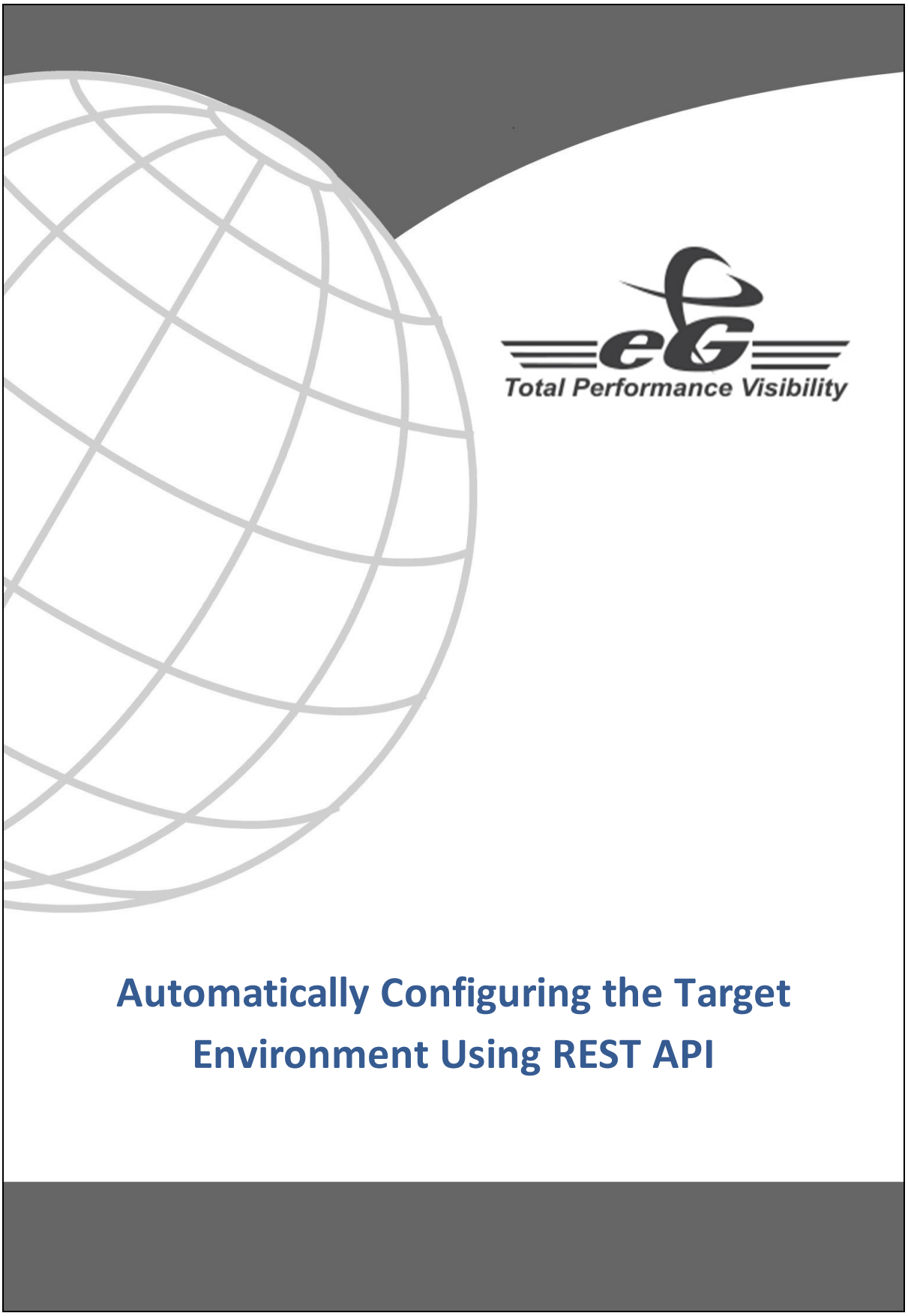

Automatically Configuring the Target Environment Using REST API

Table of Contents

Table of Figures

Chapter 1: Introduction

eG Enterprise is a 100%, web-based management console that allows users to view performance metrics collected from a target infrastructure. Users with administrative rights can configure the infrastructure that needs to be monitored. Configuration typically involves a sequence of tasks that prepares the environment for monitoring - this includes identifying and adding the components to be monitored, configuring the tests pertaining to these components, setting thresholds, configuring additional external and remote agents for the environment, etc. Typically, a user must login to the web-based eG administrative interface as an admin user in order to perform the above-mentioned tasks.

To perform critical configuration tasks on the eG manager without logging into the eG manager, eG Enterprise previously offered only an eG CLI capability. However, to keep pace with the growth observed in the technology world, eG REST API capability is also available for a similar purpose.

A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. It is based on representational state transfer (REST) technology, an architectural style and approach to communications often used in web services development.

From any REST client, administrators can hit the URL of the eG manager using the HTTP POST method to connect to the manager and perform administrative tasks on it. Moreover, using the eG REST API, administrators can also retrieve analytical data (for e.g., alarms raised in the target environment, the detailed diagnosis data of a chosen measure, health of the components in the target environment) from the eG manager. This information can be integrated with other management portals. Commands can also be executed in bulk using this eG REST API.

1.1 What does the REST API enable?

- Ability to automate admin activities (e.g., auto provision monitoring when a VM is spun up)
- Extract and analyze performance metrics automatically
- Integration with other management portals to provide a seamless user interface
- Integration and consolidation with asset / configuration tracking systems

Chapter 1: How Does eG REST API Work?

From any REST Client, administrators can perform critical configuration tasks on the eG manager.

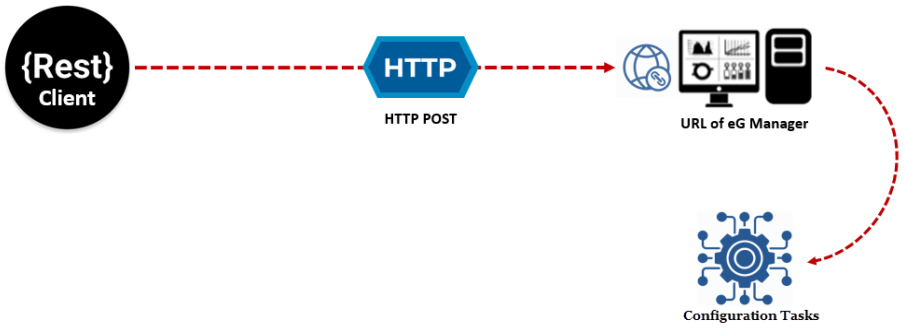

Figure 1.1: How the eG REST API works

1.2 Pre-Requisites for Configuring the Target Environment using the REST API

The eG REST API capability can perform critical configuration tasks on the eG manager only when the following pre-requisites are fulfilled:

- API consumer should have connectivity with eG manager
- API consumer requires a valid eG user account to access the API
- Provide valid password to be authenticated by the eG manager

1.3 Actions Supported by the eG REST API

The eG REST API can be used to perform the following actions:

- Orchestration
- Analytics and
- Miscellaneous Services

Each of these actions is explained in detail in the forthcoming chapters.

Chapter 2: The REST API Commands for Orchestration of eG Administrative Interface

To perform the administrative activities on the eG manager, users need to provide the following default Headers parameters. These parameters can also be set as a global variable in the REST Client.

- **managerurl**: The URL of the eG manager. Example: http://192.168.8.206:7077 (Note that this URL of the eG manager will be used in this document, wherever applicable)
- **user**: The user authorized to access the eG manager. Example: john
- **pwd**: The password for the user. Ensure that you provide an encrypted value of the password in this field. Note that the password should be encrypted in Base64 format.

  Note:

  The managerurl, user and pwd parameters (referred as Key values in REST Client) should be specified in the **Headers** tab of the REST Client.

The REST API supports commands for performing a bunch of administrative activities on the eG manager that are explained in detail in the following sections.

2.1 Adding Components

This API aids in adding new components to the eG Enterprise.

Note:

A few key values of the Body parameter are optional. These optional key values are mentioned separately in the below table.

URL: http://192.168.8.206:7077/api/eg/orchestration/addcomponent

Method: POST

Content-Type: application/json

Inputs to be Specified

<table>
<tr><th>Parameters</th><th>Key values</th><th>Example</th></tr>
<tr><td>Headers</td><td>managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
user: eG username or domain/eG username
pwd: Base64 encoded password</td><td rowspan="3">Example containing both Default and Optional key values:
{
"hostip":"192.168.10.175",
"componenttype":"Oracle Database",
"componentname":"ora8",
"port":"1521" ,
"sid":"egoracle",
"ispassive":"no",
"externalagents":agent1"
}
Example containing both Default and Optional key values:
{
"hostip":"192.168.10.175",
"componenttype":"Oracle Database",
"componentname":"ora8",
"port":"1521" ,
"sid":"egoracle",
"ispassive":"no",
"externalagents":agent1"
}</td></tr>
<tr><td rowspan="2">Body</td><td>Default:
{
"hostip":"IP address of the component",
"componenttype":"ComponentType",
"componentname":"nick name of the component",
"port":"port at which the component listens"
}</td></tr>
<tr><td>Optional:
{
"sid":"comma-separated list of SIDs",
"'externalagents":"comma-separated list of external agents assigned to the server",
"agentless":"yes/no",
"os":"Operating System of the server",
"mode":"Mode using which metrics are collected",
"encrypttype":"Password/Keybased",
"keyfilename":"Key file name",</td></tr>
</table>

Parameters	Key values	Example
	"remoteagent":"The remote agent that monitors the target", "remoteport":"the port at which Rexec/SSH listens", "remoteuser":"Valid user name on the target", "remotepwd":"A valid password", "internalagentassignment":"yes/no", "internalagent":"IP address/nick name of the internal agent", "mtsenabled":"yes/no", "virtualenv":"yes/no", "virtualserver":"Virtual server name", "ispassive":"yes/no" }	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component has been added successfully."
		}

Failure Response

Type	Content
JSON	{ "Error": "Component already exist under this type."

Type	Content
	}

	{
	"Error": "Cannot add agent based component for this component type."
	
	}

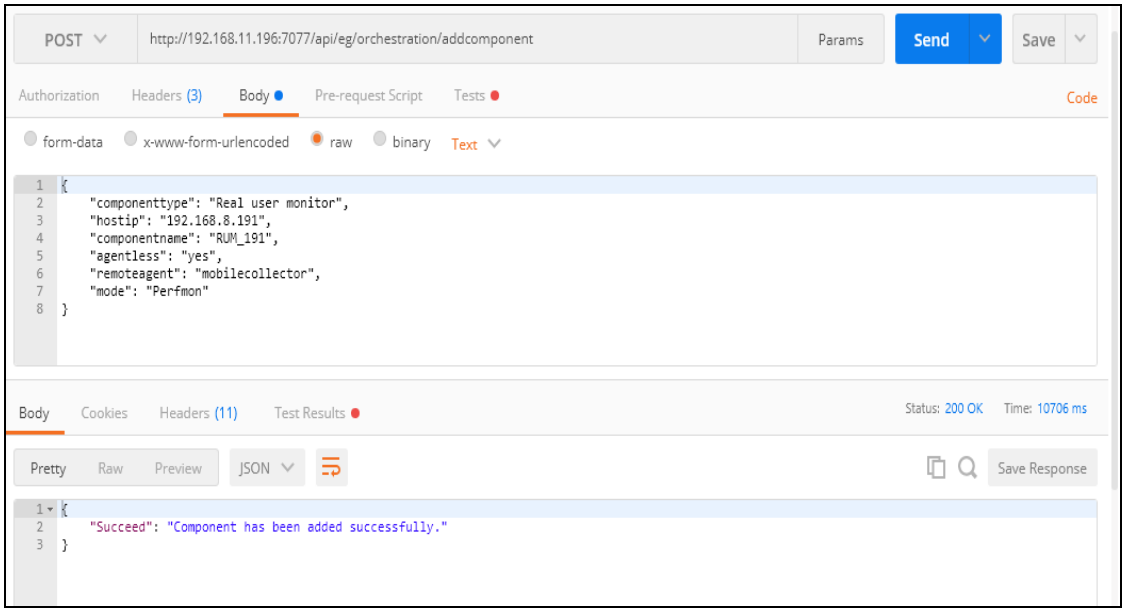

Figure 2.1: Example to add components using Postman REST Client

2.1.1 Adding Components using cURL

To add components through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/addcomponent" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'componenttype': 'ComponentType', 'hostip': 'IP
address of the component', 'componentname': 'nick name of the component', 'port': 'port
at which the component listens', 'agentless': 'yes/no', 'remoteagent': 'The remote agent
that monitors the target', 'mode': 'Mode using which metrics are collected', 'sid':
'comma-separated list of SIDs', 'externalagents': 'comma-separated list of external
agents assigned to the server', 'os': 'Operating System of the server', 'encrypttype':
'Password/Keybased', 'remoteuser': 'Valid user name on the target', 'remoteport': 'the
port at which Rexec/SSH listens', 'remotepwd': 'A valid password',
'internalagentassignment': 'yes/no', 'internalagent': 'IP address/nick name of the
internal agent', 'mtsenabled': 'yes/no', 'virtualenv': 'yes/no', 'virtualserver':
'Virtual server name', 'ispassive':'yes/no'}"
```

Note that the command specified above contains both the Default and Optional key values. Figure 2.2 shows an example of adding components using cURL.

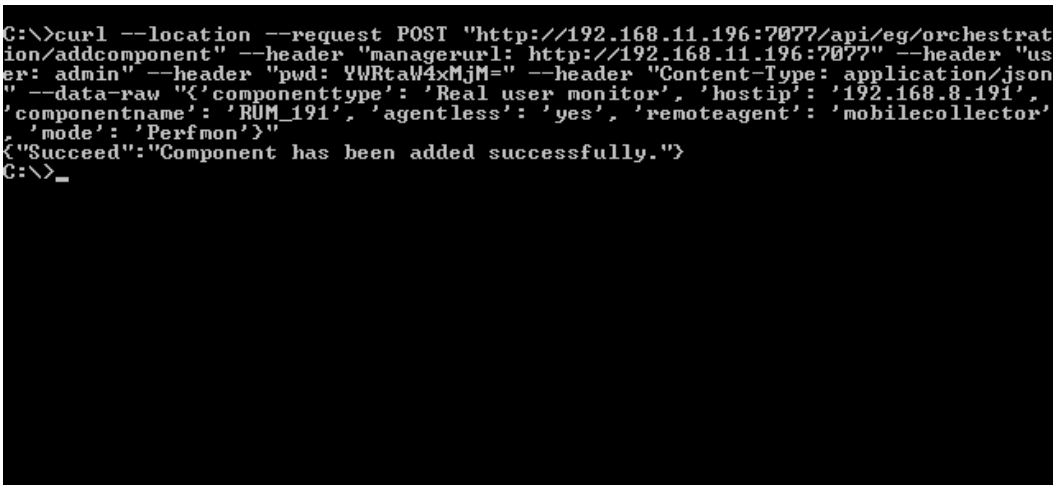

Figure 2.2: Adding components using cURL

2.2 Adding External Agents

Use this REST API to add external agents to the target eG manager.

URL: http://192.168.8.206:7077/api/eg/orchestration/addexternalagent

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"hostip":"192.168.8.191",
		"agentname":"egdp119"
		}
		Example containing both Default and Optional key values:
		{
		"hostip":"192.168.8.191",
		"agentname":"egdp119",
		"clientemulation":"yes"
		}

Body	**Default:**
	{
	"hostip":"IP address of the component",
	"agentname":"Agent name"
	}	

	Optional:
	{
	"clientemulation":"yes/no"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "External agent has been added successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Space is not allowed in component name/agent name."
	}

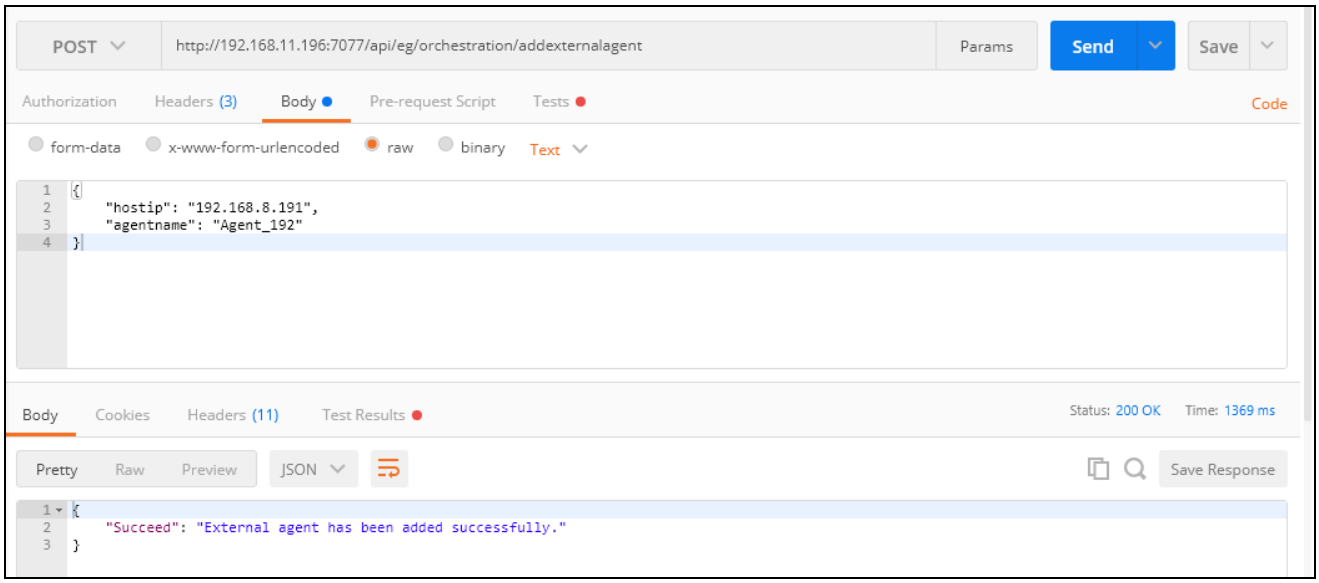

Figure 2.3: Example to add an external agent using Postman REST Client

2.2.1 Adding External Agents using cURL

To add external agents through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/addexternalagent" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{ 'hostip': 'IP address of the component',
'agentname': 'Agent name', 'clientemulation': 'yes/no'}"
```

Note that the command specified above contains both Default and Optional key values.

Figure 2.4 shows an example of adding external agents using cURL.

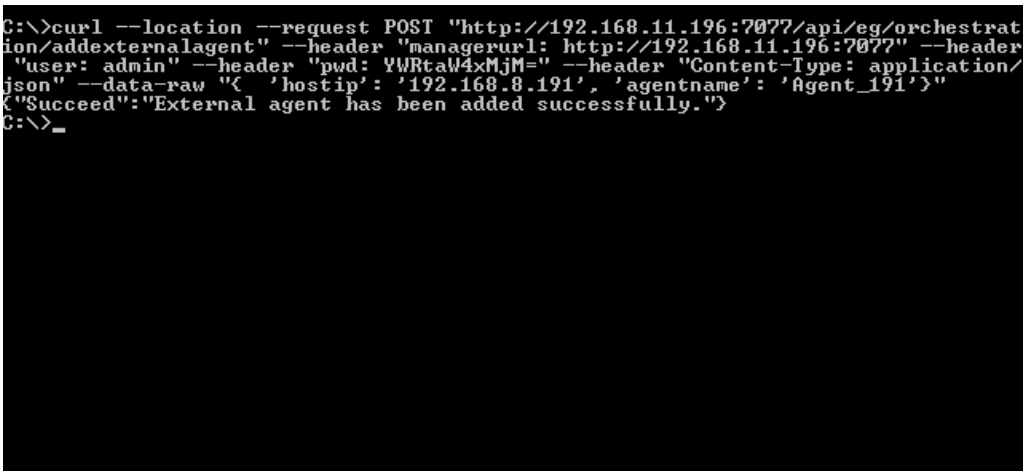

Figure 2.4: Adding an external agent using cURL

2.3 Adding Groups

Use this API to add a group comprising of one/more components to the eG Enterprise.

