReencrypt>N</SSLReencrypt>

<Intercept>Y</Intercept>

<InterceptOpts><Opt>opblock</Opt>

<Opt>auditall</Opt>

<Opt>reqdatadisable</Opt>

<Opt>resdatadisable</Opt>

</InterceptOpts><AlertThreshold>0</AlertThreshold>

<InterceptRules>A/cpanel_attacks</InterceptRules>

<Transactionlimit>0</Transactionlimit>

<Transparent>N</Transparent>

<ServerInit>0</ServerInit>

<StartTLSMode>0</StartTLSMode>

<Idletime>0</Idletime>

<Cache>N</Cache>

<Compress>N</Compress>

<Verify>0</Verify>

<UseforSnat>N</UseforSnat>

<ForceL7>Y</ForceL7>

<ClientCert>0</ClientCert>

<ErrorCode>0</ErrorCode>

<ErrorUrl>https:%h%s</ErrorUrl>

<CheckUse1.1>N</CheckUse1.1>
```

```
<MatchLen>0</MatchLen>

<CheckUseGet>0</CheckUseGet>

<SSLRewrite>0</SSLRewrite>

<VStype>http</VStype>

<FollowVSID>2</FollowVSID>

<FollowVS>tcp/20.200.25.201:80</FollowVS>

<Protocol>tcp</Protocol>

<Schedule>rr</Schedule>

<CheckType>none</CheckType>

<PersistTimeout>360</PersistTimeout>

<Cookie>Set_Me</Cookie>

<CheckPort>44</CheckPort>

<NRules>0</NRules>

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>0</NPreProcessRules>

<EspEnabled>Y</EspEnabled>

<EspLogs>7</EspLogs>

<PermanentCookies>0</PermanentCookies>

<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

<MasterVS>0</MasterVS>

<MasterVSID>0</MasterVSID>

<AddVia>0</AddVia>

<TlsType>N</TlsType>
```

```
<NeedHostName>N</NeedHostName>

<OCSPVerify>N</OCSPVerify>

<NumberOfRSs>5</NumberOfRSs>

<Rs>

<Status>Up</Status>

<VSIndex>1</VSIndex>

<RsIndex>1</RsIndex>

<Addr>20.200.15.20</Addr>

<Port>80</Port>

<Forward>nat</Forward>

<Weight>1000</Weight>

<Limit>0</Limit>

<Enable>Y</Enable>

</Rs>

<Rs>

<Status>Up</Status>

<VSIndex>1</VSIndex>

<RsIndex>2</RsIndex>

<Addr>20.200.15.21</Addr>

<Port>80</Port>

<Forward>nat</Forward>

<Weight>1000</Weight>

<Limit>0</Limit>

<Enable>Y</Enable>

</Rs>
```

```
<Rs>
<Status>Up</Status>
<VSIndex>1</VSIndex>
<RsIndex>3</RsIndex>
<Addr>20.200.15.22</Addr>
<Port>80</Port>
<Forward>nat</Forward>
<Weight>1000</Weight>
<Limit>0</Limit>
<Enable>Y</Enable>
</Rs>
<Rs>
<Status>Up</Status>
<VSIndex>1</VSIndex>
<RsIndex>4</RsIndex>
<Addr>20.200.15.23</Addr>
<Port>80</Port>
<Forward>nat</Forward>
<Weight>1000</Weight>
<Limit>0</Limit>
<Enable>Y</Enable>
</Rs>
<Rs>
<Status>Up</Status>
<VSIndex>1</VSIndex>
```

```
<RsIndex>5</RsIndex>

<Addr>20.200.15.24</Addr>

<Port>80</Port>

<Forward>nat</Forward>

<Weight>1000</Weight>

<Limit>0</Limit>

<Enable>Y</Enable>

</Rs>

</Data>

</Success>

</Response>
```

Create a SubVS

The following command adds a SubVS to the 20.200.25.200 Virtual Service:

```
curl -k "https://bal:1fourall@10.154.25.100/access/
modvs?vs=20.200.25.200&port=80&prot=tcp&createsubvs="
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Status>Down</Status>

<Index>22</Index>

<VSAddress>20.200.25.200</VSAddress>

<VSPort>80</VSPort>

<Enable>Y</Enable>

.

.
```

```
.

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<OCSPVerify>N</OCSPVerify>

<NumberOfRSs>1</NumberOfRSs>

<SubVS>

<VSIndex>23</VSIndex>

<RsIndex>24</RsIndex>

<Name>-</Name>

<Forward>nat</Forward>

<Weight>1000</Weight>

<Limit>0</Limit>

<Enable>Y</Enable>

</SubVS>

</Data>

</Success>

</Response>
```

Modify a SubVS

The following command changes the nickname on SubVS 23 to “newnickname”:

```
curl -k "https://bal:1fourall@10.154.25.100/access/
modvs?vs=23&nickname=newnickname"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Status>Down</Status>
```

```
<Index>23</Index>

<VSPort>0</VSPort>

<NickName>newnickname</NickName>

<OCSPVerify>N</OCSPVerify>

.

.

.

<NumberOfRSs>0</NumberOfRSs>

</Data>

</Success>

</Response>
```

Delete a SubVS

The following command deletes SubVS 23.

Situation before delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Status>Down</Status>

<Index>22</Index>

<VSAddress>20.200.25.200</VSAddress>

<VSPort>80</VSPort>

<Enable>Y</Enable>

<NumberOfRSs>1</NumberOfRSs>

.

.

.
```

```
<SubVS>

<VSIndex>23</VSIndex>

<RsIndex>24</RsIndex>

<Name>nickname</Name>

<Forward>nat</Forward>

<Weight>1000</Weight>

<Limit>0</Limit>

<Enable>Y</Enable>

</SubVS>

</Data>

</Success>

</Response>

curl -k "https://bal:1fourall@10.154.25.100/access/delvs?vs=23
```

Situation after delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Status>Down</Status>

<Index>22</Index>

<VSAddress>20.200.25.200</VSAddress>

<VSPort>80</VSPort>

<Enable>Y</Enable>

.

.

.

<NumberOfRSs>0</NumberOfRSs>
```

```
</Data>

</Success>

</Response>
```

Enable the Web Application Firewall (WAF) on a Virtual Service

The following command enables WAF (**intercept** parameter) on the **20.200.25.200** Virtual Service:

```
curl -k https://bal:1fourall@20.200.25.100/access/
modvs?vs=20.200.25.200&port=80&prot=tcp&intercept=Y
```

Response:

```
<Response stat="200" code="ok">

<Success>

<Data>

<Status>Down</Status>

<Index>2</Index>

<VSAddress>20.200.25.200</VSAddress>

<VSPort>80</VSPort>

<NickName>newname</NickName>

<Enable>Y</Enable>

<SSLReverse>N</SSLReverse>

<SSLReencrypt>N</SSLReencrypt>

<Intercept>Y</Intercept>

<InterceptOpts>

<Opt>opnormal</Opt>

<Opt>auditnone</Opt>

<Opt>reqdatadisable</Opt>

<Opt>resdatadisable</Opt>
```

```
</InterceptOpts>

<AlertThreshold>0</AlertThreshold>

<Transactionlimit>0</Transactionlimit>

<Transparent>Y</Transparent>

<ServerInit>0</ServerInit>

<StartTLSMode>0</StartTLSMode>

<Idletime>0</Idletime>

<Cache>N</Cache>

<Compress>N</Compress>

<Verify>0</Verify>

<UseforSnat>N</UseforSnat>

<ForceL7>Y</ForceL7>

<ClientCert>0</ClientCert>

<ErrorCode>0</ErrorCode>

<CheckUse1.1>N</CheckUse1.1>

<MatchLen>0</MatchLen>

<CheckUseGet>0</CheckUseGet>

<SSLRewrite>0</SSLRewrite>

<VStype>http</VStype>

<FollowVSID>0</FollowVSID>

<Protocol>tcp</Protocol>

<Schedule>rr</Schedule>

<CheckType>http</CheckType>

<PersistTimeout>0</PersistTimeout>

<CheckPort>0</CheckPort>
```

```
<NRules>0</NRules>

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>0</NPreProcessRules>

<EspEnabled>N</EspEnabled>

<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

<MasterVS>0</MasterVS>

<MasterVSID>0</MasterVSID>

<AddVia>0</AddVia>

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<OCSPVerify>N</OCSPVerify>

<NumberOfRSs>0</NumberOfRSs>

</Data>

</Success>

</Response>
```

Add a WAF Rule to a Virtual Service

The following command adds the application-specific **cpanel_attacks** rule to the **20.200.25.200** Virtual Service:

```
curl -k https://bal:1fourall@20.200.25.100/access/
vsaddwafrule?vs=20.200.25.200&port=80&prot=tcp&rule=A/cpanel_attacks
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Command completed ok</Success>
```

```
</Response>
```

Delete a Virtual Service

The following command deletes the **20.200.25.250** Virtual Service.

Situation before delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><VS>

<Index>1</Index>

<VSAddress>20.200.25.210</VSAddress>

<VSPort>80</VSPort>

<Enable>Y</Enable>

.

.

.

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<NumberOfRSs>1</NumberOfRSs>

<Rs>

<VSIndex>1</VSIndex>

<RsIndex>1</RsIndex>

<Addr>20.200.25.22</Addr>

<Port>80</Port>

<Forward>nat</Forward>

<Weight>1000</Weight>

<Limit>0</Limit>
```

```
<Enable>Y</Enable>

</Rs>

</VS>

<VS>

<Index>2</Index>

<VSAddress>20.200.25.250</VSAddress>

<VSPort>80</VSPort>

<Enable>Y</Enable>

.

.

.

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<NumberOfRSs>0</NumberOfRSs>

</VS>

</Data>

</Success>

</Response>

curl -k "https://bal:1fourall@20.200.25.100/access/
delvs?vs=20.200.25.250&port=80&prot=tcp"
```

Response:

```
<Response stat="200" code="ok">

<Success>Command completed ok</Success>

</Response>
```

If you run the **listvs** command after deleting the Virtual Service, the existing Virtual Service details will be displayed but the deleted Virtual Service details will no longer appear.

Situation after delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><VS>

<Index>1</Index>

<VSAddress>20.200.25.210</VSAddress>

<VSPort>80</VSPort>

<Enable>Y</Enable>

.

.

.

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<NumberOfRSs>1</NumberOfRSs>

<Rs>

<VSIndex>1</VSIndex>

<RsIndex>1</RsIndex>

<Addr>20.200.25.22</Addr>

<Port>80</Port>

<Forward>nat</Forward>

<Weight>1000</Weight>

<Limit>0</Limit>

<Enable>Y</Enable>

</Rs>

</VS>
```

```
</Data>

</Success>

</Response>
```

Add Client Side Single Sign On (SSO) Configuration

The following command adds a client side SSO domain with the name **clientdomainname**:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
adddomain?domain=clientdomainname"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Domain 'CLIENTDOMAINNAME' created</Success>

</Response>
```

Display a Client Side SSO Configuration

The following command displays details about the **clientdomainname** SSO domain:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
showdomain?domain=clientdomainname"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Domain>

<Id>4</Id>

<Name>CLIENTDOMAINNAME</Name>

<testuser></testuser>

<ldap_version>3</ldap_version>

<server_side>0</server_side>
```

```
<auth_type>LDAP-StartTLS</auth_type>

<logon_fmt>Principalname</logon_fmt>

<logon_domain></logon_domain>

<kerberos_domain></kerberos_domain>

<kerberos_kdc></kerberos_kdc>

<kcd_username></kcd_username>

<max_failed_auths>0</max_failed_auths>

<reset_fail_tout>60</reset_fail_tout>

<unblock_tout>1800</unblock_tout>

<sess_tout_type>idle time</sess_tout_type>

<sess_tout_idle_pub>900</sess_tout_idle_pub>

<sess_tout_duration_pub>1800</sess_tout_duration_pub>

<sess_tout_idle_priv>900</sess_tout_idle_priv>

<sess_tout_duration_priv>28800</sess_tout_duration_priv>

<cert_check_asi>0</cert_check_asi>

</Domain>

</Data>

</Success>

</Response>
```

Modify a Client Side SSO Configuration

The following command modifies the **clientdomainname** SSO domain by changing the authentication type to **RADIUS** and changing the maximum number of failed authentication requests to **5**:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
moddomain?domain=clientdomainname&authtype=RADIUS&max_failed_auths=5"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<Response stat="200" code="ok">
<Success><Data><Domain>
<Id>4</Id>
<Name>CLIENTDOMAINNAME</Name>
<testuser></testuser>
<ldap_version>3</ldap_version>
<server_side>0</server_side>
<auth_type>RADIUS</auth_type>
<logon_fmt>Principalname</logon_fmt>
<logon_domain></logon_domain>
<kerberos_domain></kerberos_domain>
<kerberos_kdc></kerberos_kdc>
<kcd_username></kcd_username>
<max_failed_auths>5</max_failed_auths>
<reset_fail_tout>60</reset_fail_tout>
<unblock_tout>1800</unblock_tout>
<sess_tout_type>idle time</sess_tout_type>
<sess_tout_idle_pub>900</sess_tout_idle_pub>
<sess_tout_duration_pub>1800</sess_tout_duration_pub>
<sess_tout_idle_priv>900</sess_tout_idle_priv>
<sess_tout_duration_priv>28800</sess_tout_duration_priv>
<cert_check_asi>0</cert_check_asi>
</Domain>
</Data>
</Success>
```

```
</Response>
```

Delete a Client Side SSO Configuration

The following command deletes the **clientdomainname** SSO domain.