URL: http://192.168.8.206:7077/api/eg/orchestration/addgroup

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"groupname":"group_X",
		"elements":"Citrix Logon Simulator:TestLogon,Microsoft SQL:MSSQL:1433"
		}

Body	Default: { "groupname":"Group	

Parameters	Key values	Example
	name", "elements":"comma-separated list of elements" }	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Group has been added successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Required group details"
	}

	{
	"Error": "One or more elements do not exist or not available to associate. Invalid elements"
	}

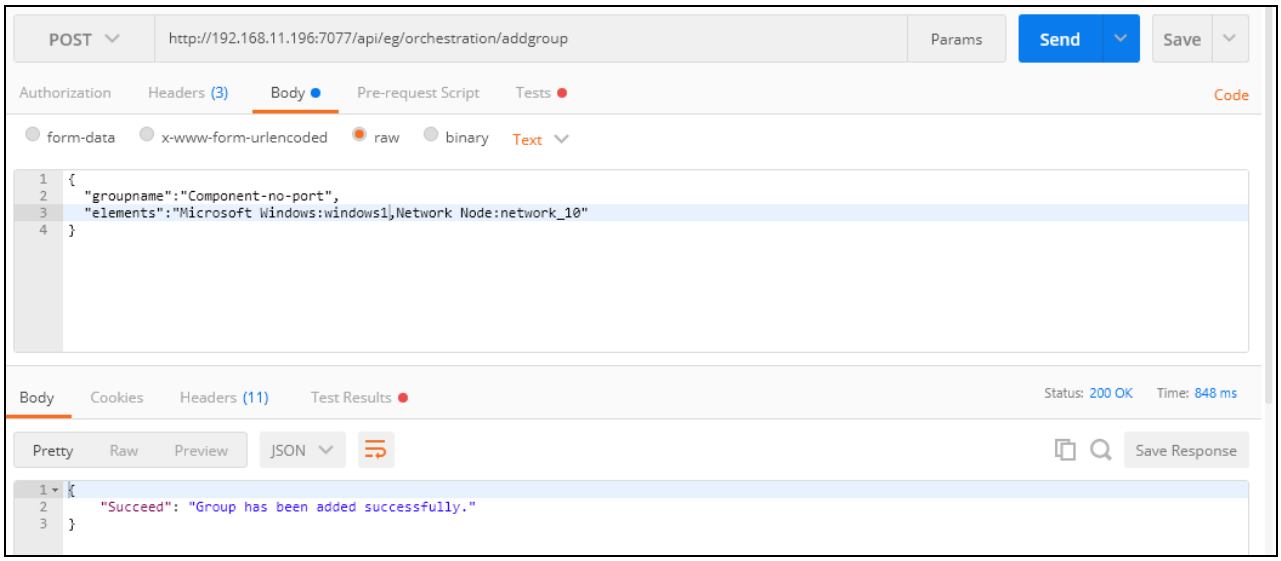

Figure 2.5: Example to add a group using Postman REST Client

2.3.1 Adding a Group using cURL

To add a group through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/addgroup" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'groupname': 'Group name', 'elements': 'comma-
separated list of elements'}"
```

Figure 2.6 shows an example of adding a group using cURL.

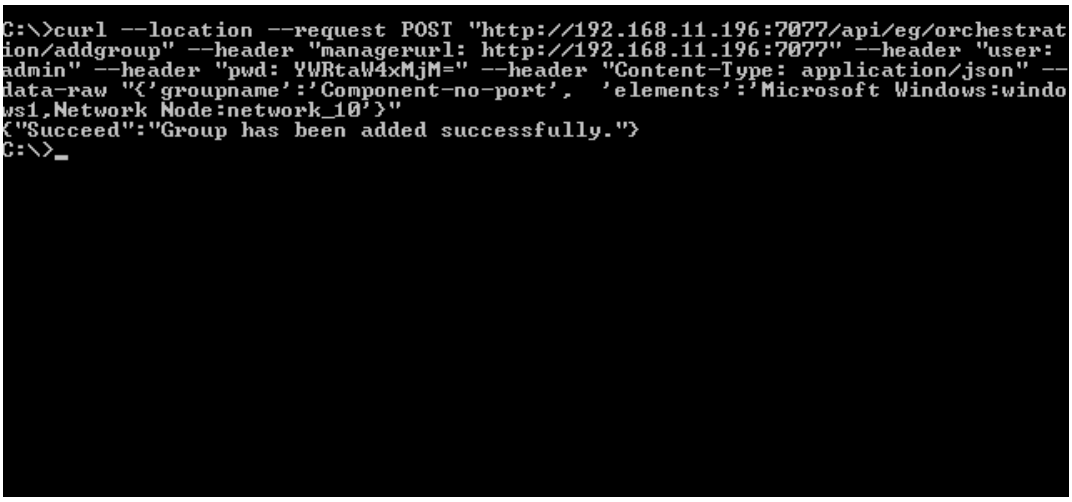

Figure 2.6: Adding a group using cURL

2.4 Adding Maintenance Policies

This API helps administrators add maintenance policies to the eG manager.

URL: http://192.168.8.206:7077/api/eg/orchestration/addmaintenancepolicy

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"policyname":"QMPPolicy",
		"timefrequency":" Thursday=10:15-11:15"
		}

Body	**Default:** { "policyname":"Policy name", "timefrequency":"[Daily]/[First day of month]/[Last day of month]/	

Parameters	Key values	Example
	[Sunday/Monday/Tuesday/Wednesday/ Thursday/Friday/ Saturday]/ Start Date-End Date]=Start Time-End Time" }	

Note:

The format for **Start Date** and **End Date** is *MM/DD/YYYY*

The format for **Start Time** and **End Time** is *Hours:Minutes*

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Maintenance policy added successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Invalid Time frequency."
	}

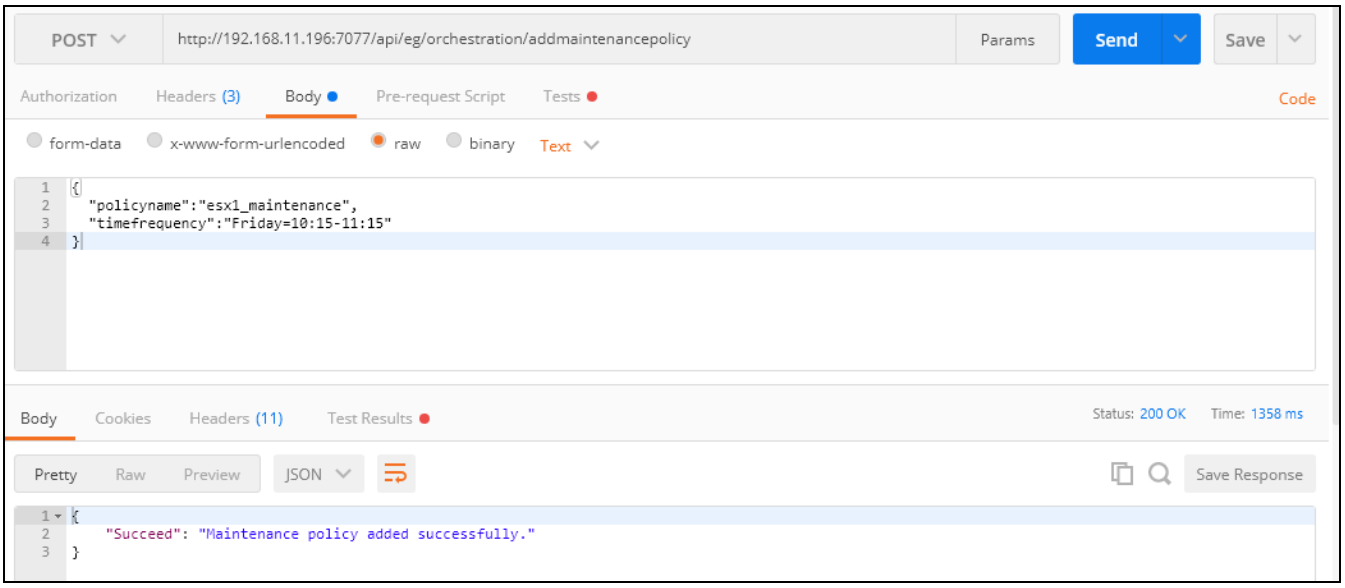

Figure 2.7: Example to add a maintenance policy using Postman REST Client

2.4.1 Adding Maintenance Policies using cURL

To add a maintenance policy through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/addmaintenancepolicy" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'policyname': 'Policy name',
'timefrequency': '[Daily]/[First day of month]/[Last day of month]/
[Sunday/Monday/Tuesday/Wednesday/Thursday/Friday/Saturday]/Start Date-End Date]=Start
Time-End Time'}"
```

Figure 2.8 shows an example of adding a maintenance policy using cURL.

Figure 2.8: Adding a maintenance policy using cURL

2.5 Adding Remote Agents

Use this API to add remote agents for monitoring to the eG manager.

URL: http://192.168.8.206:7077/api/eg/orchestration/addremoteagent

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Header	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"hostip":"192.168.8.192",
		"agentname":"remote191"
		}

Body	Default: { "hostip":"Host IP",	

Parameters	Key values	Example
	"agentname":"Remote Agent name" }	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Remote agent has been added successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "The agent name you are trying to add already exists. Please use another agent name."
	}

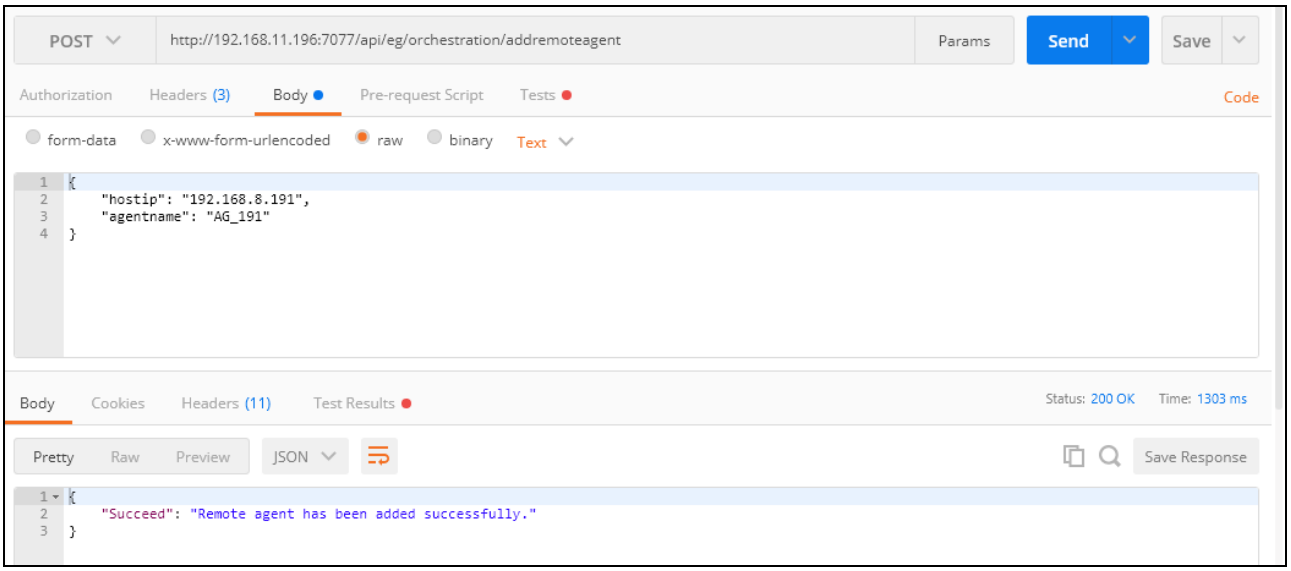

Figure 2.9: Example to add a remote agent using Postman REST Client

2.5.1 Adding Remote Agents using cURL

To add a remote agent through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/adaddremoteagent" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'hostip': 'Host IP', 'agentname': 'Remote
Agent name'}"
```

Figure 2.10 shows an example of adding a remote agent using cURL.

Figure 2.10: Adding a remote agent using cURL

2.6 Adding a User

Use this API to add a user to the eG manager.

Note:

A few key values of the Body parameter are optional. These optional key values are mentioned separately in the below table.

URL: http://192.168.8.206:7077/api/eg/orchestration/adduser

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"userid":"john",
		"userrole":"monitor",
		"password":"******",
		"expirydate":"05/20/2021"
		}
		Example with both Default and Optional key values:
		{
		"userid":"john",
		"userrole":"monitor",
		"password":"******",
		"expirydate":"05/20/2021",
		"alarmsbymail":"critical",
		"to":"saranya1@eginnovations.com",
		"cc":"saran1@eginnovations.com,

Parameters	Key values	Example

Body	**Default:**
	{
	"userrole":"User role",
	"userid":"User ID",
	"password":"Password",
	"expirydate":"MM/DD/YYYY"
	}
		9840391695",
		"bcc":"shara@eginnovations.com"
		}

	Optional:
	{
	"alarmsbymail":"Critical/Major/Minor/All",
	"to":"comma-separated list of Mail IDs/Mobile numbers",
	"cc":"comma-separated list of Mail IDs/Mobile numbers",
	"bcc":"comma-separated list of Mail IDs/Mobile numbers"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "User has been created successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Please provide a user role."
	}

	{
	"Error": "Please enter a valid date."
	}

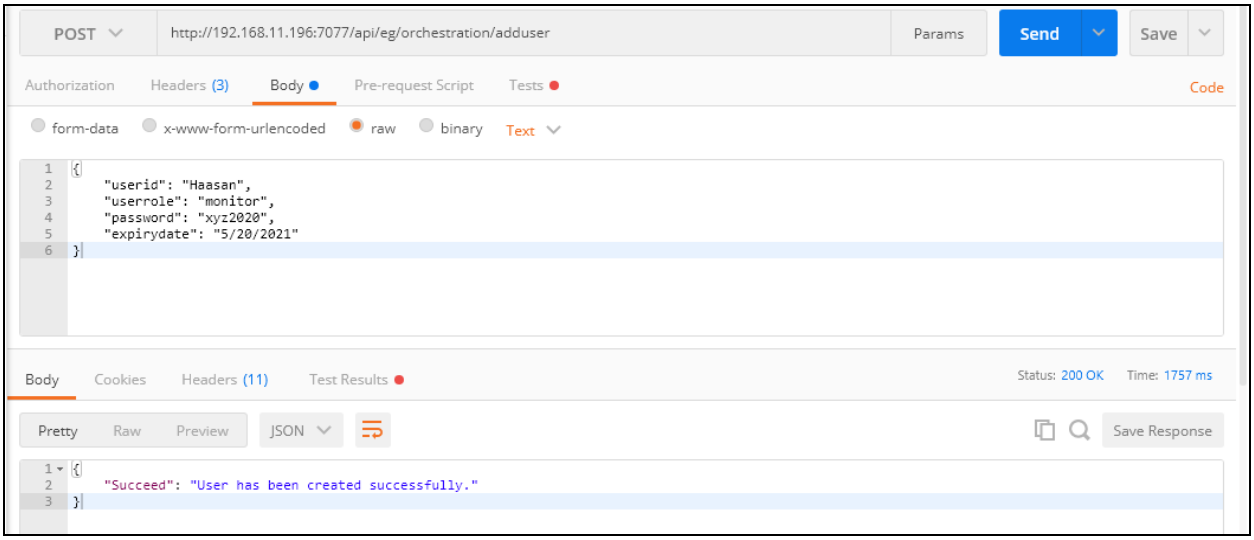

Figure 2.11: Example to add a new user using Postman REST Client

2.6.1 Adding a User using cURL

To add a user through the REST API using cURL, specify the command in the following format:

```
curl --location --request POST "http://<eG Manager IP:Port>/api/eg/orchestration/adduser"
-H "managerurl:http://<eG Manager IP:Port>"-H "user:<eG username or domain/eG username>"
-H "pwd:Base64 encoded password" -H "Content-Type: application/json" --data-raw "
{'userrole': 'User role', 'userid': 'User ID', 'password': 'Password', 'expirydate':
'MM/DD/YYYY', 'alarmsbymail':"Critical/Major/Minor/All', 'to': 'comma-separated list
of Mail IDs/Mobile numbers', 'cc': 'comma-separated list of Mail IDs/Mobile numbers',
'bcc': 'comma-separated list of Mail IDs/Mobile numbers'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.12 shows an example of adding a user using cURL.

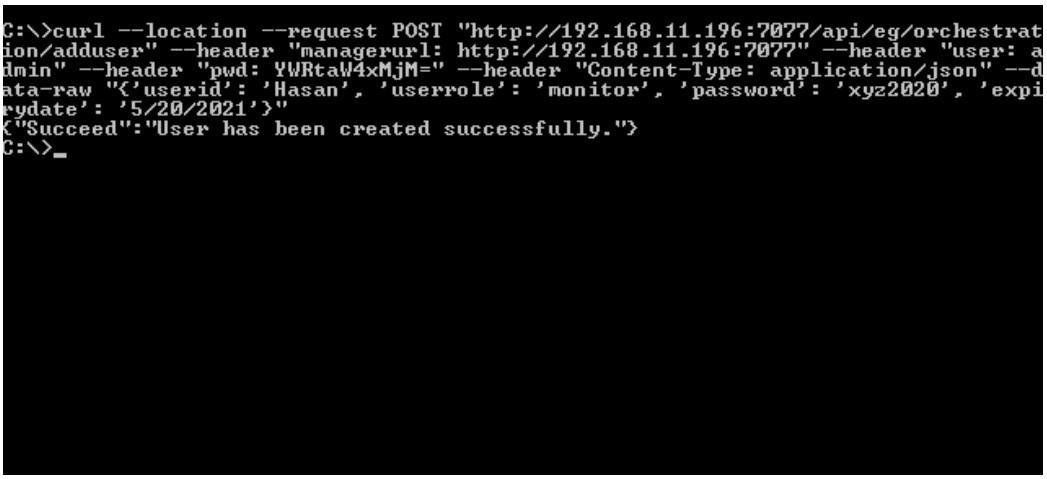

Figure 2.12: Adding a user using cURL

2.7 Adding a Zone

Use this API to add a zone to the eG manager.

Note:

A few key values of the Body parameter are optional. These optional key values are mentioned separately in the below table.

URL: http://192.168.8.206:7077/api/eg/orchestration/addzone

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with both Default and Optional key values:
		{
		"zonename":"eastzone",
		"elements":"IIS Web:web_ 2:80,group:dbgroup",
		"displayimage":"Web",
		"autoassociate":"yes"
		}

Parameters	Key values	Example

Body	**Default:**
	{
	"zonename":"Zone name",
	"elements":"comma-separated list of elements"
	}
		Example with Default key values:
		{
		{
		"zonename":"westzone",
		"elements":"IIS Web:web_2:80",
		}

	Optional:
	{
	"displayimage":"Display image",
	"autoassociate":"yes/no"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Zone has been added successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "One or more elements do not exist or not available to associate. Invalid elements"
	}

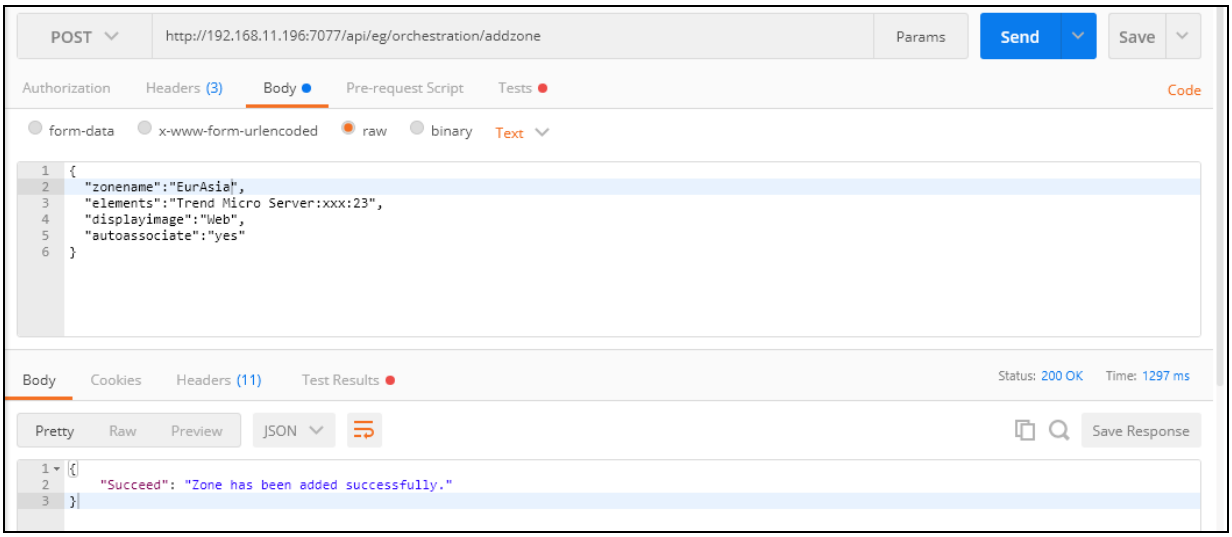

Figure 2.13: Example to add a zone using Postman REST Client

2.7.1 Adding a Zone using cURL

To add components through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager IP:Port>/api/eg/orchestration/addzone"
-H "managerurl:http://<eG Manager IP:Port>"-H "user:<eG username or domain/eG username>"
-H "pwd:Base64 encoded password" -H "Content-Type: application/json" --data-raw "
{'zonename':'Zone name', 'elements':'comma-separated list of elements',
'displayimage':'Display image', 'autoassociate':'yes/no'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.14 shows an example of adding a zone using cURL.

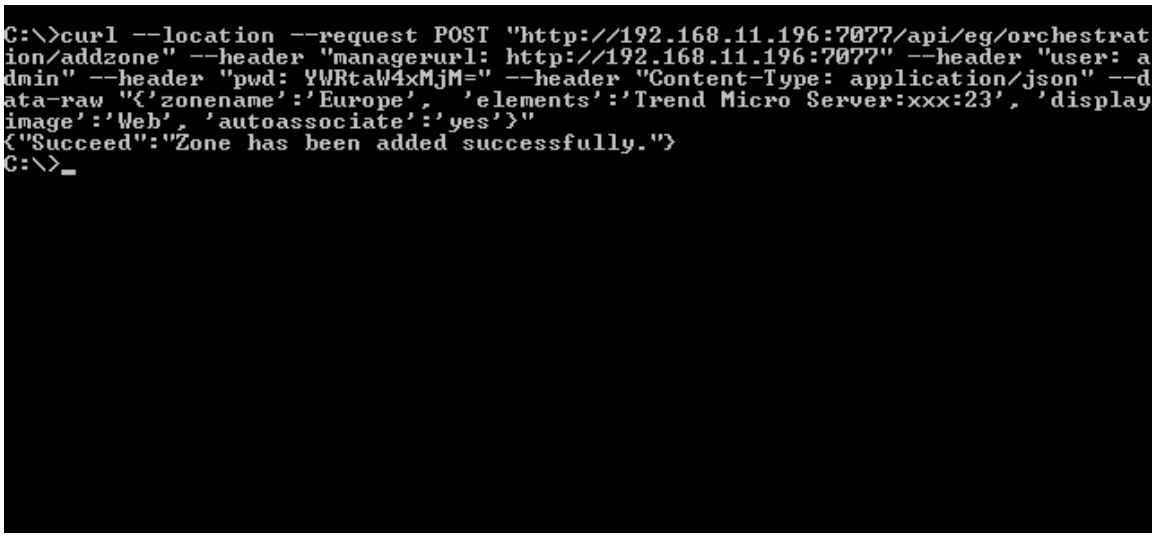

Figure 2.14: Adding a zone using cURL

2.8 Assigning an Agent

Use this command to assign agents to a manager in a redundant setup.

URL: http://192.168.8.206:7077/api/eg/orchestration/assignagents

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"managerip":"192.168.8.191",
		"agentname":"egdp119,egdp201"
		}

Body	{ "managerip":"IP of	

Parameters	Key values	Example
	the eG manager to which agents are to be assigned", "agents":"comma-separated list of agents" }	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "one or more agents assigned successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "One or more agents in not valid. Please give valid agent name."
	}

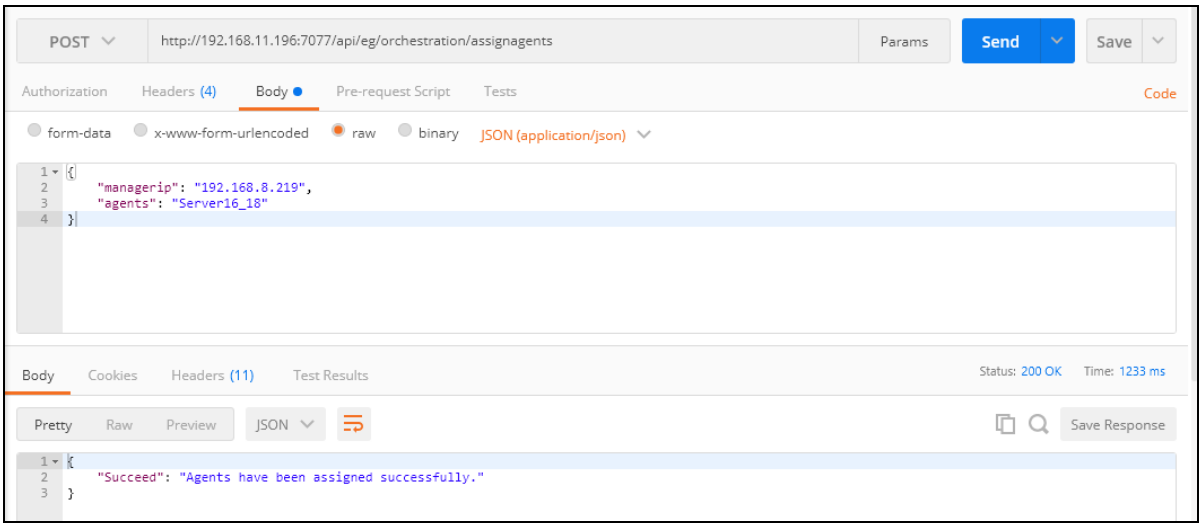

Figure 2.15: Assigning an agent to the eG manager in a redundant setup using Postman REST Client

2.8.1 Assigning an Agent using cURL

To assign an eG agent to an eG manager in a redundant setup through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/unassignagents" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'managerip':'IP of the eG manager from
which agents are to be delinked', 'agents':'Comma-separated list of agents'}"
```

Figure 2.16 shows an example of assigning the eG agent to an eG manager in a redundant setup using cURL.

Figure 2.16: Assigning an agent to the eG manager in a redundant setup using cURL

2.9 Assigning a Maintenance Policy

Use this API to associate/dissociate a maintenance Policy to a Component/Host/Test/Test For Host/Test For Component/Test For component type.

URL: http://192.168.8.206:7077/api/eg/orchestration/assignmaintenancepolicy

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key Values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"policyname":"QMP1",
		"associatefor":"Component",
		"componentsby": "Component Type",
		"componenttype":"microsoft windows",
		"associateelements":"windowsos191 ,windows195"
		}

Body	Default: { "policyname":"Policy name"	

Parameters	Key Values	Example
	"associatefor":"Host/Component/Test/ Test For Host/Test For component/ Test For component type" "componentsby":"Zone/Segment /Service/Component Type" "zone":"Zone name" "segment":"Segment name" "service":"Service name" "componenttype":"Component type" "test":"Test name" "associateelements":"comma-separated list of elements" "disassociateelements":"comma-separated list of elements" }	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Maintenance policy has been associated/dissociated successfully."
		}

Failure Response

Type	Content
JSON	{ "Error": "Element(s) you are trying to add does/do not exist."

Type	Content
	}

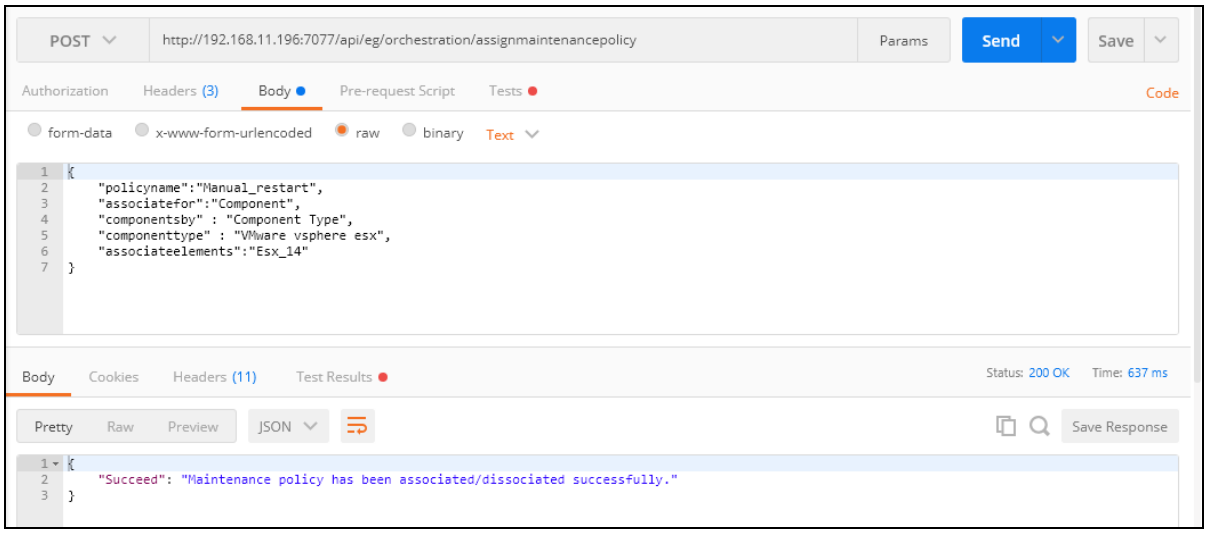

Figure 2.17: Assigning a Maintenance Policy using Postman REST Client

2.9.1 Assigning a Maintenance Policy using cURL

To assign a maintenance policy through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/assignmaintenancepolicy" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'policyname':'Policy name',
'associatefor':'Host/Component/Test/Test For Host/Test For component/Test For component
type', 'componentsby':'Zone/Segment/Service/Component Type', 'zone':'Zone name',
'segment':'Segment name', 'service':'Service name', 'componenttype':'Component type',
'test':'Test name', 'associateelements':'comma-separated list of elements',
'disassociateelements':'comma-
separated list of elements'}"
```

Figure 2.18 shows an example of assigning a maintenance policy using cURL.

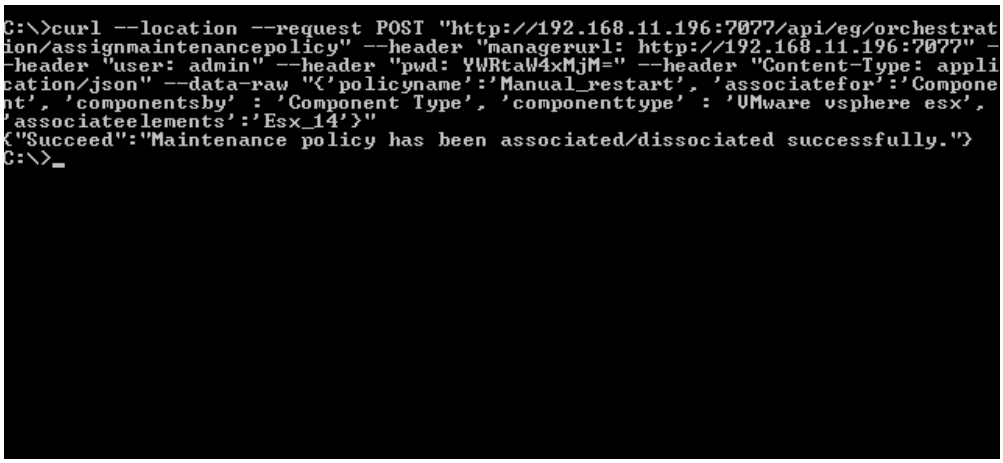

Figure 2.18: Assigning a maintenance policy using cURL

2.10 Associating Components to User

Using this API, administrators can associate components to a user.

Note:

A few key values of the Body parameter are optional. These optional key values are mentioned separately in the below table.

URL: http://192.168.8.206:7077/api/eg/orchestration/associatecomponentstouser

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default Key Values:
		{
		"userid":"john",
		"componenttype":"microsoft windows",
		"components":"dev153,win155,win156"
		}
		Example with both Default and Optional Key Values:

Parameters	Key values	Example

Body	**Default:**
	{
	"userid":"User ID",
	"componenttype":"Component type",
	"components":"comma-separated list of Nick names",
	}
		{
		"userid":"john",
		"componenttype":"microsoft windows",
		"components":"dev153,win155,win156"
		"autossociatetype":"yes"
		}

	Optional:
	{
	"autoassociatetype":"yes/no"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "One or more components have been associated successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "One or more component names do not exist."
	}

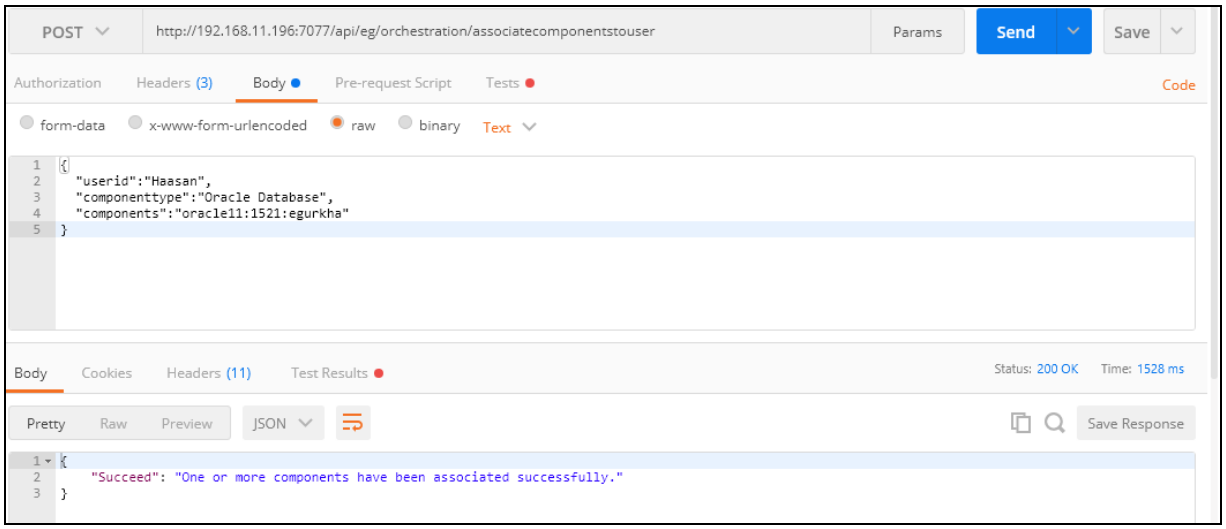

Figure 2.19: Example to associate components to a user using Postman REST Client

2.10.1 Associating Components to User using cURL

To associate components to a user through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/associatecomponentstouser" -H "managerurl:http://<eG
Manager IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded
password" -H "Content-Type: application/json" --data-raw "{'userid':'User ID',
'componenttype':'Component type', 'components':'comma-separated list of Nick names',
'autoassociatetype':'yes/no'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.20 shows an example of associating components to a user using cURL.

Figure 2.20: Associating components to a user using cURL

2.11 Deleting a Component

Use this API to delete a component from the eG manager.

URL: http://192.168.8.206:7077/api/eg/orchestration/deletecomponent

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"componenttype":"Microsoft SQL",
		"componentname":"MSSQL",
		"port":"1433"
		}
		Example with both Default and Optional Key Values:
		{
		"componenttype":"Oracle Database",

Parameters	Key values	Example

Body	**Default:**
	{
	"componenttype":"Component type",
	"componentname":"The nick name of the component",
	"port":"Port",
	}
		"componentname":"oradb4",
		"port":"1521"
		"sid":"egora"
		}

	Optional:
	{
	"sid":"SID"
	}	

Note:

If an Oracle Database server is added with multiple SIDs, then the eG Enterprise system will monitor each SID as a different Oracle Database server. Therefore, while removing an Oracle Database server that supports multiple SIDs, you cannot issue a single command to remove all the SIDs at one shot. Instead, this command should be invoked separately for each SID.

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component has been removed successfully."
		}

Failure Response

Type	Content
JSON	{

Type	Content
	"Error": "The selected component does not exist." }

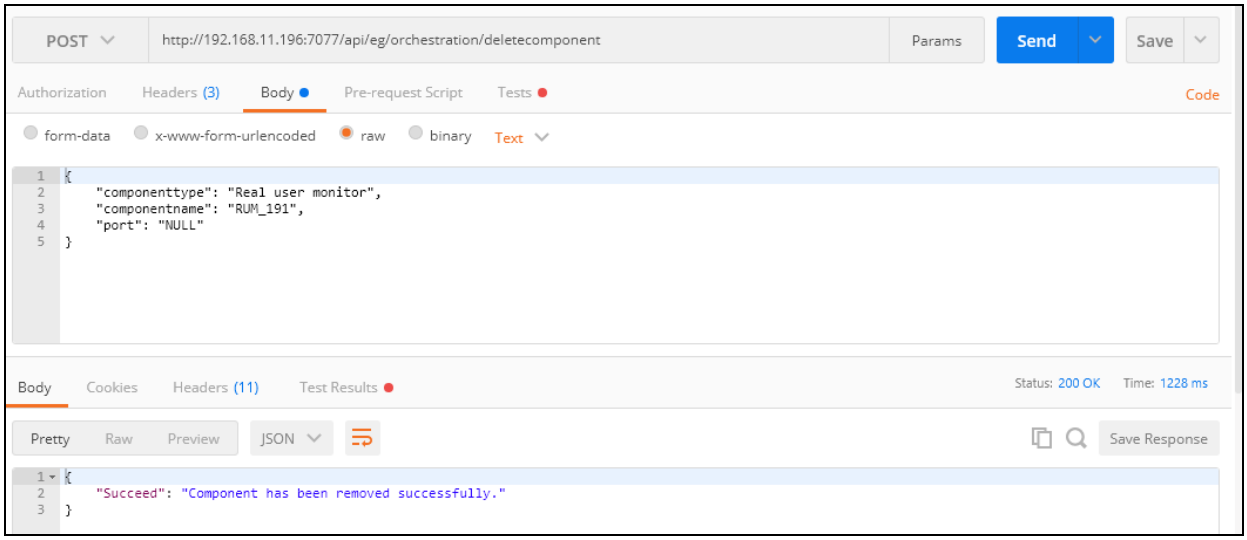

Figure 2.21: Deleting a Component using Postman REST Client

2.11.1 Deleting a Component using cURL

To delete a component through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/deletecomponent" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'componenttype': 'ComponentType',
'componentname': 'nick name of the component', 'port': 'port at which the component
listens', 'sid': 'SID'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.22 shows an example of deleting a component using cURL.

Figure 2.22: Deleting a component using cURL

2.12 Deleting an External Agent

Using this API, administrators can delete an external agent from the eG manager.