Situation before delete:

The below XML response shows the situation before the delete (using the **showdomain** command):

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Domain>

<Id>4</Id>

<Name>CLIENTDOMAINNAME</Name>

<testuser></testuser>

<ldap_version>3</ldap_version>

<server_side>0</server_side>

<auth_type>LDAP-StartTLS</auth_type>

<logon_fmt>Principalname</logon_fmt>

<logon_domain></logon_domain>

<kerberos_domain></kerberos_domain>

<kerberos_kdc></kerberos_kdc>

<kcd_username></kcd_username>

<max_failed_auths>0</max_failed_auths>

<reset_fail_tout>60</reset_fail_tout>

<unblock_tout>1800</unblock_tout>

<sess_tout_type>idle time</sess_tout_type>

<sess_tout_idle_pub>900</sess_tout_idle_pub>

<sess_tout_duration_pub>1800</sess_tout_duration_pub>
```

```
<sess_tout_idle_priv>900</sess_tout_idle_priv>

<sess_tout_duration_priv>28800</sess_tout_duration_priv>

<cert_check_asi>0</cert_check_asi>

</Domain>

</Data>

</Success>

</Response>

curl -k "https://bal:1fourall@20.200.25.100/access/
deldomain?domain=clientdomainname"
```

Situation after delete:

After executing the command to delete the **clientdomainname** SSO domain, if a **showdomain** command is run for **clientdomainname**, an error will display saying the **Domain does not exist**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="422" code="fail">

<Error>Domain does not exist</Error>

</Response>
```

Add a Custom SSO Image Set

The following command uploads a custom SSO image set file called **Custom_Image_Set.tar**.

This command requires a file upload. In this example, the cURL **-X POST** and**--data-binary** options are used to specify the file to be uploaded to the LoadMaster:

Situation before upload:

The **listssoimages** command can be used to list any existing custom SSO image sets that are currently installed on the LoadMaster:

```
curl -k "https://bal:1fourall@20.200.25.100/access/listssoimages"
```

Response:

```
<Response stat="200" code="ok">

<Success>
```

```
<Data>

<Imagesets></Imagesets>

</Data>

</Success>

</Response>
```

The following command uploads the custom SSO image set:

```
curl -X POST --data-binary "@Custom_Image_Set.tar" -k
"https://bal:1fourall@20.200.25.100/access/ssoimages"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Custom image sets installed</Success>

</Response>
```

Delete a Custom SSO Image Set

The following example deletes the **IMAGESETNAME_1** custom SSO image set.

Situation before delete:

```
Before deleting the custom SSO image set, the listssoimages command can be run
to show the existing custom SSO image sets.<?xml version="1.0"
encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Imagesets>

<Imageset>

<Name>IMAGESETNAME_1</Name>

<Installed>Mon Jan 12 17:23:25 2015</Installed>

<Info></Info>

</Imageset>
```

```
</Imagesets>

</Data>

</Success>

</Response>

curl -k "https://bal:1fourall@20.200.25.100/access/
delssoimage?name=IMAGESETNAME_1"
```

Situation after delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Imagesets>

</Imagesets>

</Data>

</Success>

</Response>
```

List WAF Custom Rules

The following command displays a list of the installed WAF rules:

```
curl -k "https://bal:1fourall@20.200.25.100/access/listwafrules"
```

Response:

```
<Response stat="200" code="ok">

<Success>

<Data>

<Rules>

<Inactive1>Generic/ip_reputation</Inactive1>

<Inactive2>Generic/malware_detection</Inactive2>

<Inactive3>Generic/botnet_attacks</Inactive3>
```

<Inactive4>Generic/creditcard_known</Inactive4>

<Inactive5>Generic/creditcard_track_pan</Inactive5>

<Inactive6>ApplicationSpecific/cpanel_attacks</Inactive6>

<Inactive7>ApplicationSpecific/drupal_attacks</Inactive7>

<Inactive8>ApplicationSpecific/joomla_attacks</Inactive8>

<Inactive9>ApplicationSpecific/modx_attacks</Inactive9>

<Inactive10>ApplicationSpecific/netcat_attacks</Inactive10>

<Inactive11>ApplicationSpecific/oscommerce_attacks</Inactive11>

<Inactive12>ApplicationSpecific/phpbb_attacks</Inactive12>

<Inactive13>ApplicationSpecific/sharepoint_attacks</Inactive13>

<Inactive14>ApplicationSpecific/typo3_attacks</Inactive14>

<Inactive15>ApplicationSpecific/vbulletin_attacks</Inactive15>

<Inactive16>ApplicationSpecific/wordpress_attacks</Inactive16>

<Inactive17>ApplicationSpecific/owa_attacks</Inactive17>

<Inactive18>ApplicationSpecific/iis_attacks</Inactive18>

<Inactive19>ApplicationGeneric/lfi_attacks</Inactive19>

<Inactive20>ApplicationGeneric/rfi_attacks</Inactive20>

<Inactive21>ApplicationGeneric/sqli_attacks</Inactive21>

<Inactive22>ApplicationGeneric/xss_attacks</Inactive22>

</Rules>

</Data>

</Success>

</Response>

Upload a WAF Custom Rule

The following command uploads a custom WAF rule fule called **CustomRules.conf**:

```
curl -X POST --data-binary "@CustomRules.conf" -k
"https://bal:1fourall@20.200.25.100/access/
addwafcustomrule?filename=CustomRules.conf"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Custom rule(s) installed</Success>

</Response>
```

Download a WAF Custom Rule

The following command downloads a custom rule file called **Custom Rules**:

```
curl -o DownloadedCustomRules.conf -k "https://bal:1fourall@20.200.25.100/
access/downloadwafcustomrule?filename=CustomRules"
```

This command downloads one file at a time. One file could contain multiple rules.

Delete a WAF Custom Rule

The following command deletes a WAF custom rule called **CustomRules**.

Note: Only one rule can be deleted at a time.

Situation before delete:

To view a list of installed rules, run the **listwafrules** command.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Rules><Inactive1>Custom/</Inactive1>

<Inactive2>Custom/CustomRules</Inactive2>

<Inactive3>Generic/ip_reputation</Inactive3>
```

```
<Inactive4>Generic/malware_detection</Inactive4>

.

.

.
```

Note: The file extension is not used in the **filename** parameter in this command.

```
curl -k "https://bal:1fourall@20.200.25.100/access/
delwafcustomrule?filename=CustomRules"
```

Situation after delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><Rules><Inactive1>Custom/</Inactive1>

<Inactive2>Generic/ip_reputation</Inactive2>

<Inactive3>Generic/malware_detection</Inactive3>

.

.

.
```

Statistics

Refer to the following sections for examples on running the LoadMaster RESTful API commands relating to statistics.

Show stats

The following command displays a number of LoadMaster statistics such as connections per second and bits per second:

```
curl -k "https://bal:1fourall@20.200.25.100/access/stats"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<Response stat="200" code="ok">
<Success><Data><VStotals>
<ConnsPerSec>0</ConnsPerSec>
<BitsPerSec>0</BitsPerSec>
<BytesPerSec>0</BytesPerSec>
<PktsPerSec>0</PktsPerSec>
</VStotals>
<Vs>
<VSAddress>20.200.25.94</VSAddress>
<VSPort>80</VSPort>
<VSProt>tcp</VSProt>
<Index>14</Index>
<ErrorCode>0</ErrorCode>
<Enable>1</Enable>
<TotalConns>0</TotalConns>
<TotalPkts>0</TotalPkts>
<TotalBytes>0</TotalBytes>
<TotalBits>0</TotalBits>
<ActiveConns>0</ActiveConns>
<BytesRead>0</BytesRead>
<BytesWritten>0</BytesWritten>
<ConnsPerSec>0</ConnsPerSec>
<WafEnable>0</WafEnable>
</Vs>
<Vs>
```

```
<VSAddress>20.200.25.99</VSAddress>

<VSPort>80</VSPort>

<VSProt>tcp</VSProt>

<Index>13</Index>

<ErrorCode>0</ErrorCode>

<Enable>1</Enable>

<TotalConns>0</TotalConns>

<TotalPkts>0</TotalPkts>

<TotalBytes>0</TotalBytes>

<TotalBits>0</TotalBits>

<ActiveConns>0</ActiveConns>

<BytesRead>0</BytesRead>

<BytesWritten>0</BytesWritten>

<ConnsPerSec>0</ConnsPerSec>

<WafEnable>0</WafEnable>

</Vs>

.