URL: http://192.168.8.206:7077/api/eg/orchestration/deleteexternalagent

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"agentname":"ext191"
		}

Body	Default: {	

Parameters	Key values	Example
	"agentname":"Agent name" }	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "External agent has been deleted successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "The external agent you are trying to delete does not exist."
	}

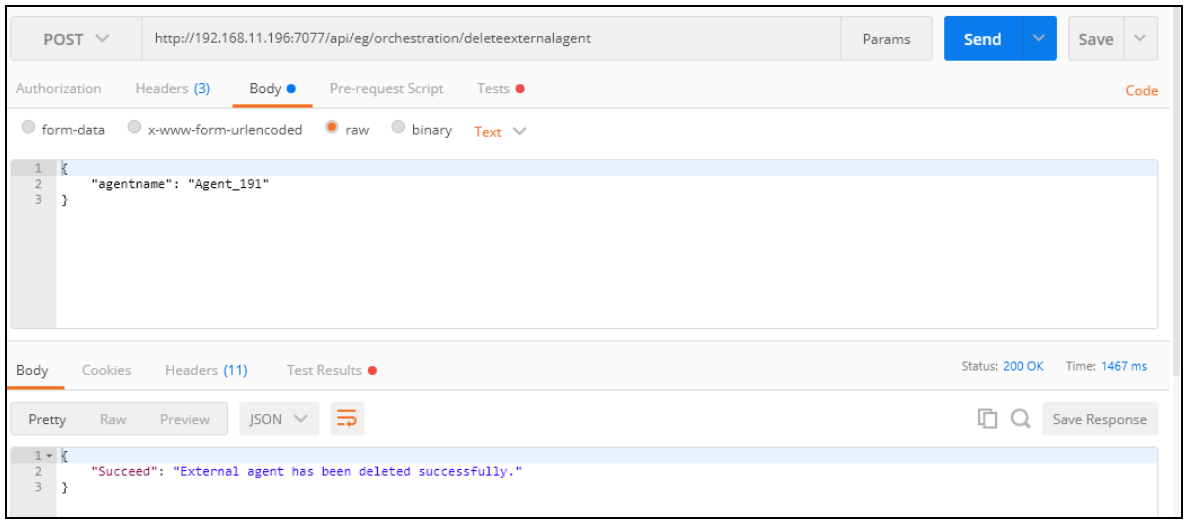

Figure 2.23: Deleting an external agent using Postman REST Client

2.12.1 Deleting an External Agent using cURL

To delete an external agent through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/deleteexternalagent" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'agentname':'Agent name'}"
```

2.12 shows an example of deleting an external agent using cURL.

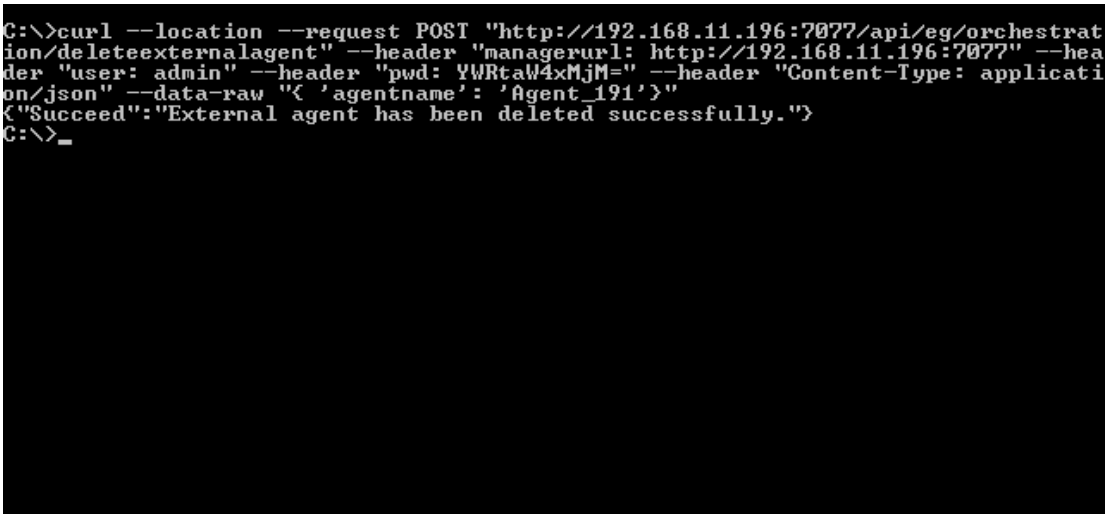

Figure 2.24: Deleting an external agent using cURL

2.13 Deleting a Group

Administrators can use this API to delete a group from the eG manager.

URL: http://192.168.8.206:7077/api/eg/orchestration/deletegroup

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"groupname":"mynewgroup,egdbgroup"
		}

Body	**Default:**
	{
	"groupname":"comma-separated list of groups"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Group has been deleted successfully."
		}

Failure Response

Type	Code	Content

JSON	401 UNAUTHORIZED	{
		"Error": "One or more given groups do not exist or is not associated to any zone/segment/services. Invalid groups : <comma-separated list of group names>"
		}

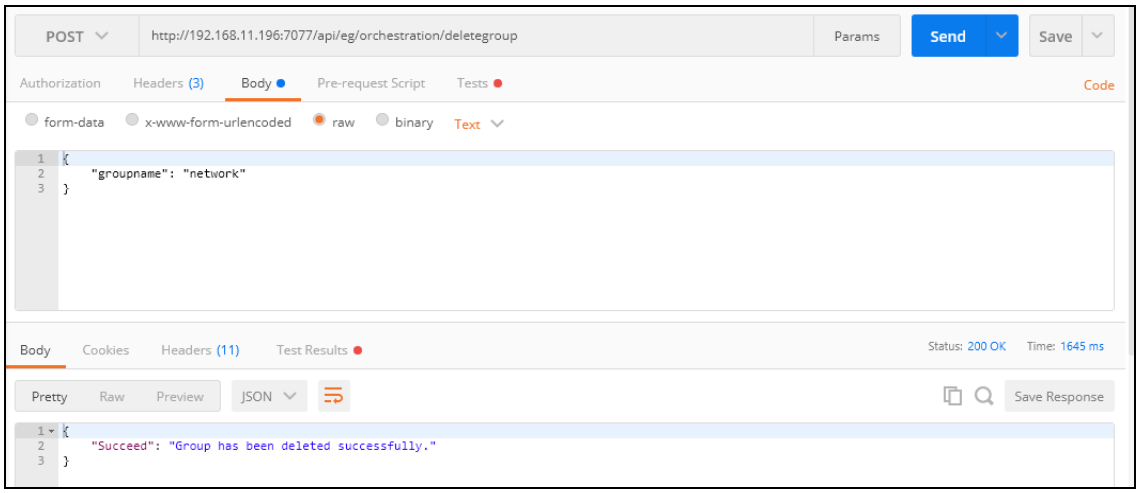

Figure 2.25: Deleting a Group using Postman REST Client

2.13.1 Deleting a Group using cURL

To delete a group through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/deletegroup" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'groupname':'comma-separated list of groups'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.26 shows an example of deleting a group using cURL.

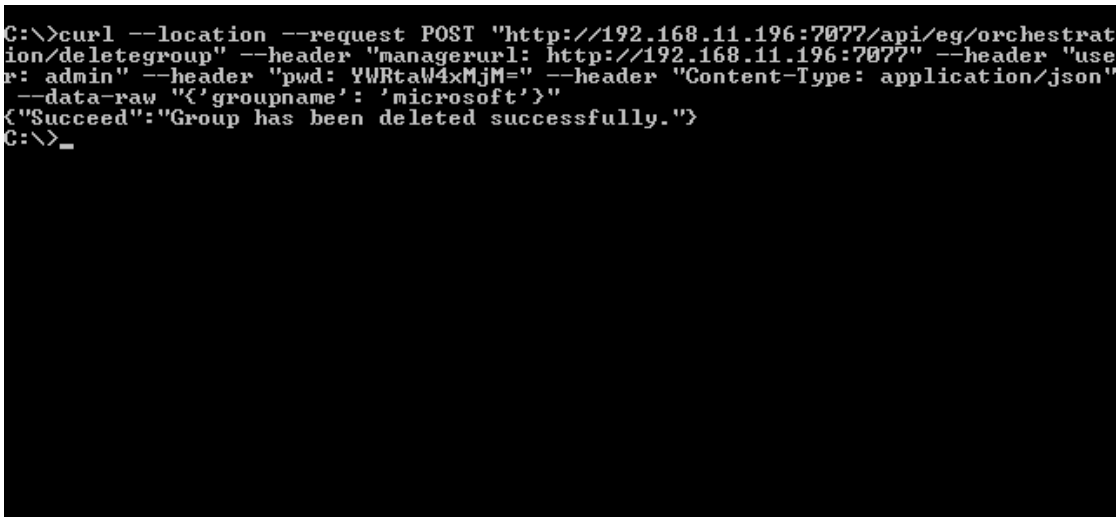

Figure 2.26: Deleting a group using cURL

2.14 Deleting a Maintenance Policy

Use this API to delete a maintenance policy configured in the eG manager.

URL: http://192.168.8.206:7077/api/eg/orchestration/deletemaintenancepolicy

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"policyname":"QMP1,QMP2"
		}

Body	Default: {	

Parameters	Key values	Example
	"policyname":"comma-separated list of maintenance policies" }	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Maintenance policy deleted successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Maintenance policy does not exist."
	}

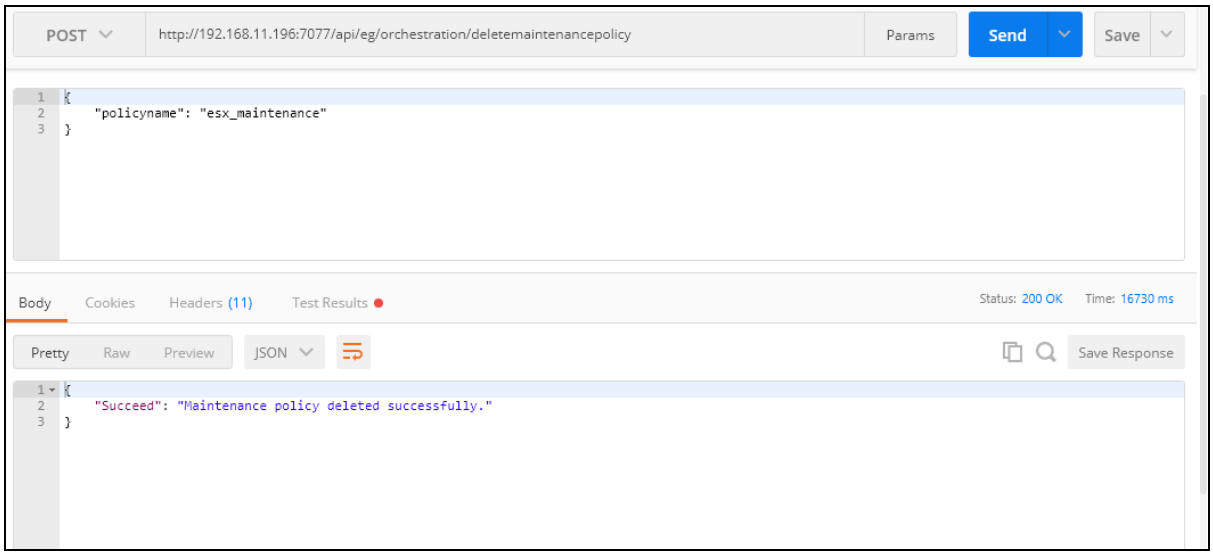

Figure 2.27: Example to delete a maintenance policy using Postman REST Client

2.14.1 Deleting a Maintenance Policy using cURL

To delete a maintenance policy through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/deletemaintenancepolicy" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'policyname':'comma-separated list of
maintenance policies'}"
```

Figure 2.28 shows an example of deleting a maintenance policy using cURL.

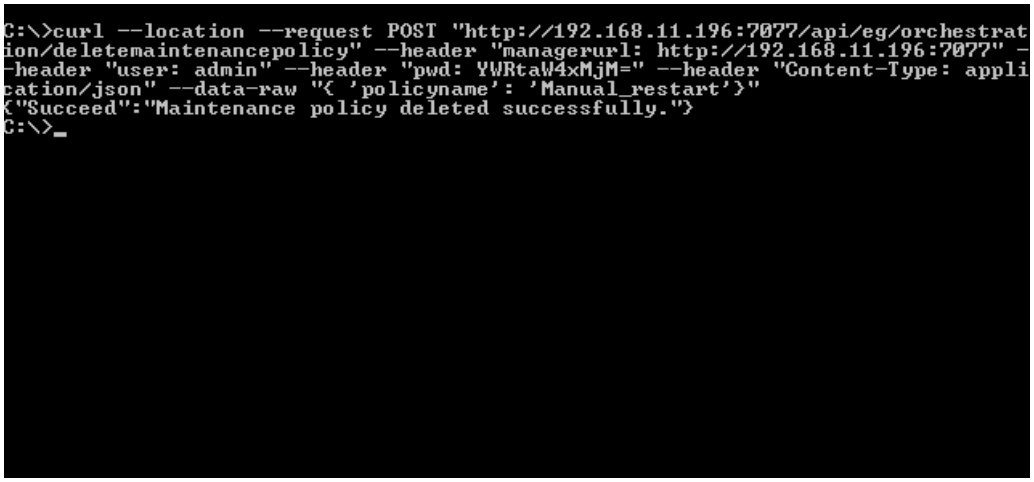

Figure 2.28: Deleting a maintenance policy using cURL

2.15 Deleting a Remote Agent

Use this API to delete a remote agent from the eG manager.

URL: http://192.168.8.206:7077/api/eg/orchestration/deleteremoteagent

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"agentname":"AG_191"
		}

Body	**Default:**
	{
	"agentname":"Agent name"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Remote agent has been deleted successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "The remote agent you are trying to delete does not exist."
	}

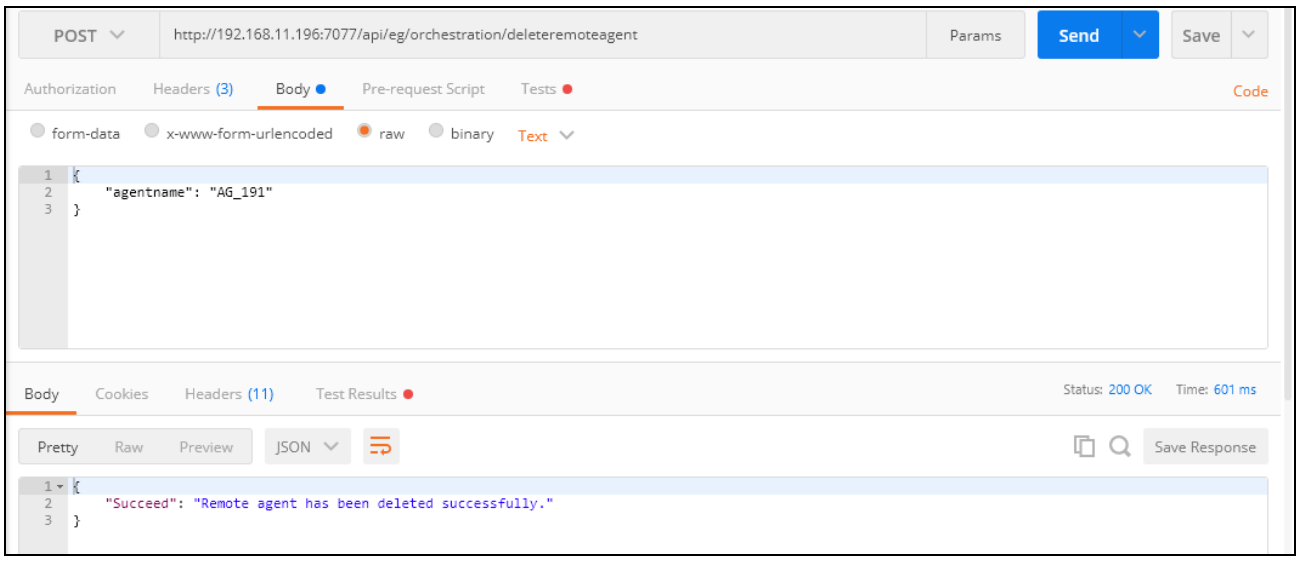

Figure 2.29: Deleting a remote agent using Postman REST Client

2.15.1 Deleting a Remote Agent using cURL

To delete a remote agent through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/deleteremoteagent" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'agentname':'Agent name'}"
```

Figure 2.30 shows an example of deleting a remote agent using cURL.

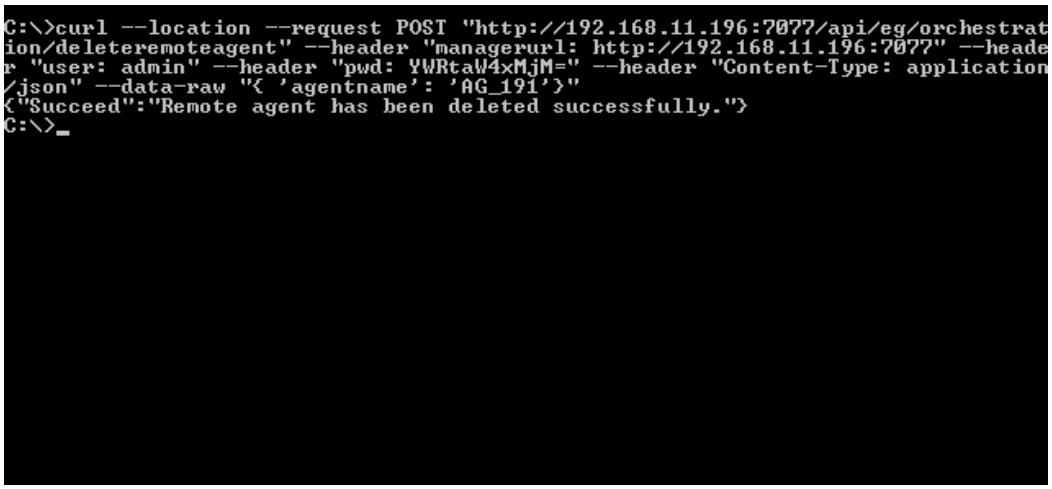

Figure 2.30: Deleting a remote agent using cURL

2.16 Deleting a User

Use this API to delete a user from the eG Enterprise.

URL: http://192.168.8.206:7077/api/eg/orchestration/deleteuser

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"userid":"john,kim,sarah"
		}

Body	**Default:** {	

Parameters	Key values	Example
	"userid":"comma-separated list of User IDs" }	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "User(s) has/have been deleted successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "One or more users do not exist."
	}

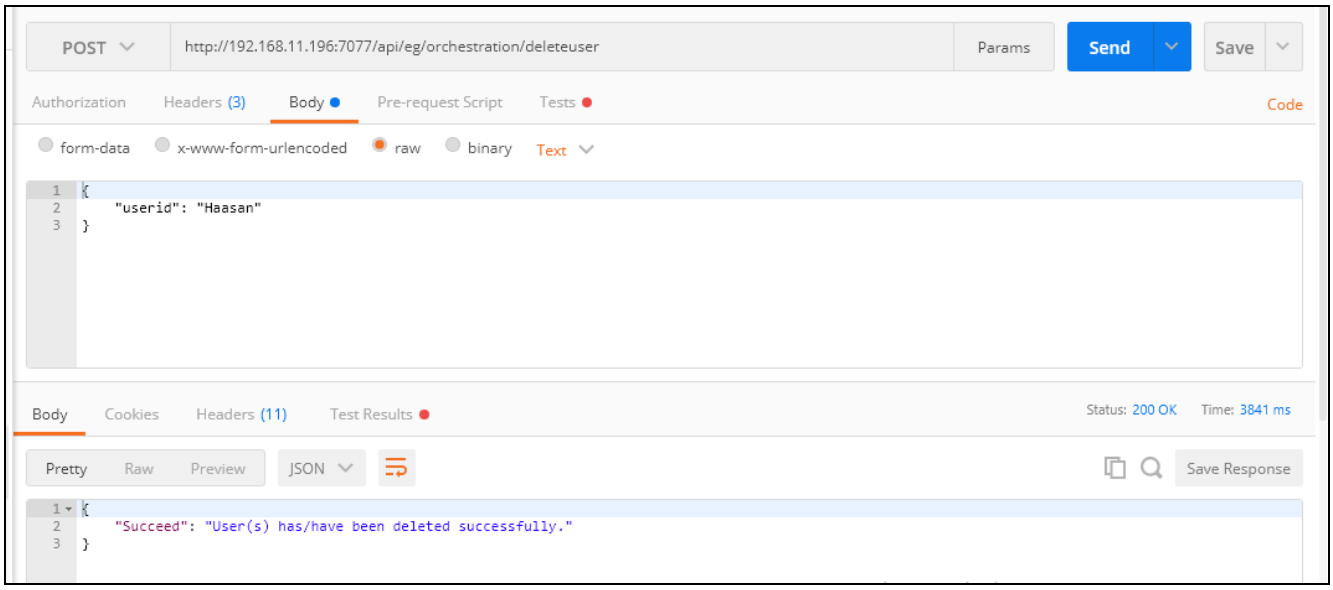

Figure 2.31: Deleting a User using Postman REST Client

2.16.1 Deleting a User using cURL

To delete a user through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/deleteuser" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'userid':'comma-separated list of User IDs'}"
```

Figure 2.32 shows an example of deleting a user using cURL.

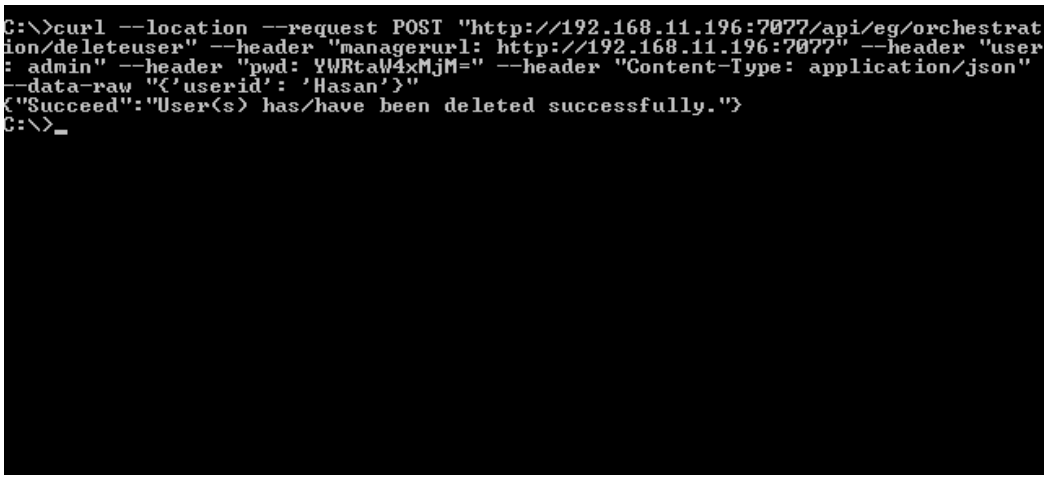

Figure 2.32: Deleting a user using cURL

2.17 Deleting a Zone

Use this API to delete a zone from the eG manager.

URL: http://192.168.8.206:7077/api/eg/orchestration/deletezone

Method: POST

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"zonename":"EurAsia,globalwest"
		}

Body	**Default:**
	{
	"zonename":"comma-separated list of zones"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Zone has been deleted successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "One or more given zone names do not exist."
	}

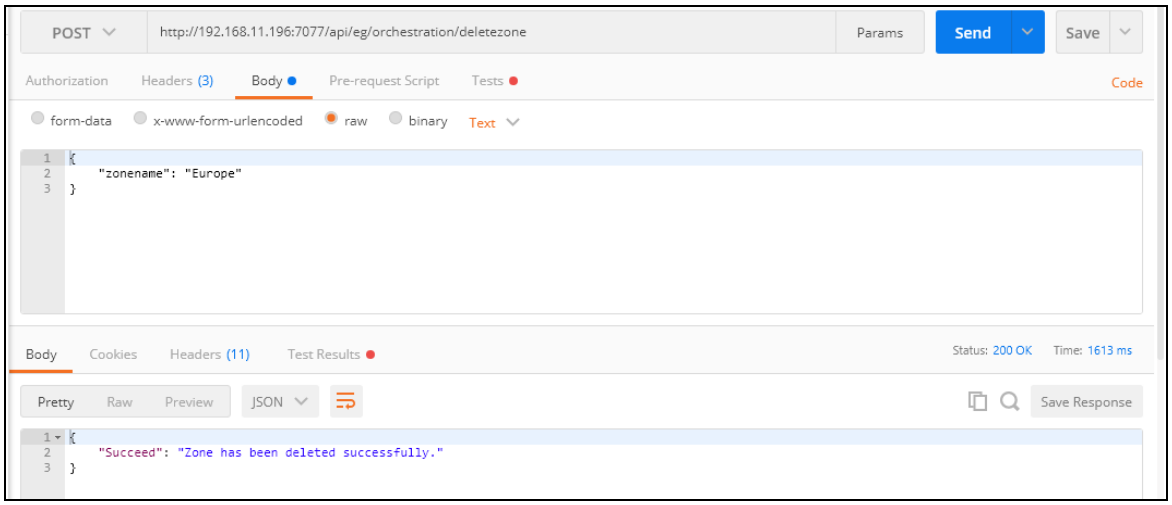

Figure 2.33: Deleting a zone using Postman REST Client

2.17.1 Deleting a Zone using cURL

To delete a zone through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/deletezone" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'zonename':'comma-separated list of zones'}"
```

Figure 2.34 shows an example of deleting a zone using cURL.

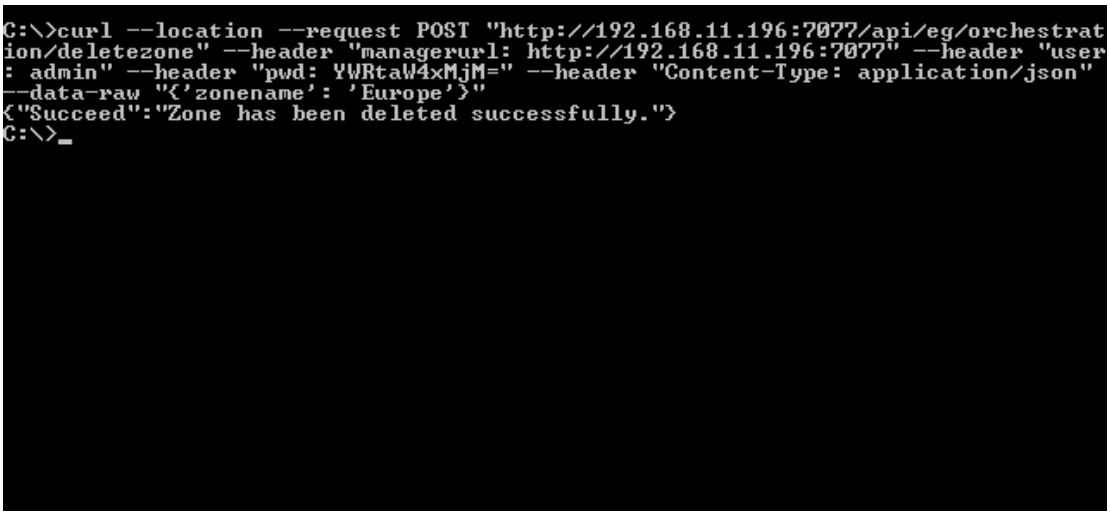

Figure 2.34: Deleting a zone using cURL

2.18 Disabling Tests

Use this API to disable one/more tests of a chosen component type.

URL: http://192.168.8.206:7077/api/eg/orchestration/disabletests

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"componenttype":"Microsoft SQL",
		"tests":"SQL Blocker Processes, SQL locks"
		}
		Example with both Default and Optional Key Values:
		{
		"componenttype":"Microsoft SQL",
		testtype:"configuration"

Parameters	Key values	Example

Body	**Default:**
	{
	"componenttype":"Component Type",
	"tests":"comma-separated list of tests"
	}
		"tests":"Drives"
		}

	Optional:
	{
	"testtype":"performance/ configuration",
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Test(s) is/are disabled for this component type."
		}

Failure Response

Type	Content

JSON	{
	"Error": "One or more tests are not available for this component type."
	}

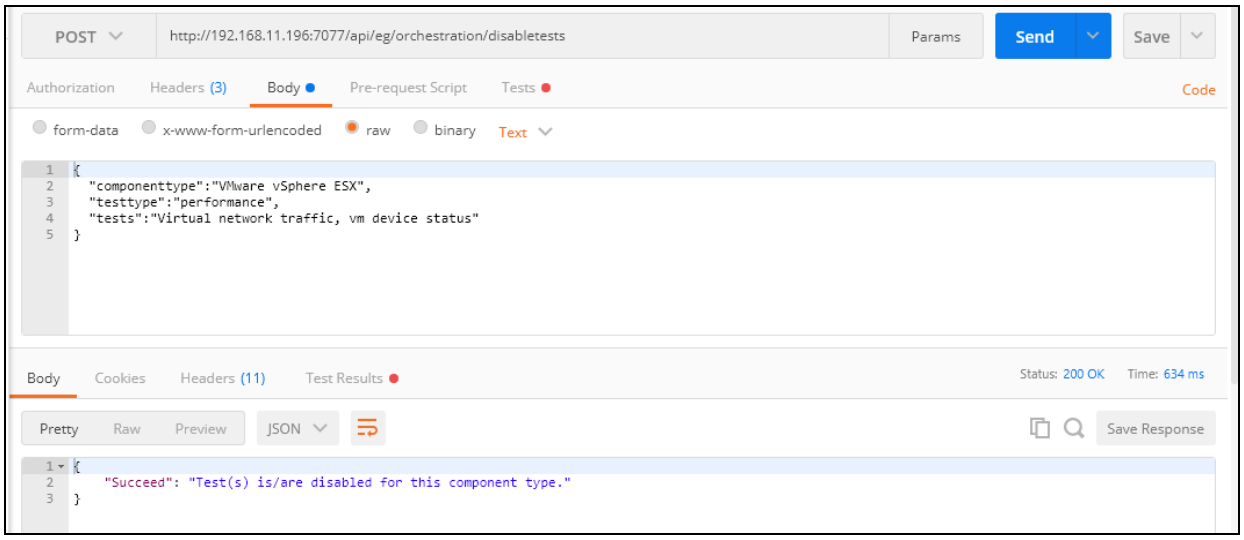

Figure 2.35: Disabling one/more tests of a chosen component type using Postman REST Client

2.18.1 Disabling Tests using cURL

To disable one/more tests of a chosen component type through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/disabletests" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'componenttype':'Component Type', 'tests':'comma-
separated list of tests', 'testtype':'performance/configuration'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.36 shows an example of disabling one/more tests of a chosen component type using cURL.

Figure 2.36: Disabling one/more tests of a chosen component type using cURL

2.19 Enabling Tests

Use this API to enable one/more tests for a chosen component type.

URL: http://192.168.8.206:7077/api/eg/orchestration/enabletests

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with both Default and Optional Key Values:
		{
		"componenttype":"Microsoft SQL",
		testtype:"configuration"
		"tests":"Operating System,Drives"
		}
		Example with Default key values:
		{
		"componenttype":"Microsoft SQL",

Parameters	Key values	Example

Body	**Default:**
	{
	"componenttype":"Component Type",
	"tests":"comma-separated list of tests"
	}
		"tests":"SQL Blocker Processes, SQL locks"
		}

	Optional:
	{
	"testtype":"performance/configuration",
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Test(s) is/are enabled for this component type."
		}

Failure Response

Type	Content

JSON	{
	"Error": "One or more tests are not available for this component type."
	}

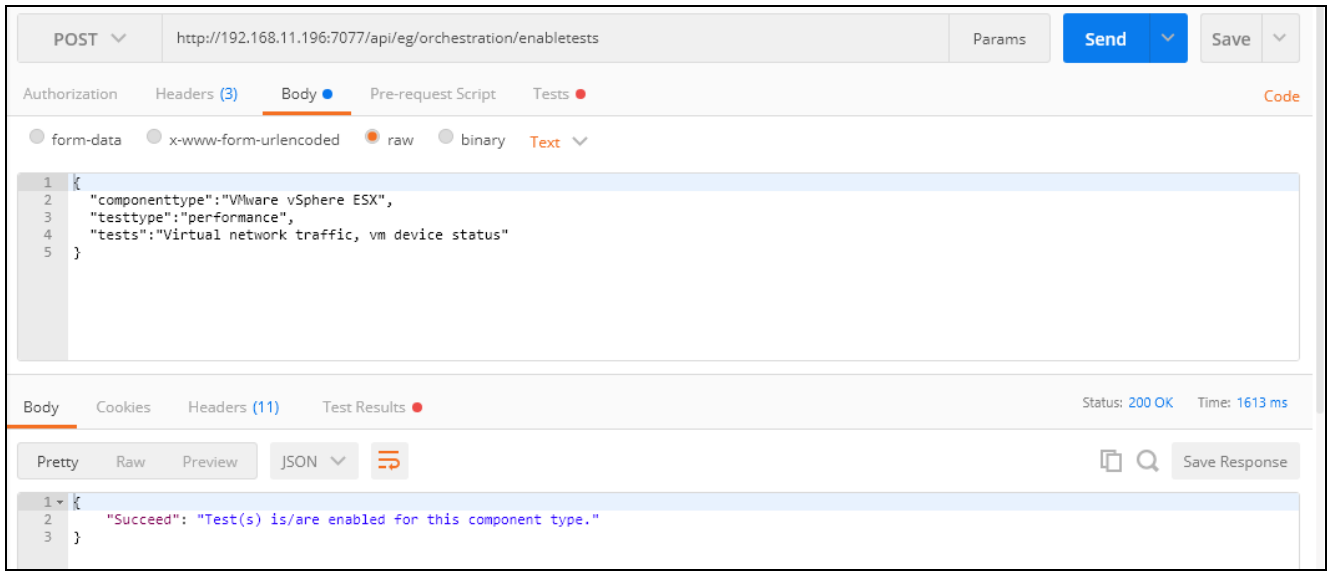

Figure 2.37: Enabling one/more tests for a chosen component type using Postman REST Client

2.19.1 Enabling Tests using cURL

To enable one/more tests for a chosen component type through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/enabletests" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'componenttype':'Component Type', 'tests':'comma-
separated list of tests', 'testtype':'performance/configuration'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.38 shows an example of enabling one/more tests of a chosen component type using cURL.

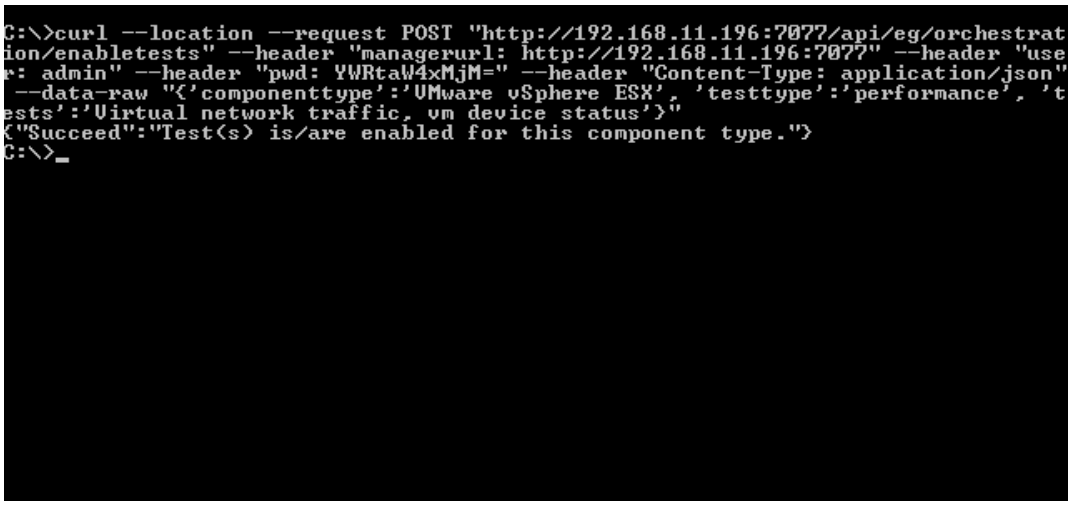

Figure 2.38: Enabling one/more tests for a chosen component type using cURL

2.20 Exclude Components for Test

Use this API to exclude one/more components for a test.

URL: http://192.168.8.206:7077/api/eg/orchestration/excludecomponentsfortest

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"componenttype":"2X Client Gateway",
		"componentname":"client_1:80,client_2:80",
		"testname":"2X Gateway Status"
		}
		Example with both Default and Optional Key Values:
		{
		"componenttype":"2X Client Gateway",

Parameters	Key values	Example

Body	**Default:**
	{
	"componenttype":"component type",
	"componentname":"comma-separated list of component names:Port number",
	"testname":"Test name"
	}
	Optional:
	{
	"testtype":"performance/ configuration",
	}
		"componentname":"client_1:80,client_2:80",
		"testtype":"configuration",
		"testname":"Drives"
		}

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component(s) is/are excluded successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "One or more component names do not exist."
	}

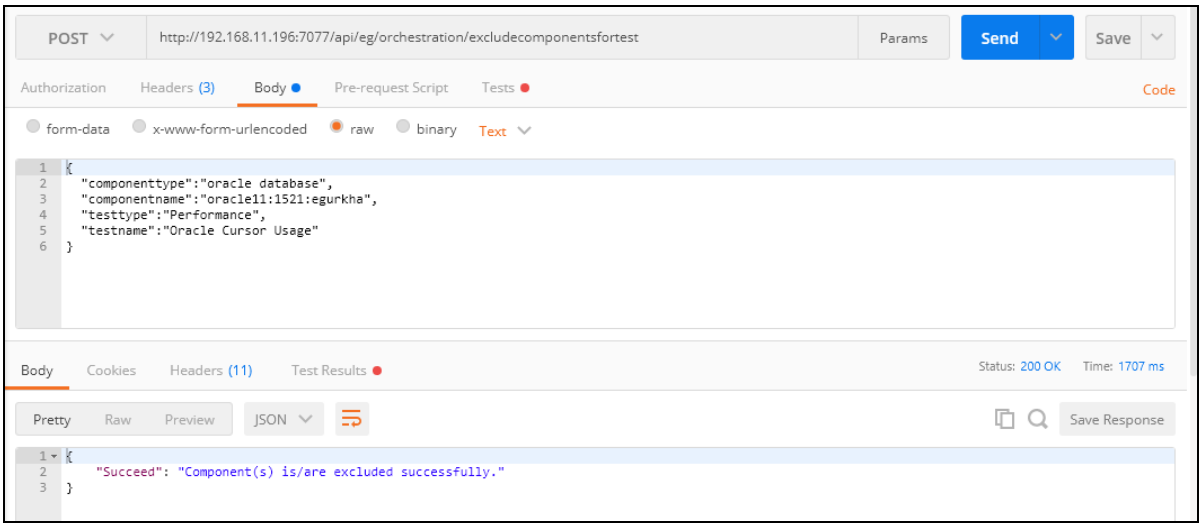

Figure 2.39: Example for excluding Components for Test using Postman REST Client

2.20.1 Excluding Components for Test using cURL

To exclude one/more components for a test through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/excludecomponentsfortest" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'componenttype':'component type',
'componentname':'comma-separated list of component names:Port number', 'testname':'Test
name', 'testtype':'performance/configuration'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.40 shows an example of excluding one/more components for a test using cURL.

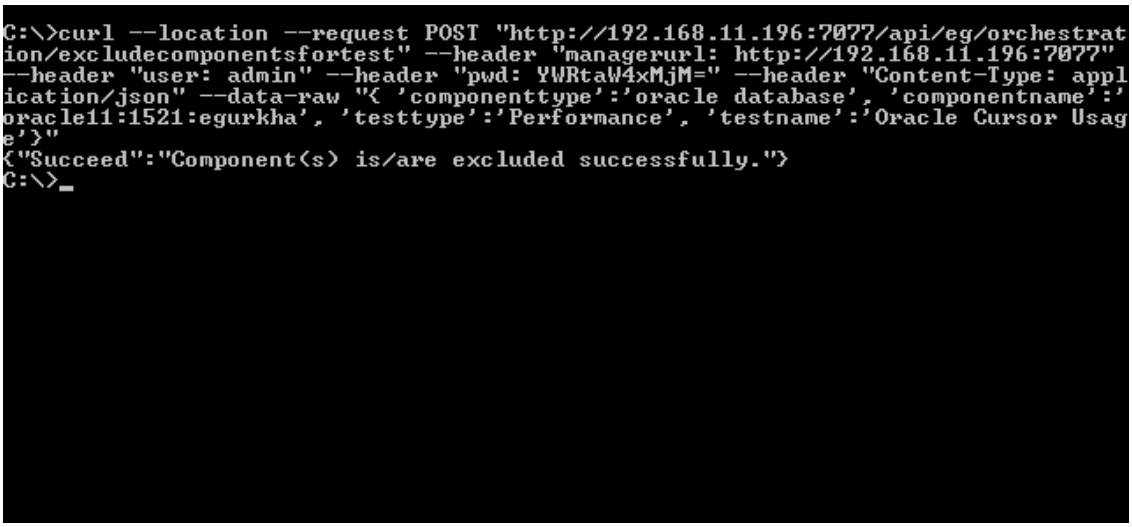

Figure 2.40: Excluding one/more components for a test using cURL

2.21 Exclude Tests for Component

Use this API to exclude one/more tests for a chosen component.

URL: http://192.168.8.206:7077/api/eg/orchestration/excludetestsforcomponent

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with both Default and Optional Key Values:
		{
		"componenttype":"Active Directory",
		"componentname":"actDir:389",
		"testtype":"configuration",
		"testname":"Operating System,Drives"
		}
		Example with Default key values:
		{

Parameters	Key values	Example

Body	**Default:**
	{
	"componenttype":"component type",
	"componentname":Component name:Port number",
	"testname":"comma-separated list of test names"
	}
		"componenttype":"Active Directory",
		"componentname":"actDir:389",
		"testname":"AD Replications,Application Events."
		}

	Optional:
	{
	"testtype":"performance/ configuration",
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Test(s) is/are excluded successfully."
		}

Failure Response

Type	Content
JSON	{

Type	Content
	"Error": "One or more tests for the component are already excluded." }

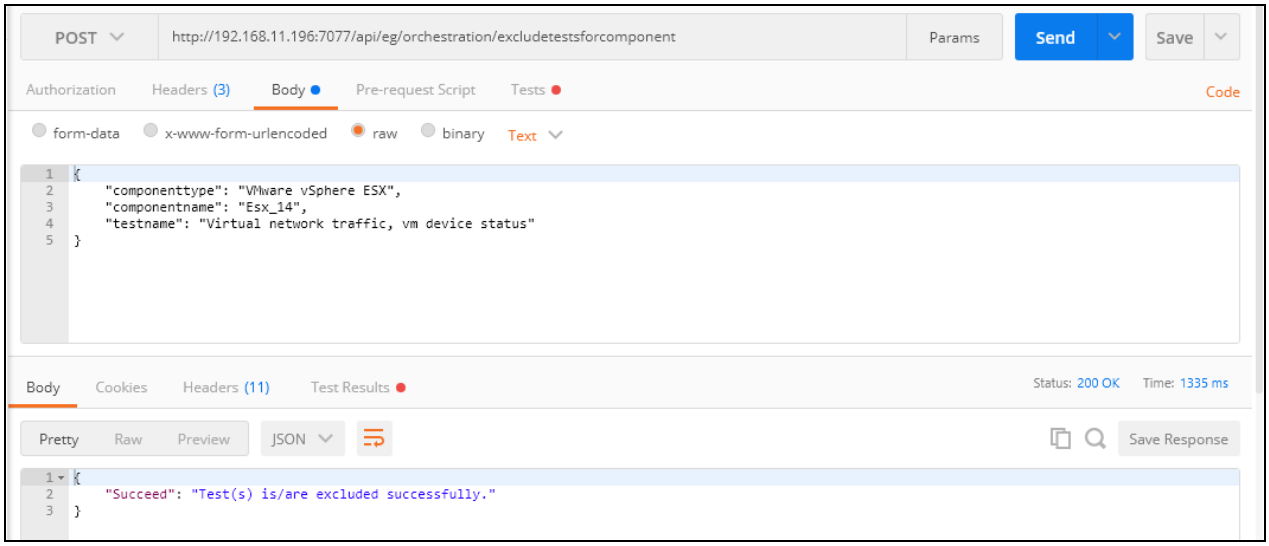

Figure 2.41: Excluding one/more tests for a Component using Postman REST Client

2.21.1 Excluding Tests for Component using cURL

To exclude one/more tests for a component through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/excludetestsforcomponent" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'componenttype':'component type',
'componentname':Component name:Port number', 'testname':'comma-separated list of test
names', 'testtype':'performance/ configuration'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.42 shows an example of excluding one/more tests for a component using cURL.

Figure 2.42: Excluding one/more tests for a Component using cURL

2.22 Include Components for Test

Use this API to include one/more components for a test.

URL: http://192.168.8.206:7077/api/eg/orchestration/includecomponentsfortest

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"componenttype":"2X Client Gateway",
		"componentname":"client_1:80,client_2:80",
		"testname":"2X Gateway Status"
		}
		Example with both Default and Optional Key Values:
		{
		"componenttype":"2X Client Gateway",

Parameters	Key values	Example

Body	**Default:**
	{
	"componenttype":"component type",
	"componentname":"comma-separated list of component names:Port number",
	"testname":"Test name"
	}
		"componentname":"client_1:80,client_2:80",
		"testtype":"configuration",
		"testname":"Drives"
		}

	Optional:
	{
	"testtype":"performance/ configuration",
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component(s) is/are included successfully."
		}

Failure Response

Type	Content
JSON	{

Type	Content
	"Error": "One or more component names do not exist." }

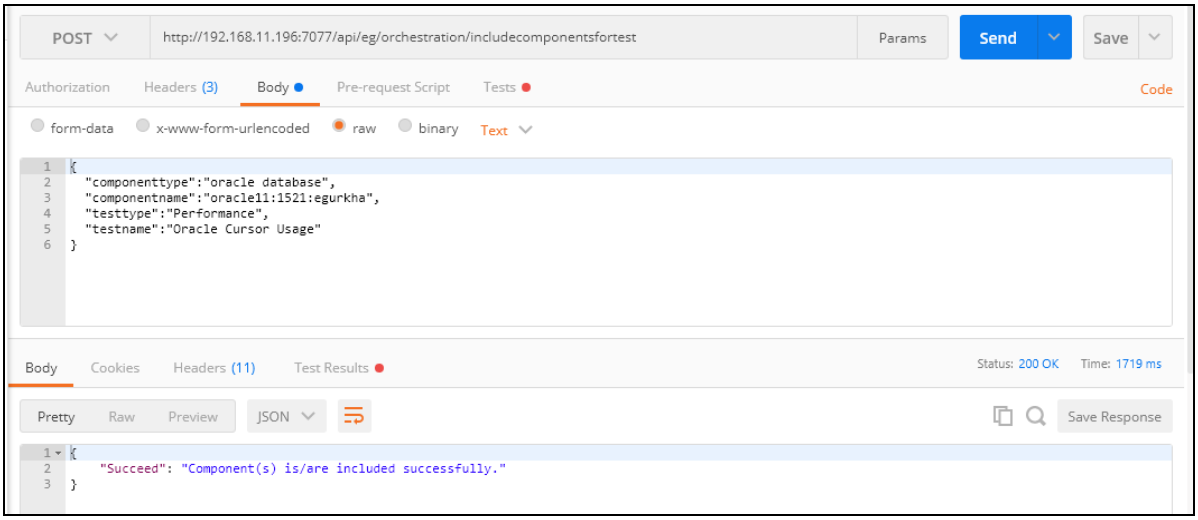

Figure 2.43: Example to include one/more components for a test using Postman REST Client

2.22.1 Include Components for Test using cURL

To include one/more components for a test through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/includecomponentsfortest" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'componenttype':'component type',
'componentname':'comma-separated list of component names:Port number', 'testname':'Test
name', 'testtype':'performance/configuration'}"
```