.

.
```

Rules & Checking

Refer to the following sections for examples on running the LoadMaster RESTful API commands relating to content rules and check parameters.

Add a content rule to the system

The following command adds a content rule to the LoadMaster:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
addrule?name=contentrule&pattern=pattern"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><MatchContentRule>

<Name>contentrule</Name>

<Pattern>pattern</Pattern>

<MatchType>Regex</MatchType>

<AddHost>N</AddHost>

<Negate>N</Negate>

<CaseIndependent>N</CaseIndependent>

<IncludeQuery>N</IncludeQuery>

<Header></Header>

<MustFail>N</MustFail>

</MatchContentRule>

</Data>

</Success>

</Response>
```

Modify a Content Rule

The following command modifies an existing rule called **contentrule** by changing the **matchtype** to **prefix** and enabling the option to ignore case:

Note: Reference the rule by its name.

```
curl -k "https://bal:1fourall@20.200.25.100/access/
modrule?name=contentrule&pattern=pattern&matchtype=prefix&nocase=Y"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><MatchContentRule>

<Name>contentrule</Name>

<Pattern>pattern</Pattern>

<MatchType>prefix</MatchType>

<AddHost>N</AddHost>

<Negate>N</Negate>

<CaseIndependent>Y</CaseIndependent>

<IncludeQuery>N</IncludeQuery>

<Header></Header>

<MustFail>N</MustFail>

</MatchContentRule>

</Data>

</Success>

</Response>
```

Delete Rule

The following command deletes the rule named **contentrule**.

Situation before delete:

Before deleting the content rule, the **showrule** command can be run to display a list of currently installed rules.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><MatchContentRule>

<Name>contentrule</Name>
```

```
<Pattern>pattern</Pattern>

<MatchType>Regex</MatchType>

<AddHost>N</AddHost>

<Negate>N</Negate>

<CaseIndependent>N</CaseIndependent>

<IncludeQuery>N</IncludeQuery>

<Header></Header>

<MustFail>N</MustFail>

</MatchContentRule>

</Data>

</Success>

</Response>

curl -k "https://bal:1fourall@20.200.25.100/access/delrule?name=contentrule"
```

Situation after delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data></Data>

</Success>

</Response>
```

List the Content Rules Assigned to a Virtual Service

The **listvs** command shows the content rules assigned to a Virtual Service:

```
curl -k "https://bal:1fourall@20.200.25.100/access/listvs"
```

Response:

```
<Response stat="200" code="ok">

<Success>
```

```
<Data>
<VS>
<Status>Down</Status>
<Index>1</Index>
<VSAddress>20.200.25.250</VSAddress>
<VSPort>80</VSPort>
<NickName>Test</NickName>
<Enable>Y</Enable>
<SSLReverse>N</SSLReverse>
<SSLReencrypt>N</SSLReencrypt>
<Intercept>Y</Intercept>
<InterceptOpts>
<Opt>opnormal</Opt>
<Opt>auditnone</Opt>
<Opt>reqdatadisable</Opt>
<Opt>resdatadisable</Opt>
</InterceptOpts>
<AlertThreshold>0</AlertThreshold>
<Transactionlimit>0</Transactionlimit>
<Transparent>Y</Transparent>
<ServerInit>0</ServerInit>
<StartTLSMode>0</StartTLSMode>
<Idletime>0</Idletime>
<Cache>N</Cache>
<Compress>N</Compress>
```

<Verify>0</Verify>

<UseforSnat>N</UseforSnat>

<ForceL7>Y</ForceL7>

<ClientCert>0</ClientCert>

<ErrorCode>0</ErrorCode>

<CheckUse1.1>N</CheckUse1.1>

<MatchLen>0</MatchLen>

<CheckUseGet>0</CheckUseGet>

<SSLRewrite>0</SSLRewrite>

<VStype>http</VStype>

<FollowVSID>0</FollowVSID>

<Protocol>tcp</Protocol>

<Schedule>rr</Schedule>

<CheckType>http</CheckType>

<PersistTimeout>0</PersistTimeout>

<CheckPort>0</CheckPort>

<NRules>0</NRules>

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>1</NPreProcessRules>

<PreProcessRules>

<Name>test</Name>

</PreProcessRules>

<EspEnabled>N</EspEnabled>

<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

```
<MasterVS>0</MasterVS>

<MasterVSID>0</MasterVSID>

<AddVia>0</AddVia>

<TlsType>N</TlsType>

<NeedHostName>N</NeedHostName>

<OCSPVerify>N</OCSPVerify>

<NumberOfRSs>0</NumberOfRSs>

</VS>

</Data>

</Success>

</Response>
```

Add a Content Rule to a Virtual Service

The following command adds the rule **contentrule** to the **20.200.25.94** Virtual Service:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
addprerule?vs=20.200.25.94&port=80&prot=tcp&rule=contentrule"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Command completed ok</Success>

</Response>
```

Unassign a Content Rule from a Virtual Service

The following command unassigns the **contentrule** rule from the **20.200.25.94** Virtual Service:

Situation before delete:

A **listvs** command can be run before deleting to view the content rules that are assigned to the Virtual Services.

```
.

.

.

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>1</NPreProcessRules>

<PreProcessRules>

<Name>contentrule</Name>

</PreProcessRules>

<EspEnabled>N</EspEnabled>

<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

.

.

.

curl -k "https://bal:1fourall@20.200.25.100/access/
delprerule?vs=20.200.25.94&port=80&prot=tcp&rule=contentrule"
```

Situation after delete:

```
.

.

.

<NRequestRules>0</NRequestRules>

<NResponseRules>0</NResponseRules>

<NPreProcessRules>0</NPreProcessRules>

<EspEnabled>N</EspEnabled>
```

```
<InputAuthMode>0</InputAuthMode>

<OutputAuthMode>0</OutputAuthMode>

.

.

.
```

Set Retry Interval

A health check is performed on each Real Server. The retry interval defines the amount of time (in seconds) between checks. The following command sets the **RetryInterval** to **12** seconds:

```
curl -k "https://bal:1fourall@20.200.25.100/access/modhealth?RetryInterval=12"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><RetryInterval>12</RetryInterval>

<Timeout>4</Timeout>

<RetryCount>2</RetryCount>

</Data>

</Success>

</Response>
```

Certificates

Refer to the following section for examples on RESTful API commands relating to certificates.

Add a Certificate

The following command uploads a certificate called **cert.pem** to the LoadMaster:

```
curl -X POST --data-binary "@cert.pem" -k "https://bal:1fourall@20.200.25.100/
access/addcert?cert=acert"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Certificate Successfully Installed</Success>

</Response>
```

Delete a Certificate

Note: The following command deletes a certificate called **acert** from the LoadMaster. Reference the certificate by name.

```
curl -X POST --data-binary "@cert.pem" -k "https://bal:1fourall@20.200.25.100/
access/delcert?cert=acert"
```

Response:

```
<Response stat="200" code="ok">

<Success>Command completed ok</Success>

</Response>
```

Backup a Certificate

The following command downloads a password-protected backup file containing all of the certificates that exist on the LoadMaster:

```
curl -o backedupcert -k "https://bal:1fourall@20.200.25.100/access/
backupcert?password=password"
```

Note: The password parameter is mandatory.

Response:

<BackupFileDownloads>

Restore Certificates

The following command restores Virtual Service certificates from a backup file called **backedupcert**:

```
curl -X POST --data-binary "@backedupcert" -k
"https://bal:1fourall@20.200.25.100/access/
restorecert?password=password&type=vs"
```

The **type** parameter has three possible values:

- **Full** - All Virtual Service and intermediate certificates
- **Third** - Intermediate certificates only
- **Vs** - Virtual Service certificates only

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Certificates Restored</Success>

</Response>
```

GEO

Refer to the following sections for examples on running the LoadMaster RESTful API commands relating to GEO.

Add an FQDN

The following command adds an **FQDN** called **addedfqdn.com** which has default settings:

```
curl -k "https://bal:1fourall@20.200.25.100/access/addfqdn?fqdn=addedfqdn.com"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Added FQDN addedfqdn.com</Success>

</Response>
```

Display the FQDN Settings

To display the FQDN settings, run the **showFQDN** command:

```
curl -k "https://bal:1fourall@20.200.25.100/access/showfqdn?fqdn=addedfqdn.com"
```

Response:

```
<Response stat="200" code="ok">

<Success>

<Data>

<fqdn>

<Status>Down</Status>

<FullyQualifiedDomainName>addedfqdn.com.</FullyQualifiedDomainName>

<SelectionCriteria>rr</SelectionCriteria>

<FailTime>0</FailTime>

<SiteRecoveryMode>auto</SiteRecoveryMode>

<Mapping>0</Mapping>

<failover>N</failover>

</fqdn>

</Data>

</Success>

</Response>
```

Modify an FQDN

The following command modifies the existing **addedfqdn.com** FQDN to set the **selectioncriteria** to Weighted Round Robin (WRR):

```
curl -k "https://bal:1fourall@20.200.25.100/access/
modfqdn?fqdn=addedfqdn.com&selectioncriteria=wrr"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><fqdn><Status>Up</Status>

<FullyQualifiedDomainName>addedfqdn.com.</FullyQualifiedDomainName>
```

<SelectionCriteria>wrr</SelectionCriteria>

```
<FailTime>0</FailTime>

<SiteRecoveryMode>auto</SiteRecoveryMode>

<Mapping>0</Mapping>

<failover>N</failover>

</fqdn></Data>

</Success>

</Response>
```

Add an IP Address to an FQDN

The following command adds an IP address (**2.2.2.2**) to the **addedfqdn.com** FQDN:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
addmap?fqdn=addedfqdn.com&ip=2.2.2.2"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Added Map 2.2.2.2 to FQDN addedfqdn.com.</Success>

</Response>

The details of the IP address map can be seen by running the showFQDN
command.<Response stat="200" code="ok">

<Success>

<Data>

<fqdn>

<Status>Up</Status>

<FullyQualifiedDomainName>addedfqdn.com.</FullyQualifiedDomainName>

<SelectionCriteria>rr</SelectionCriteria>

<FailTime>0</FailTime>

<SiteRecoveryMode>auto</SiteRecoveryMode>
```

```
<Mapping>1</Mapping>

<failover>N</failover>

<Map>

<Status>Up</Status>

<Index>1</Index>

<IPAddress>2.2.2.2</IPAddress>

<Checker>icmp</Checker>

<CheckerPort>0</CheckerPort>

<Weight>1000</Weight>

<Enable>Y</Enable>

<LocationLatitude>0.00000000</LocationLatitude>

<LocationLongitude>0.00000000</LocationLongitude>

</Map>

</fqdn>

</Data>

</Success>

</Response>
```

Modify the IP Address of an FQDN

The following command edits the **2.2.2.2** map settings in the **addedfqdn.com** FQDN by changing the **checker** to **icmp**:

Note: Provide the IP address and the FQDN name that is belongs to.

```
curl -k "https://bal:1fourall@20.200.25.100/access/
modmap?fqdn=addedfqdn.com&ip=2.2.2.2&checker=icmp"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<Response stat="200" code="ok">

<Success><Data><fqdn><Status>Up</Status>

<FullyQualifiedDomainName>addedfqdn.com.</FullyQualifiedDomainName>

<SelectionCriteria>rr</SelectionCriteria>

<FailTime>0</FailTime>

<SiteRecoveryMode>auto</SiteRecoveryMode>

<Mapping>2</Mapping>

<failover>N</failover>

<Map><Status>Up</Status>

<Index>2</Index>

<IPAddress>2.2.2.2</IPAddress>

<Checker>icmp</Checker>

<CheckerPort>0</CheckerPort>

<Weight>1000</Weight>

<Enable>Y</Enable>

<LocationLatitude>0.00000000</LocationLatitude>

<LocationLongitude>0.00000000</LocationLongitude>

</Map></fqdn></Data>

</Success>

</Response>
```

Delete an IP Address from an FQDN

The following command deletes the **2.2.2.2** IP address map from the **addedfqdn.com** FQDN.

Situation before delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">
```

```
<Success><Data><fqdn><Status>Up</Status>

<FullyQualifiedDomainName>addedfqdn.com.</FullyQualifiedDomainName>

<SelectionCriteria>rr</SelectionCriteria>

<FailTime>0</FailTime>

<SiteRecoveryMode>auto</SiteRecoveryMode>

<Mapping>1</Mapping>

<failover>N</failover>

<Map><Status>Up</Status>

<Index>1</Index>

<IPAddress>2.2.2.2</IPAddress>

<Checker>icmp</Checker>

<CheckerPort>0</CheckerPort>

<Weight>1000</Weight>

<Enable>Y</Enable>

<LocationLatitude>0.00000000</LocationLatitude>

<LocationLongitude>0.00000000</LocationLongitude>

</Map></fqdn></Data>

</Success>

</Response>

curl -k "https://bal:1fourall@20.200.25.100/access/
delmap?fqdn=addedfqdn.com&ip=2.2.2.2"
```

Situation after delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><fqdn><Status>Down</Status>
```