Note that the command specified above contains both the Default and Optional key values.

2.22 shows an example of including one/more components for a test using cURL.

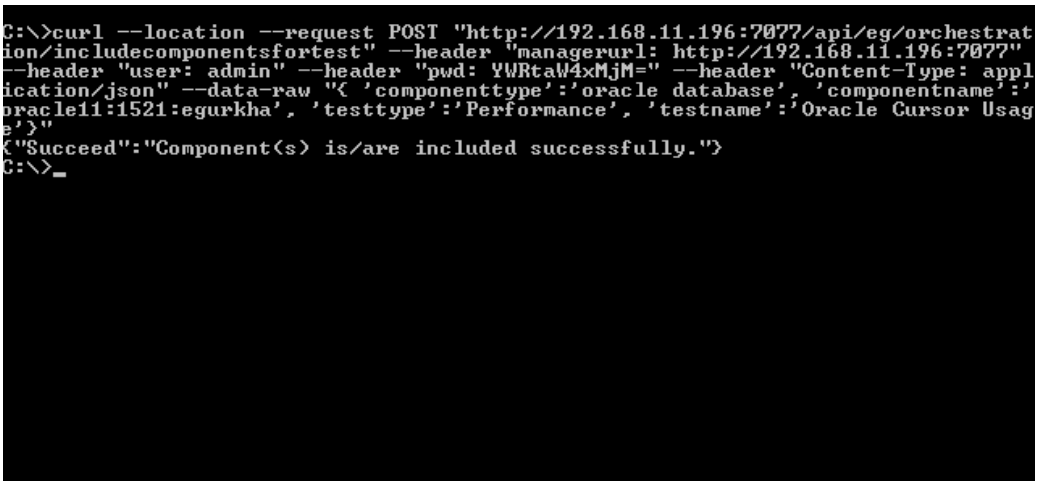

Figure 2.44: Including one/more components for a test using cURL

2.23 Include Tests for Component

Use this API to include one/more tests for a component.

URL: http://192.168.8.206:7077/api/eg/orchestration/includetestsforcomponent

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"componenttype":"Active Directory",
		"componentname":"actDir:389",
		"testname":"AD Replications,Application Events."
		}
		Example with both Default and Optional Key Values:
		{
		"componenttype":"Active Directory",

Parameters	Key values	Example

Body	**Default:**
	{
	"componenttype":"component type",
	"componentname":Component name:Port number",
	"testname":"comma-separated list of test names"
	}
		"componentname":"actDir:389",
		"testtype":"configuration",
		"testname":"Operating System,Drives"
		}

	Optional:
	{
	"testtype":"performance/ configuration",
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Test(s) is/are included successfully."
		}

Failure Response

Type	Content
JSON	{

Type	Content
	"Error": "One or more tests are not available for this component type." }

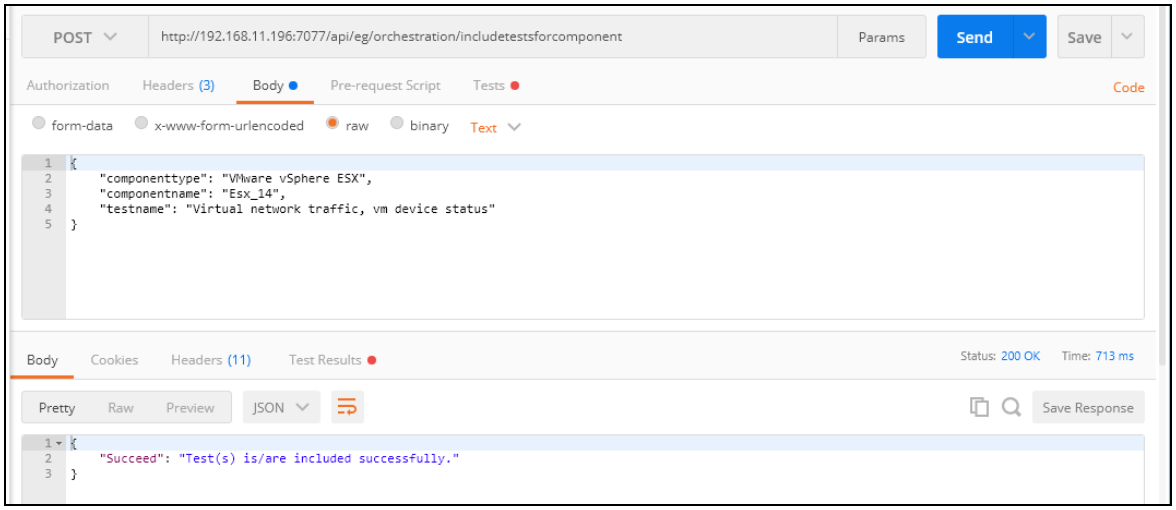

Figure 2.45: Example to include one/more tests for a component using Postman REST Client

2.23.1 Including Tests for Component using cURL

To include one/more tests for a component through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/includetestsforcomponent" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'componenttype':'component type',
'componentname':Component name:Port number', 'testname':'comma-separated list of test
names', 'testtype':'performance/ configuration'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.46 shows an example of including one/more tests for a component using cURL.

Figure 2.46: Including one/more tests for a component using cURL

2.24 Managing Components

Administrators can use this API to manage components in eG Enterprise.

URL: http://192.168.8.206:7077/api/eg/orchestration/managecomponent

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"componenttype":"Microsoft SQL",
		"componentname":"MSSQL",
		"port":"1433"
		}
		Example with both Default and Optional Key Values:
		{

Parameters	Key values	Example

Body	**Default:**
	{
	"componentname":"The nick name of the component",
	"componenttype":"Component type",
	"port":"Port"
	}
		"componenttype":"Oracle Database Server",
		"componentname":"Oradb",
		"port":"1521",
		"sid":"egora"
		}

	Optional:
	{
	"sid":"SID"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component has been managed successfully."
		}

Failure Response

Type	Content
JSON	{ "Error": "The selected component does not exist."

Type	Content
	}

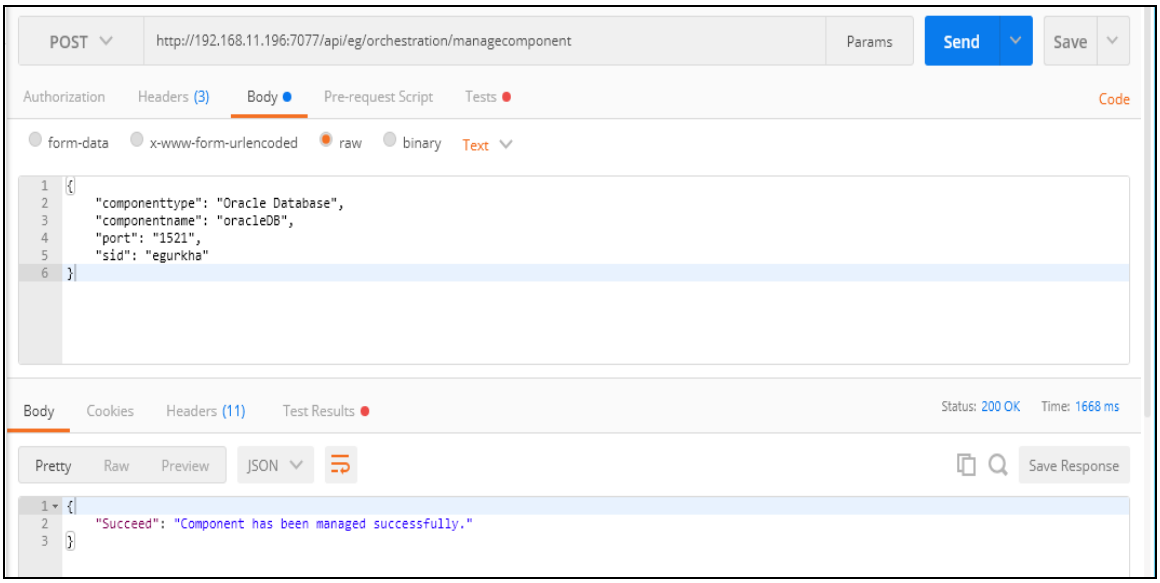

Figure 2.47: Example to manage components using Postman REST Client

2.24.1 Managing Components using cURL

To manage a component through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/managecomponent" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'componentname':'The nick name of the
component', 'componenttype':'Component type', 'port':'Port', 'sid':'SID'}"
```

Note that the command specified above contains both the Default and Optional key values.

Figure 2.48 shows an example of managing a component using cURL.

Figure 2.48: Managing a component using cURL

2.25 Modifying a Component

Use this API to modify the details of a component.

URL: http://192.168.8.206:7077/api/eg/orchestration/modifycomponent

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with both Default and Optional Key Values:
		{
		"hostip":"192.168.11.175",
		"componenttype":"Microsoft SQL",
		"componentname":"MSSQL",
		"oldcomponentname":"MSSQL",
		"newcomponentname":"MSSQL_DB",
		"newport":"1433"
		}

<table>
<tr><th>Parameters</th><th>Key values</th><th>Example</th></tr>
<tr><td rowspan="2">Body</td><td>Default:
{
"hostip":"Host IP",
"componentname":"The nick name of the component",
"componenttype":"Component type"
}</td><td rowspan="2">Example with Default key values:
{
"componenttype":"Microsoft SQL",
"componentname":"MSSQL",
"port":"1433"
}</td></tr>
<tr><td>Optional:
{
"port":"Port",
"oldhostip":"Old host IP",
"newhostip":"New host IP",
"oldcomponentname":"Old nick name",
"newcomponentname":"New nick name",
"oldport":"Old port",
"newport":"New port"
}</td></tr>
</table>

Note:

If an Oracle Database server is added with multiple SIDs, then the eG Enterprise system will monitor each SID as a different Oracle Database server. Therefore, while removing an Oracle Database server that supports multiple SIDs, you cannot issue a single command to remove all the SIDs at one shot. Instead, this command should be invoked separately for each SID.

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component has been modified successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "The old component name or port does not exist."
	}

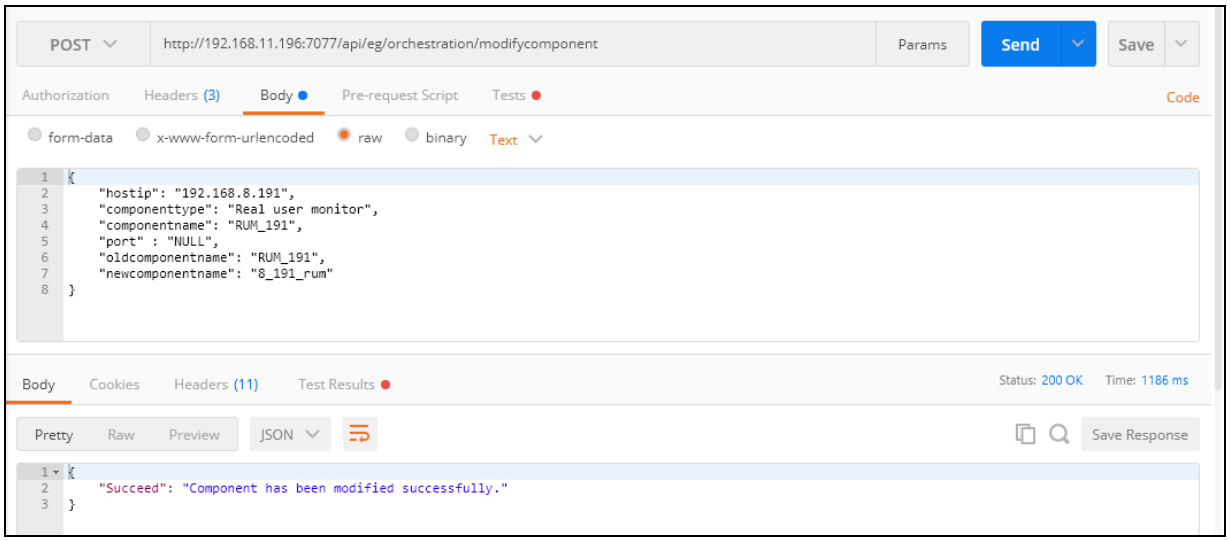

Figure 2.49: Example to modify the details of a component using Postman REST Client

2.25.1 Modifying a Component using cURL

To modify the details of a component through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/modifycomponent" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'componenttype': 'ComponentType',
'hostip': 'IP address of the component', 'componentname': 'nick name of the component',
'port': 'port at which the component listens', 'oldhostip':'Old host IP',
'newhostip':'New host IP', 'oldcomponentname':'Old nick name', 'newcomponentname':'New
nick name', 'oldport':'Old port', 'newport':'New port'}"
```

Note that the command specified above contains both the Default and Optional key values. Figure 2.50 shows an example of modifying a component using cURL.