```
<FullyQualifiedDomainName>addedfqdn.com.</FullyQualifiedDomainName>

<SelectionCriteria>rr</SelectionCriteria>

<FailTime>0</FailTime>

<SiteRecoveryMode>auto</SiteRecoveryMode>

<Mapping>0</Mapping>

<failover>N</failover>

</fqdn></Data>

</Success>

</Response>
```

Delete an FQDN

The following command deletes the **addedfqdn.com** FQDN.

Situation before delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><fqdn><FullyQualifiedDomainName>addedfqdn.com.</FullyQualifiedDo
mainName>

<SelectionCriteria>rr</SelectionCriteria>

<FailTime>0</FailTime>

<SiteRecoveryMode>auto</SiteRecoveryMode>

<Mapping>0</Mapping>

<failover>N</failover>

</fqdn></Data>

</Success>

</Response>

curl -k "https://bal:1fourall@20.200.25.100/access/delfqdn?fqdn=addedfqdn.com"
```

Response:

```
<Response stat="200" code="ok">

<Success>Deleted FQDN addedfqdn.com.</Success>

</Response>
```

Situation after delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="404" code="fail">

<Error>No geo data found</Error>

</Response>
```

Add a Cluster

The following command adds a cluster called **addedcluster** with an IP address of **3.3.3.3**. A cluster requires an IP address and a name.

```
curl -k "https://bal:1fourall@20.200.25.100/access/
addcluster?ip=3.3.3.3&name=addedcluster"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Cluster 3.3.3.3 added</Success>

</Response>
```

To view the cluster, run the listclusters command.

```
<Response stat="200" code="ok">

<Success>

<Data>

<cluster>

<Index>1</Index>

<Name>addedcluster</Name>
```

```
<ClusterVSAddress/>

<IPAddress>3.3.3.3</IPAddress>

<Checker>none</Checker>

<CheckerPort>0</CheckerPort>

<Type>default</Type>

<Enable>Y</Enable>

<LocationLatitude>0.00000000</LocationLatitude>

<LocationLongitude>0.00000000</LocationLongitude>

</cluster>

</Data>

</Success>

</Response>
```

Modify a Cluster

The following command modifies the cluster check type:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
modcluster?ip=3.3.3.3&checker=icmp"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><cluster><Status>Up</Status>

<Index>1</Index>

<Name>addedcluster</Name>

<ClusterVSAddress></ClusterVSAddress><IPAddress>3.3.3.3</IPAddress>

<Checker>icmp</Checker>

<CheckerPort>0</CheckerPort>
```

```
<Type>default</Type>

<Enable>Y</Enable>

<LocationLatitude>0.00000000</LocationLatitude>

<LocationLongitude>0.00000000</LocationLongitude>

</cluster></Data>

</Success>

</Response>
```

Modify a Cluster Location

The following command modifies the existing **3.3.3.3** cluster by setting the longitude and latitude to 43.3442deg N, 6.2675deg W. Values must be given in seconds, with a negative value for south or west:

```
curl -k "https://bal:2fourall@10.154.25.100/access/
clustchangeloc?ip=3.3.3.3&latsecs=156039&longsecs=-22563"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>Cluster location updated</Success>

</Response>
```

Delete a Cluster

The following command deletes the **3.3.3.3** cluster.

Situation before delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><cluster><Index>1</Index>

<Name>addedcluster</Name>

<ClusterVSAddress></ClusterVSAddress><IPAddress>3.3.3.3</IPAddress>
```

```
<Checker>none</Checker>

<CheckerPort>0</CheckerPort>

<Type>default</Type>

<Enable>Y</Enable>

<LocationLatitude>0.00000000</LocationLatitude>

<LocationLongitude>0.00000000</LocationLongitude>

</cluster></Data>

</Success>

</Response>

curl -k "https://bal:1fourall@20.200.25.100/access/delcluster?ip=3.3.3.3"
```

Response:

```
<Response stat="200" code="ok">

<Success>Cluster deleted</Success>

</Response>
```

Situation after delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="404" code="fail">
```

<Error>No geo data found</Error>

```
</Response>
```

Modify the GEO Miscellaneous Parameters

The following command updates the **checkinterval** to **60** seconds and the connection timeout to **10** seconds:

```
curl -k "https://bal:1fourall@20.200.25.100/access/
modparams?checkinterval=60&conntimeout=10"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<Response stat="200" code="ok">

<Success><Data><soa><TTL>10</TTL>

<persist>0</persist>

</soa><check><CheckInterval>60</CheckInterval>

<ConnTimeout>10</ConnTimeout>

<RetryAttempts>2</RetryAttempts>

</check></Data>

</Success>

</Response>
```

Add an IP Range

The following command adds an IP range of **4.4.0.0** with a **mask** of **16**:

Note: The **mask** parameter is the netmask.

```
curl -k "https://bal:1fourall@20.200.25.100/access/addip?ip=4.4.0.0&mask=16"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>IP range added</Success>

</Response>
```

Situation after Add:

To show the default settings of the IP range just added, run the **listips** command.

```
<Response stat="200" code="ok">

<Success>

<Data>

<IPAddress>

<Index>1</Index>
```

```
<Used>Y</Used>

<Ip>4.4.0.0</Ip>

<IPAddress>4.4.0.0</IPAddress>

<Mask>16</Mask>

<Country>-1</Country>

<IsCustom>N</IsCustom>

<CustomLocation>0</CustomLocation>

</IPAddress>

</Data>

</Success>

</Response>
```

Modify an IP Location

The following command modifies the latitude and longitude of the **4.4.0.0** IP range:

Note: The **lat** and **long** parameters are in seconds.

```
curl -k "https://bal:1fourall@20.200.25.100/access/
modiploc?ip=4.4.0.0&lat=360&long=360"
```

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success>IP range location updated</Success>

</Response>
```

Situation after Modification

To view the IP range settings, run a **listips** command:

```
<Response stat="200" code="ok">

<Success>
```

```
<Data>

<IPAddress>

<Index>1</Index>

<Used>Y</Used>

<Ip>4.4.0.0</Ip>

<IPAddress>4.4.0.0</IPAddress>

<Mask>16</Mask>

<Latitude>360</Latitude>

<Longitude>360</Longitude>

<Country>-1</Country>

<IsCustom>N</IsCustom>

<CustomLocation>0</CustomLocation>

</IPAddress>

</Data>

</Success>

</Response>
```

Delete an IP Range

The following command deletes the IP range **4.4.0.0**:

Note: You cannot delete IP address ranges if they are configured in an FQDN.

Situation before delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="200" code="ok">

<Success><Data><IPAddress><Index>1</Index>

<Used>Y</Used>
```

```
<Ip>4.4.0.0</Ip>

<IPAddress>4.4.0.0</IPAddress>

<Mask>16</Mask>

<Country>-1</Country>

<IsCustom>N</IsCustom>

<CustomLocation>0</CustomLocation>

</IPAddress></Data>

</Success>

</Response>

curl -k "https://bal:1fourall@20.200.25.100/access/delip?ip=4.4.0.0"
```

Situation after delete:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Response stat="404" code="fail">

<Error>No geo data found</Error>

</Response>
```

3

Basic Scripting Examples

The following simple script examples show how the API can be used to automate certain tasks.

Note: These examples are shell scripts, however any language with HTTP capabilities can be used.

The examples below contain variables rather than values. Variables are preceded by a dollar sign (**$**). For example:

- **$1** - This variable is set to the LoadMaster IP address, for example **20.200.25.100**.
- **$2** - This variable is set to the username, for example **bal**.
- **$3** - This variable is set to the password, for example **1fourall**.

Related Links

For details, see the following topics:

- Shell Script Example to Add Multiple Virtual Services with Default Settings
- Shell Script Example to Delete Multiple Basic Virtual Services
- Shell Script Example to Modify the Same Parameter on Multiple Virtual Services
- Shell Script Example to Add Multiple FQDNs
- Shell Script Example to Add Multiple IP Addresses to an FQDN
- Shell Script Example to Extract the SubVSs Index from one Command and Use in Another

Shell Script Example to Add Multiple Virtual Services with Default Settings

The following example adds multiple Virtual Services with default settings:

Note: The first five arguments are: LoadMaster IP address, username, password, port and protocol, followed by the addresses of the Virtual Services to add, for example **bash addvs.sh 20.200.50.50 bal 1fourall 80 tcp 20.200.50.10 20.200.50.11 …**

```
for var in "${@:6}"

do

   curl -k "https://$2:$3@$1/access/addvs?vs=$var&port=$4&prot=$5"

done
```

Shell Script Example to Delete Multiple Basic Virtual Services

The following example deletes multiple Virtual Services with default settings:

```
for var in "${@:6}"

do

   curl -k "https://$2:$3@$1/access/delvs?vs=$var&port=$4&prot=$5"

done
```

Shell Script Example to Modify the Same Parameter on Multiple Virtual Services

The following example modifies the same parameter on multiple Virtual Services at the same time.

Note: The Virtual Service index must be used in this case.

```
for var in "${@:6}"
```

```
do
   curl -k "https://$2:$3@$1/access/modvs?vs=$var&$4=$5"
done
```

Shell Script Example to Add Multiple FQDNs

The following example adds multiple FQDNs:

```
for var in "${@:4}"
do
   curl -k "https://$2:$3@$1/access/addfqdn?fqdn=$var"
done
```

Shell Script Example to Add Multiple IP Addresses to an FQDN

The following example adds multiple IP addresses to an FQDN:

```
for var in "${@:5}"
do
   curl -k "https://$2:$3@$1/access/addmap?fqdn=$4&ip=$var"
done
```

Shell Script Example to Extract the SubVSs Index from one Command and Use in Another

When running commands on SubVSs, the SubVS index can be used to identify the relevant SubVS to run the command on. An example of how to get the SubVS index is below:

```
subvsindex=$(curl -k "https://$2:$3@$1/access/
modvs?vs=$4&port=$5&prot=$6&createsubvs=" | xpath Response/Success/Data/SubVS/
VSIndex/"text()")
```

The **subvsindex** variable can then be used in a command, for example:

```
curl -k https://$2:$3@$1/access/modvs?vs=$subvsindex&nickname=newnickname
```

4

Microsoft Exchange 2013 HTTPS Offloaded Example

This script replicates the same configuration and settings that would be applied if the Exchange 2013 template had been used to configure a Virtual Service.

A file called **exchange.sh** is run which contains all of the information below. The variables are set as follows:

- **$1** - The username (**bal**)
- **$2** - The password (**1fourall**)
- **$3** - The LoadMaster IP address (**20.200.50.50**)
- **$4** - The IP address of the Virtual Service (**20.200.50.200**)

A redirect Virtual Service called **Exchange HTTPS Offloaded - HTTPS Redirect** is created.

Another Virtual Service called **Exchange HTTPS Offloaded** is also created.

sh exchange.sh bal 1fourall 20.200.50.50 20.200.50.200

```
#

# Exchange 2013 HTTPS Offloaded

#

#Redirect Virtual Service
```

```
curl -k -u $1:$2 "https://$3/access/
addvs?vs=$4&port=80&prot=tcp&Nickname=Exchange%202013%20HTTPS%20Offloaded%20-%2
0HTTP%20Redirect&forcel7=1&errorcode=302&errorurl=https:\/\/%25h%25s&checktype=
none"

#Main Virtual Service

curl -k -u $1:$2 "https://$3/access/
addvs?vs=$4&port=443&prot=tcp&Nickname=Exchange%202013%20HTTPS%20Offloaded&Enab
le=Y&ForceL7=Y&SSLAcceleration=Y&SSLReverse=N&SSLReencrypt=N&Intercept=N&Transp
arent=Y&Schedule=rr&IdleTime=1800&CheckUse1.1=N&CheckType=http"

#

# Exchange 2013 HTTPS Offloaded - ActiveSync

#

#Counter to increment arrays for returned SubVS and Real Server indices

counter=0;

#Find the VSIndex and the RsIndex of the first subvs created and store in
arrays

subvsindexarray[$counter]=$(curl -k -u $1:$2 "https://$3/access/
modvs?vs=$4&port=443&prot=tcp&createsubvs=" | xpath Response/Success/Data/
SubVS/VSIndex/"text()")

rsindexarray[$counter]=$(curl -k "https://$1:$2@$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/SubVS/
RsIndex/"text()")

#echo "Sub VS index is $counter"

#

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&nickname=Exchange%202013%20HTTPS%20Offloa
ded%20-%20ActiveSync&Forward=nat&Weight=1000&Limit=0&Enable=Y"

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&CheckPort=443&CheckUse1.1=Y&CheckURL=/mic
rosoft-server-activesync/healthcheck.htm&CheckType=https"

#

# Exchange 2013 HTTPS Offloaded - Autodiscover

#
```

```
let counter=$counter+1

curl -k -u $1:$2 "https://$3/access/modvs?vs=$4&port=443&prot=tcp&createsubvs="

#Get indices

subvsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/VSIndex/"text()")

rsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/RsIndex/"text()")

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&nickname=Exchange%202013%20HTTPS%20Offloa
ded%20-%20Autodiscover&Forward=nat&Weight=1000&Limit=0&Enable=Y"

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&CheckPort=443&CheckUse1.1=Y&CheckURL=/aut
odiscover/healthcheck.htm&CheckType=https"

#

# Exchange 2013 HTTPS Offloaded - ECP

#

let counter=$counter+1

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=$4&port=443&prot=tcp&createsubvs="

#Get indices

subvsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/VSIndex/"text()")

rsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/RsIndex/"text()")

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&nickname=Exchange%202013%20HTTPS%20Offloa
ded%20-%20ECP&Forward=nat&Weight=1000&Limit=0&Enable=Y"

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&CheckPort=443&CheckUse1.1=Y&CheckURL=/ecp
/healthcheck.htm&CheckType=https"
```

```
#

# Exchange 2013 HTTPS Offloaded - EWS

#

let counter=$counter+1

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=$4&port=443&prot=tcp&createsubvs="

#Get indices

subvsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/VSIndex/"text()")

rsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/RsIndex/"text()")



curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&nickname=Exchange%202013%20HTTPS%20Offloa
ded%20-%20EWS&Forward=nat&Weight=1000&Limit=0&Enable=Y"

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&CheckPort=443&CheckUse1.1=Y&CheckURL=/ews
/healthcheck.htm&CheckType=https"