Figure 2.50: Modifying a component using cURL

2.26 Modifying a Group

Administrators can use this API to modify the details of an existing group.

URL: http://192.168.8.206:7077/api/eg/orchestration/modifygroup

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the	Example with Default key values: {

<table>
<tr><th>Parameters</th><th>Key values</th><th>Example</th></tr>
<tr><td></td><td>eG console:Port>
user: eG username or domain/eG username
pwd: Base64 encoded password</td><td rowspan="3">"groupname":"MGRGroup",
"disassociateelements":"IIS Web:iis1:80"
}
Example with both Default and Optional Key Values:
{
"groupname":"MGRGroup",
"disassociateelements":"IIS Web:iis1:80",
"associateelements":"Active Directory:ad:1234"
}</td></tr>
<tr><td rowspan="2">Body</td><td>Default:
{
"groupname":"Group name",
"disassociateelements":"Elements"
}</td></tr>
<tr><td>Optional:
{
"associateelements":"Elements"
}</td></tr>
</table>

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Group has been modified successfully."
		}

Failure Response

Type	Content
JSON	{

Type	Content
	"Error": "One or more invalid elements to associate. Invalid elements" }

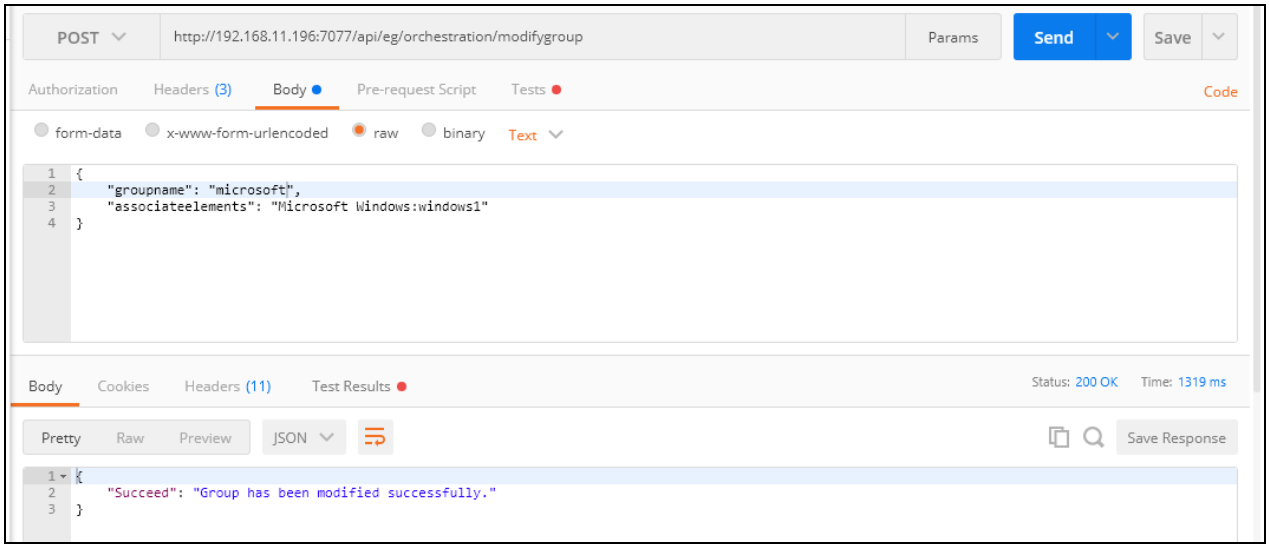

Figure 2.51: Example to modify the details of a group using Postman REST Client

2.26.1 Modifying a Group using cURL

To the details of an existing group through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/modifygroup" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'groupname':'Group name',
'disassociateelements':'Elements', 'associateelements':'Elements'}"
```

Note that the command specified above contains both the Default and Optional key values. 2.26 shows an example of the details of an existing group using cURL.

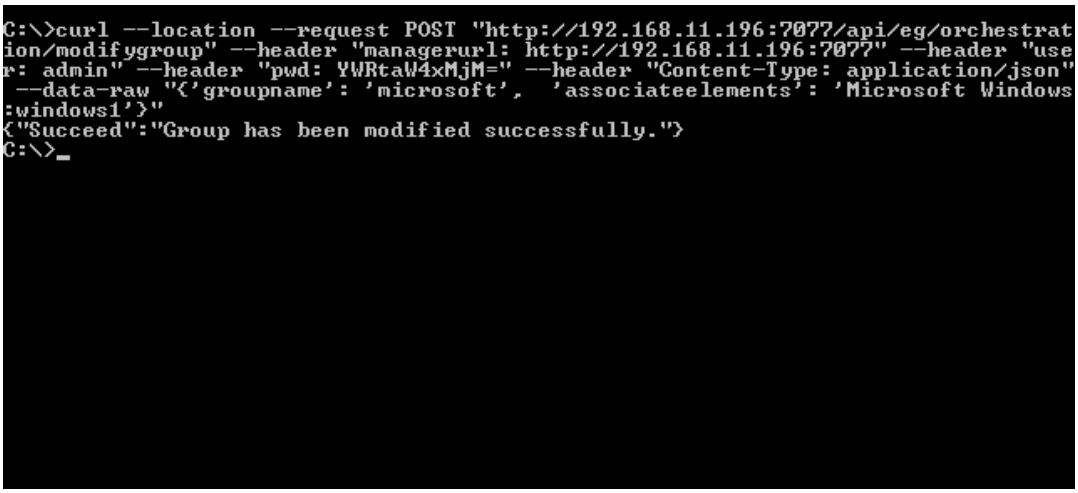

Figure 2.52: Modifying the details of an existing group using cURL

2.27 Modifying a Maintenance Policy

Use this API to modify the details of an existing maintenance policy.

URL: http://192.168.8.206:7077/api/eg/orchestration/modifymaintenancepolicy

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with both Default and Optional key values:
		{
		"policyname":"QMP1",
		"addtimefrequency":"Daily=13:30-14:30",
		"rmtimefrequency":"Daily=10:15-11:15"
		}

Body	**Default:**
	{
	"policyname":"Policy name"
	}	

	Optional:	

Parameters	Key values	Example

	{
	"addtimefrequency":"[Daily]/[First day of month]/
	[Last day of month]/
	[Sunday/Monday/Tuesday/Wednesday/Thursday/Friday/Saturday]/
	[MM/DD/YYYY-MM/DD/YYYY]=HH:MM-HH:MM",
	"rmtimefrequency":"[Daily]/[First day of month]/
	[Last day of month]/
	[Sunday/Monday/Tuesday/Wednesday/Thursday/Friday/Saturday]/
	[MM/DD/YYYY-MM/DD/YYYY]=HH:mm-HH:MM"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Maintenance policy modified successfully."
		}

Failure Response

Type	Content
JSON	{ "Error": "One or more time frequencies you are trying to remove do not exist."

Type	Content
	}

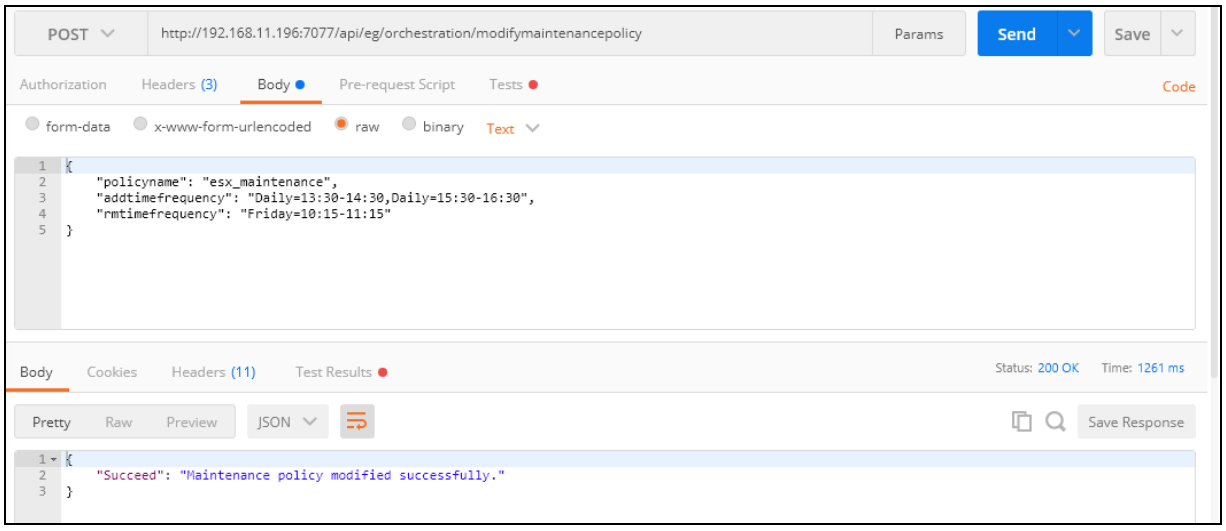

Figure 2.53: Example to modify the details of an existing maintenance policy using Postman REST Client

2.27.1 Modifying a Maintenance Policy using cURL

To modify the details of an existing maintenance policy through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/modifymaintenancepolicy" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'policyname':'Policy name',
'addtimefrequency':'[Daily]/[First day of month]/[Last day of month]/
[Sunday/Monday/Tuesday/Wednesday/Thursday/Friday/Saturday]/[MM/DD/YYYY-MM/DD/YYYY]=HH:MM-
HH:MM', 'rmtimefrequency':'[Daily]/[First day of month]/[Last day of month]/
[Sunday/Monday/Tuesday/Wednesday/Thursday/Friday/Saturday]/[MM/DD/YYYY-MM/DD/YYYY]=HH:mm-
HH:MM'}"
```

Note that the command specified above contains both the Default and Optional key values. Figure 2.54 shows an example of modifying the details of an existing maintenance policy using cURL.

Figure 2.54: Modifying the details of an existing maintenance policy using cURL

2.28 Modifying a User

Administrators can use this API to modify a user existing in eG Enterprise.

URL: http://192.168.8.206:7077/api/eg/orchestration/modifyuser

Method: POST

Content-Type: application/json

Inputs to be Specified

Para-meters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	Example: { "userid":"john", "userrole":"AlarmViewer", "password":"******", "expirydate":"12/12/2024", "alarmsbymail":"saran1@eginnovations.com, 9884011111",

Para-meters	Key values	Example

Body	**Default:**
	{
	"userid":"User ID"
	}
		"cc":"saranya@eginnovations.com, 9840391695",
		"bcc":"sample@eginnovations.com"
		}

	Optional:
	{
	"userrole":"User role",
	"password":"Password",
	"expirydate":"MM/dd/yyyy",
	"alarmsbymail":"Critical/Major/Minor/
	All",
	"to":"comma-separated list of Mail IDs/Mobile numbers",
	"cc":"comma-separated list of Mail IDs/Mobile numbers",
	"bcc":"comma-separated list of Mail IDs/Mobile numbers"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "User has been modified successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "User ID does not exist."
	}

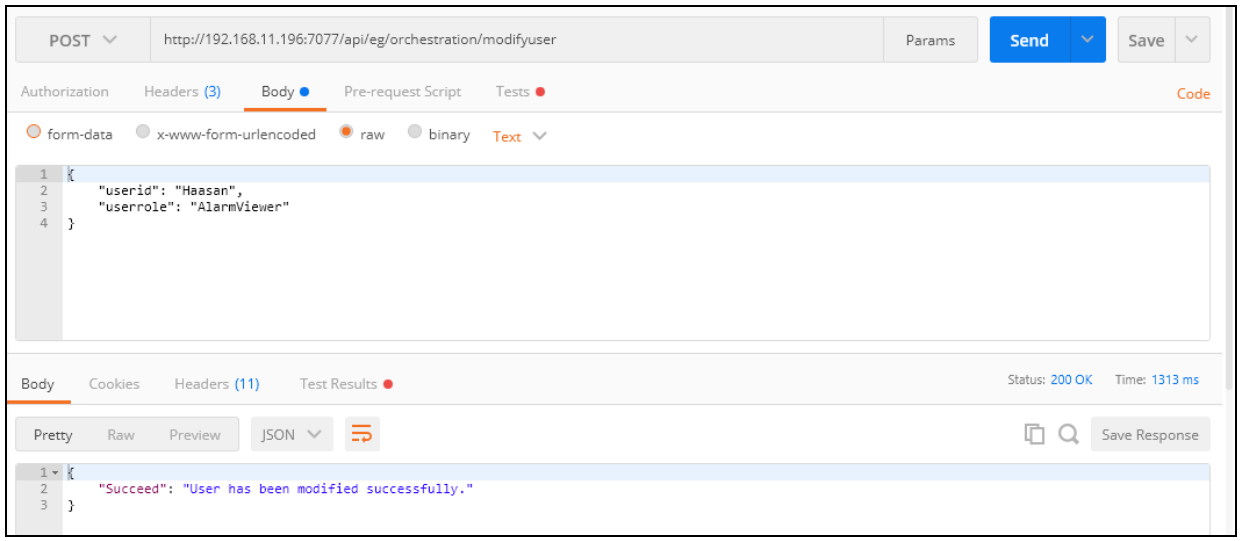

Figure 2.55: Example to modify a user using Postman REST Client

2.28.1 Modifying a User using cURL

To modify a user through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/modifyuser" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'userid':'User ID', 'userrole':'User role',
'password':'Password', 'expirydate':'MM/dd/yyyy',
'alarmsbymail':'Critical/Major/Minor/All', 'to':'comma-separated list of Mail IDs/Mobile
numbers', 'cc':'comma-separated list of Mail IDs/Mobile numbers', 'bcc':'comma-separated
list of Mail IDs/Mobile numbers'}"
```

Note that the command specified above contains both the Default and Optional key values. Figure 2.56 shows an example of modifying a user using cURL.

Figure 2.56: Modifying a user using cURL

2.29 Modifying a Zone

Administrators can use this API to modify the details of a zone created in eG Enterprise.

URL: http://192.168.8.206:7077/api/eg/orchestration/modifyzone

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"zonename":"westzone",
		"displayimage":"Banking",
		"disassociateelements":"Microsoft SQL:MSSQL_DB:1433"
		}

Body	**Default:**
	{
	"zonename":"Zone name"
	}	

	Optional:	

Parameters	Key values	Example

	{
	"associateelements":"Elements",
	"disassociateelements":"Elements",
	"displayimage":"Display image",
	"autoassociate":"yes/no"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Zone has been modified successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "One or more invalid elements to associate. Invalid elements"
	}

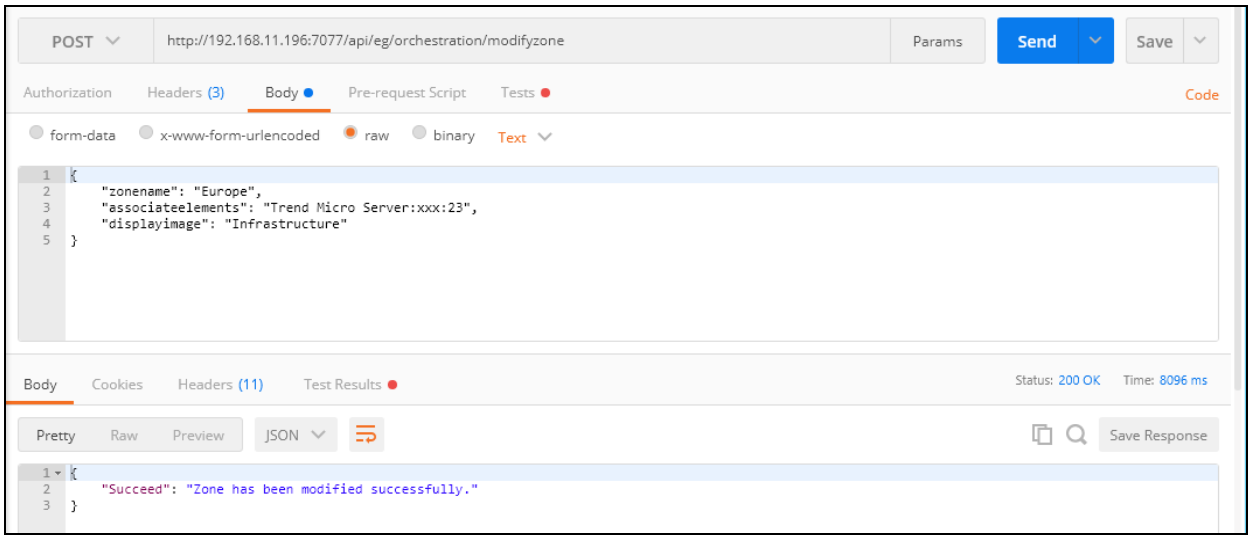

Figure 2.57: Example to modify a zone using Postman REST Client

2.29.1 Modifying a Zone using cURL

To modify the details of a zone through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/modifyzone" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'zonename':'Zone name',
'associateelements':'Elements', 'disassociateelements':'Elements',
'displayimage':'Display image', 'autoassociate':'yes/no'}"
```

Note that the command specified above contains both the Default and Optional key values. Figure 2.58 shows an example of modifying a zone using cURL.

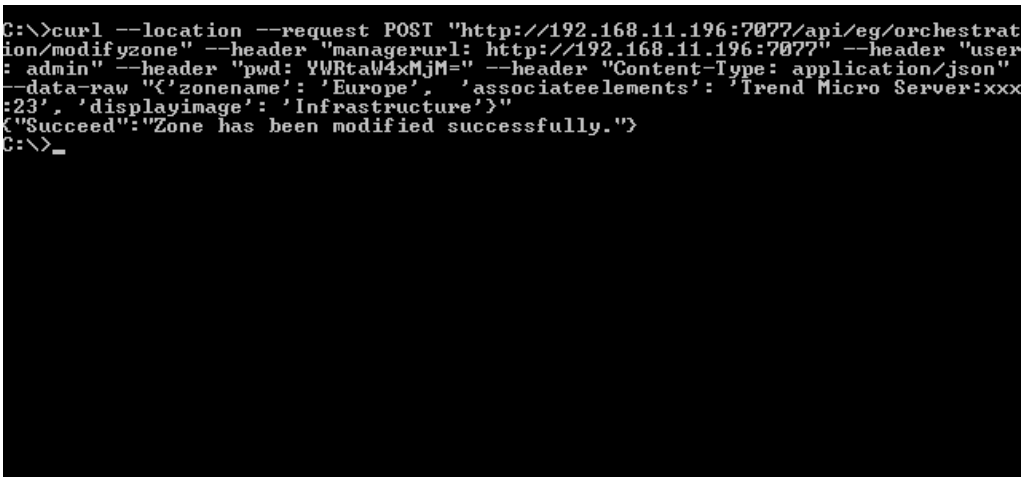

Figure 2.58: Modifying a zone using cURL

2.30 Renaming a Group

Use this API to rename an existing group in eG Enterprise.