#

# Exchange 2013 HTTPS Offloaded - MAPI

#

let counter=$counter+1

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=$4&port=443&prot=tcp&createsubvs="

#Get indices

subvsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/VSIndex/"text()")

rsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
```

```
SubVS[last\(\)]/RsIndex/"text()")

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&nickname=Exchange%202013%20HTTPS%20Offloa
ded%20-%20MAPI&Forward=nat&Weight=1000&Limit=0&Enable=Y"

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&CheckPort=443&CheckUse1.1=Y&CheckURL=/map
i/healthcheck.htm&CheckType=https"

#

# Exchange 2013 HTTPS Offloaded - OAB

#

let counter=$counter+1

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=$4&port=443&prot=tcp&createsubvs="




#Get indices

subvsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/VSIndex/"text()")

rsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/RsIndex/"text()")

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&nickname=Exchange%202013%20HTTPS%20Offloa
ded%20-%20OAB&Forward=nat&Weight=1000&Limit=0&Enable=Y"

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&CheckPort=443&CheckUse1.1=Y&CheckURL=/oab
/healthcheck.htm&CheckType=https"

#

# Exchange 2013 HTTPS Offloaded - OWA

#

let counter=$counter+1
```

```
curl  -k -u $1:$2 "https://$3/access/
modvs?vs=$4&port=443&prot=tcp&createsubvs="

#Get indices

subvsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/VSIndex/"text()")

rsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/RsIndex/"text()")

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&nickname=Exchange%202013%20HTTPS%20Offloa
ded%20-%20OWA&Forward=nat&Weight=1000&Limit=0&Enable=Y"

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&CheckPort=443&CheckUse1.1=Y&CheckURL=/owa
/healthcheck.htm&CheckType=https"




#

# Exchange 2013 HTTPS Offloaded - PowerShell

#

let counter=$counter+1

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=$4&port=443&prot=tcp&createsubvs="

#Get indices

subvsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/VSIndex/"text()")

rsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/RsIndex/"text()")

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&nickname=Exchange%202013%20HTTPS%20Offloa
ded%20-%20PowerShell&Forward=nat&Weight=1000&Limit=0&Enable=Y"
```

```
curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&CheckPort=443&CheckUse1.1=Y&CheckURL=/pow
ershell&CheckType=https"

#

# Exchange 2013 HTTPS Offloaded - RCP Forward

#

let counter=$counter+1

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=$4&port=443&prot=tcp&createsubvs="

#Get indices

subvsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/VSIndex/"text()")

rsindexarray[$counter]=$(curl -k -u $1:$2  "https://$3/access/
showvs?vs=$4&port=443&prot=tcp" | xpath Response/Success/Data/
SubVS[last\(\)]/RsIndex/"text()")



curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&nickname=Exchange%202013%20HTTPS%20Offloa
ded%20-%20RPC&Forward=nat&Weight=1000&Limit=0&Enable=Y"

curl  -k -u $1:$2 "https://$3/access/
modvs?vs=${subvsindexarray[$counter]}&CheckPort=443&CheckUse1.1=Y&CheckURL=/rpc
/healthcheck.htm&CheckType=https"
```

Note: The script below creates the necessary content rules. If these rules already exist on the LoadMaster, an error will occur.

```
#

# Create the content rules

#

rand=$RANDOM

curl -k -u $1:$2  "https://$3/access/addrule?name=Root_$rand&pattern=/^\/$/"

curl -k -u $1:$2  "https://$3/access/
```

```
addrule?name=ActiveSync_$rand&pattern=/^\/microsoft-server-
activesync.*/&nocase=1"

curl -k -u $1:$2  "https://$3/access/
addrule?name=Autodiscover_$rand&pattern=/^\/autodiscover.*/&nocase=1"

curl -k -u $1:$2  "https://$3/access/
addrule?name=ECP_$rand&pattern=/^\/ecp.*/&nocase=1"

curl -k -u $1:$2  "https://$3/access/
addrule?name=EWS_$rand&pattern=/^\/ews.*/&nocase=1"

curl -k -u $1:$2  "https://$3/access/
addrule?name=MAPI_$rand&pattern=/^\/mapi.*/&nocase=1"

curl -k -u $1:$2  "https://$3/access/
addrule?name=OAB_$rand&pattern=/^\/oab.*/&nocase=1"

curl -k -u $1:$2  "https://$3/access/
addrule?name=OWA_$rand&pattern=/^\/owa.*/&nocase=1"

curl -k -u $1:$2  "https://$3/access/
addrule?name=PowerShell_$rand&pattern=/^\/powershell.*/&nocase=1"

curl -k -u $1:$2  "https://$3/access/
addrule?name=RPC_$rand&pattern=/^\/rpc.*/&nocase=1"

curl -k -u $1:$2  "https://$3/access/
addrule?name=Redirect_Root_$rand&type=4&replacement=/owa&pattern="/^\/$/""

#

# Add the rules to the correct Virtual Service

#

#Add the rule to the parent Virtual Service

curl -k -u $1:$2   "https://$1:$2@$3/access/
addrequestrule?vs=$4&port=443&prot=tcp&rule=Redirect_Root_$rand"

#Add rules to the SubVS

curl -k -u $1:$2  "https://$1:$2@$3/access/
addrsrule?vs=$4&port=443&prot=tcp&rs=%21${rsindexarray[0]}$&rule=ActiveSync_$ra
nd"

curl -k -u $1:$2  "https://$1:$2@$3/access/
addrsrule?vs=$4&port=443&prot=tcp&rs=%21${rsindexarray[1]}&rule=Autodiscover_$r
and"
```

```
curl -k -u $1:$2  "https://$1:$2@$3/access/
addrsrule?vs=$4&port=443&prot=tcp&rs=%21${rsindexarray[2]}&rule=ECP_$rand"

curl -k -u $1:$2  "https://$1:$2@$3/access/
addrsrule?vs=$4&port=443&prot=tcp&rs=%21${rsindexarray[3]}&rule=EWS_$rand"

curl -k -u $1:$2  "https://$1:$2@$3/access/
addrsrule?vs=$4&port=443&prot=tcp&rs=%21${rsindexarray[4]}&rule=MAPI_$rand"

curl -k -u $1:$2  "https://$1:$2@$3/access/
addrsrule?vs=$4&port=443&prot=tcp&rs=%21${rsindexarray[5]}&rule=OAB_$rand"

curl -k -u $1:$2  "https://$1:$2@$3/access/
addrsrule?vs=$4&port=443&prot=tcp&rs=%21${rsindexarray[6]}&rule=OWA_$rand"

curl -k -u $1:$2  "https://$1:$2@$3/access/
addrsrule?vs=$4&port=443&prot=tcp&rs=%21${rsindexarray[7]}&rule=PowerShell_$ran
d"

curl -k -u $1:$2  "https://$1:$2@$3/access/
addrsrule?vs=$4&port=443&prot=tcp&rs=%21${rsindexarray[8]}&rule=RPC_$rand"
```

5

References

Unless otherwise specified, the following documents can be found at https://docs.progress.com.

RESTful API, Interface Description