URL: http://192.168.8.206:7077/api/eg/orchestration/renamegroup

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"groupname":"westgroup",
		"newgroupname":"West_group"
		}

Body	**Default:** { "groupname":"Group name",	

Parameters	Key values	Example
	"newgroupname":"New group name" }	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Group has been renamed successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "The given group does not exist."
	}

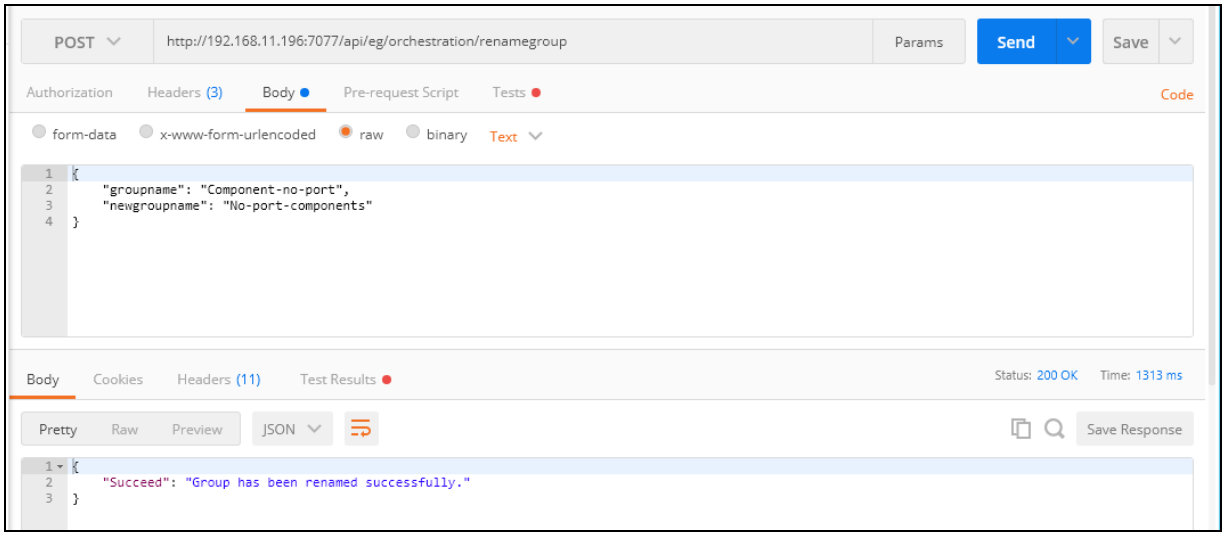

Figure 2.59: Example to rename a group using Postman REST Client

2.30.1 Renaming a Group using cURL

To rename an existing group through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/renamegroup" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'groupname':'Group name', 'newgroupname':'New group
name'}"
```

Figure 2.60 shows an example of renaming a group using cURL.

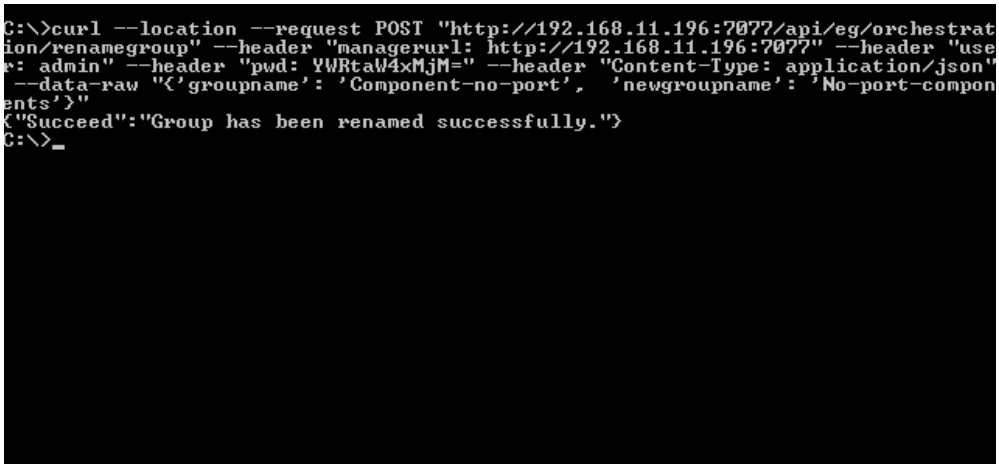

Figure 2.60: Renaming a group using cURL

2.31 Renaming a Zone

Administrators can use this API to rename an existing Zone in eG Enterprise.

URL: http://192.168.8.206:7077/api/eg/orchestration/renamezone

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the	{ "zonename":"Northzone",

Parameters	Key values	Example
	eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	"newzonename":"North_zone" }

Body	**Default:**
	{
	"zonename":"Zone name",
	"newzonename":"New zone name"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Zone has been renamed successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "The given zone does not exist."
	}

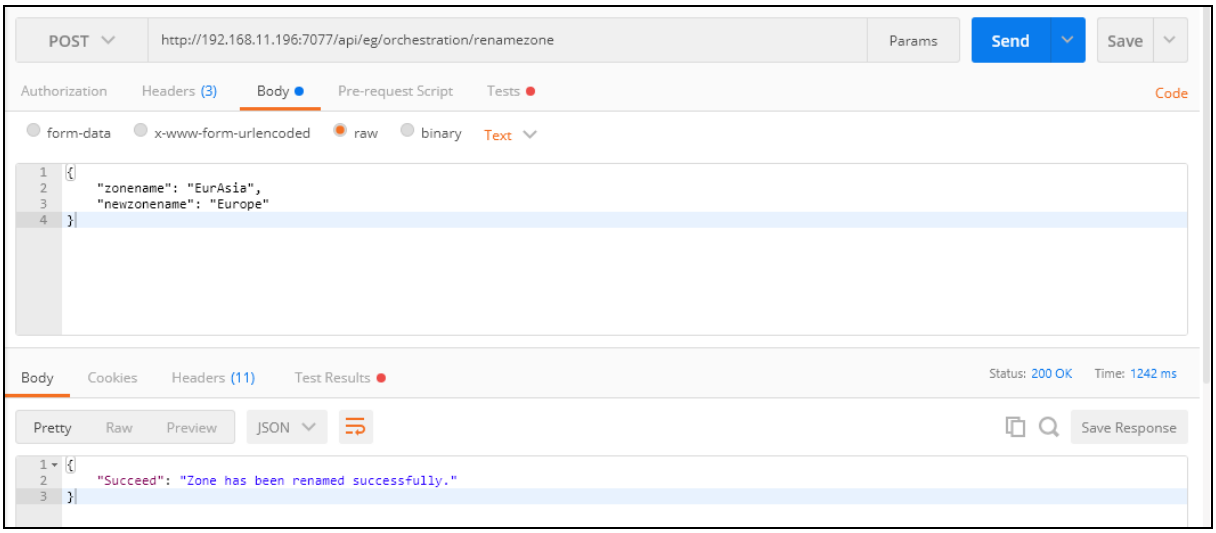

Figure 2.61: Example to rename an existing Zone using Postman REST Client

2.31.1 Renaming a Zone using cURL

To rename an existing zone through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager IP:Port>/api/eg/orchestration/renamezone" -H "managerurl:http://<eG Manager IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-Type: application/json" --data-raw "{'zonename':'Zone name', 'newzonename':'New zone name'}"
```

Figure 2.62 shows an example of renaming a zone using cURL.

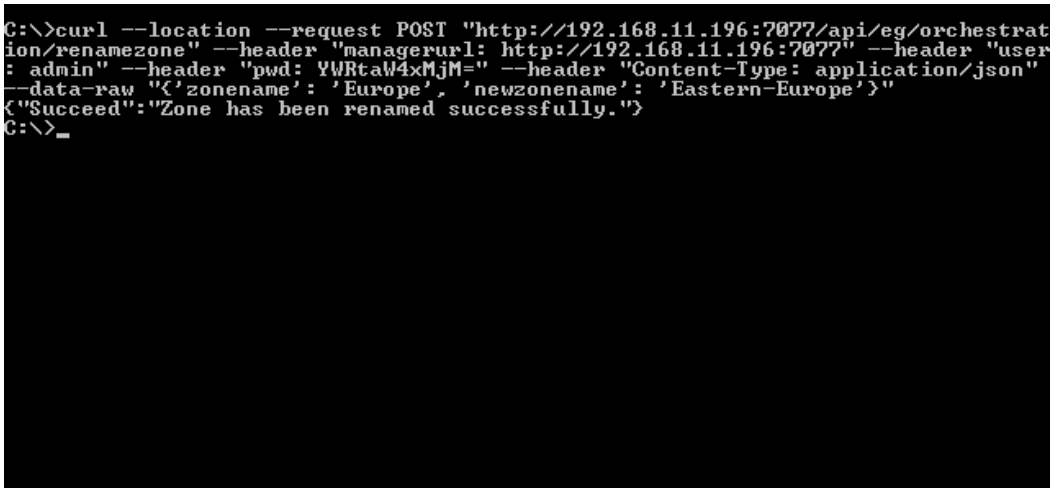

Figure 2.62: Renaming a zone using cURL

2.32 Displaying Components

Administrators can use this API to display all the components available in the target environment.

URL: http://192.168.8.206:7077/api/eg/orchestration/showcomponents

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"componenttype":"Citrix XenApp7.x"
		}

Body	**Default:**
	{
	"componenttype":"Component type"
	}	

Success Response

Type	Code	Content
JSON	200	[{ "componentType": "Real User Monitor", "ip": "192.168.8.191",

Type	Code	Content
		"componentName": "11_196_RUM", "port": "-", "externalAgent": "mobilecollector", "internalAgent": "mobilecollector" }, . . .]

Failure Response

Type	Content

JSON	{
	"Error": "Component type is not available."
	}

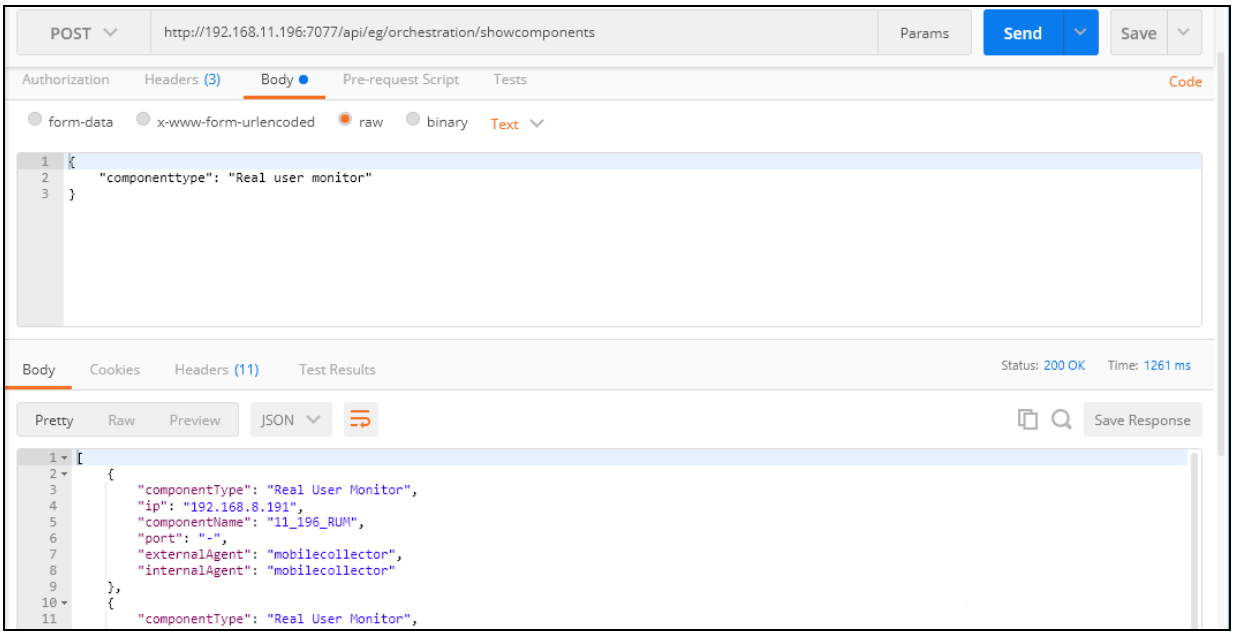

Figure 2.63: Displaying the components in the target environment using Postman REST Client

2.32.1 Displaying Components using cURL

To display the components in the target environment through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/showcomponents" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'componenttype': 'Component Type'}"
```

Figure 2.64 shows an example of displaying the components in the target environment using cURL.

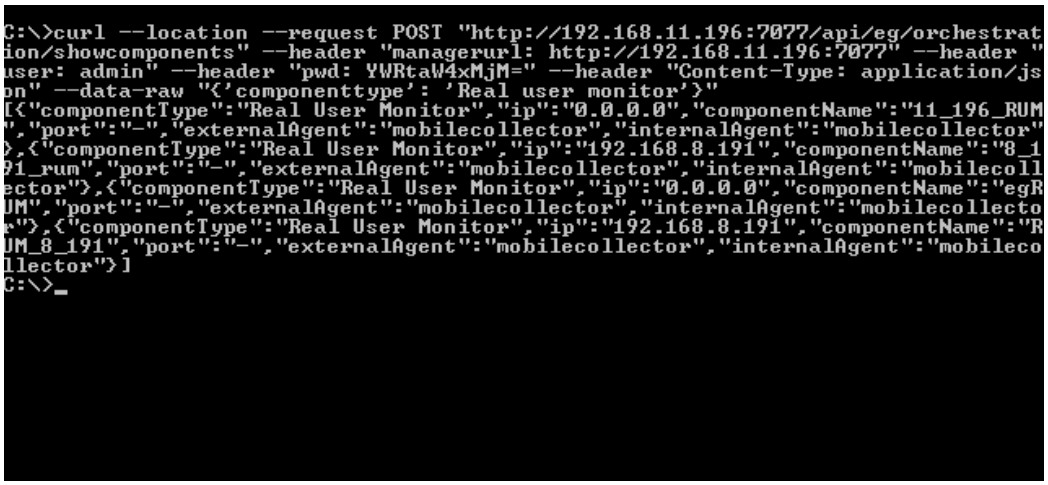

Figure 2.64: Displaying the components in the target environment using cURL

2.33 Displaying External Agents

Use this API to display all the external agents in the target environment.

URL: http://192.168.8.206:7077/api/eg/orchestration/showexternalagents

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Header	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	Not Applicable

Success Response

Type	Code	Content
JSON	200	[{ "agentName": "mobilecollector", "hostIp": "mobilecollector" }, . . .]

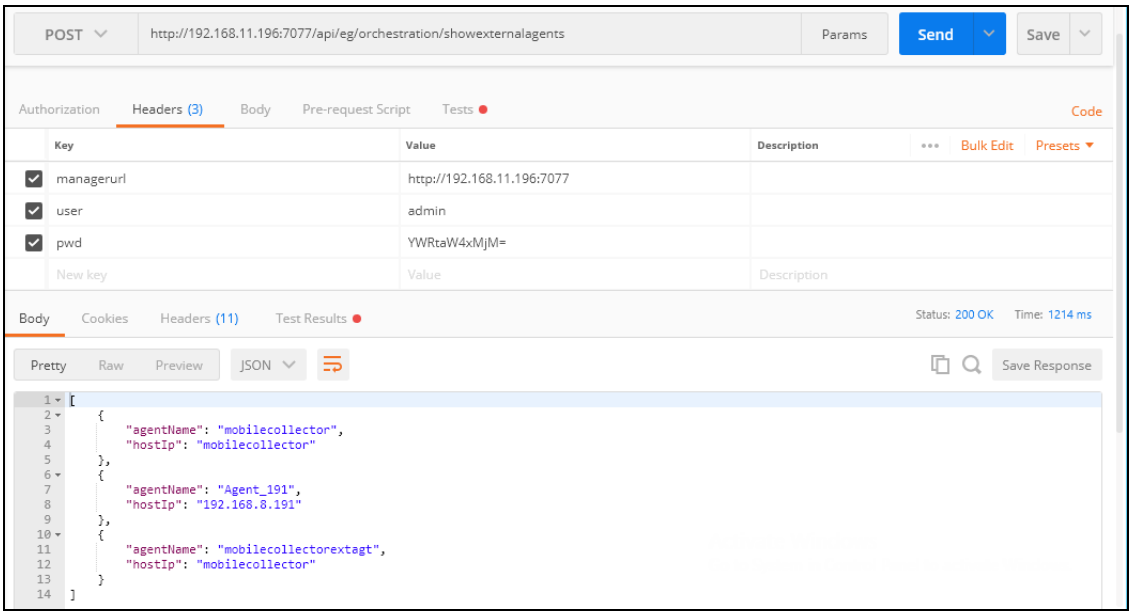

Figure 2.65: Displaying the External agents in the target environment using Postman REST Client

2.33.1 Displaying External Agents using cURL

To display all the external agents in the target environment through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/showexternalagents" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password"
```

Figure 2.66 shows an example of displaying all the external agents in the target environment using cURL.

Figure 2.66: Displaying all the external agents in the target environment using cURL

2.34 Displaying Remote Agents

Use this API to display all the remote agents in the target environment.

URL: http://192.168.8.206:7077/api/eg/orchestration/showremoteagents

Method: POST

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	Not Applicable

Success Response

Type	Code	Content
JSON	200	[{ "agentName": "mobilecollector", "hostIp": "mobilecollector" }, . . .]

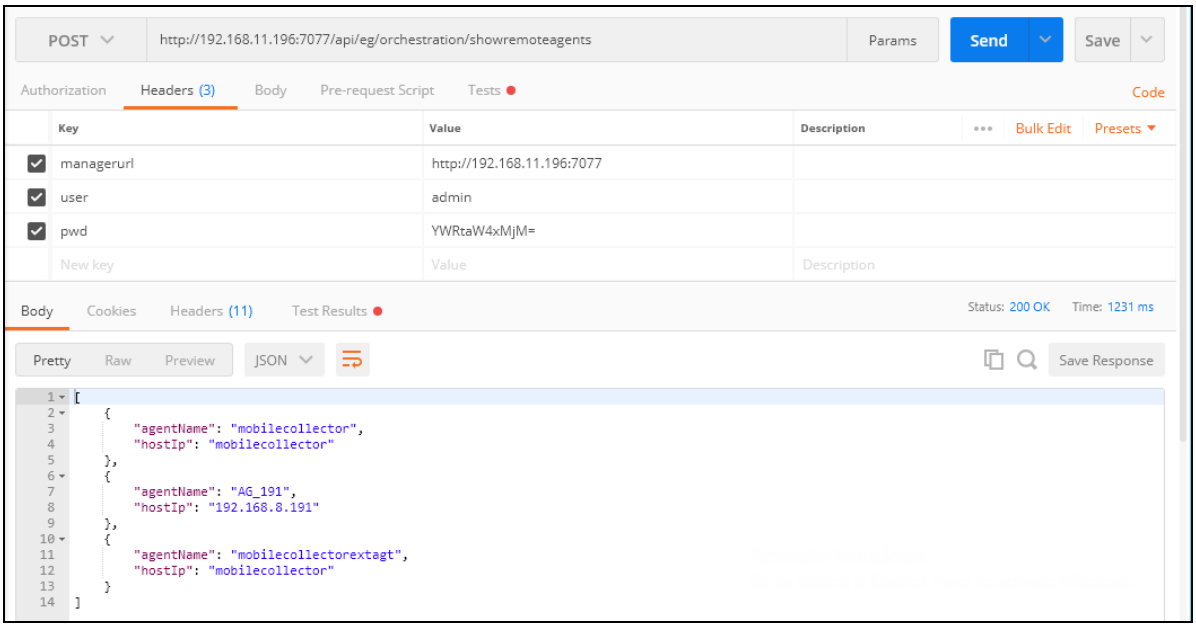

Figure 2.67: Displaying the Remote agents configured in the target environment using Postman REST Client

2.34.1 Displaying Remote Agents using cURL

To display all the remote agents in the target environment through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/showremoteagents" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password"
```

Figure 2.68 shows an example of displaying all the remote agents in the target environment using cURL.

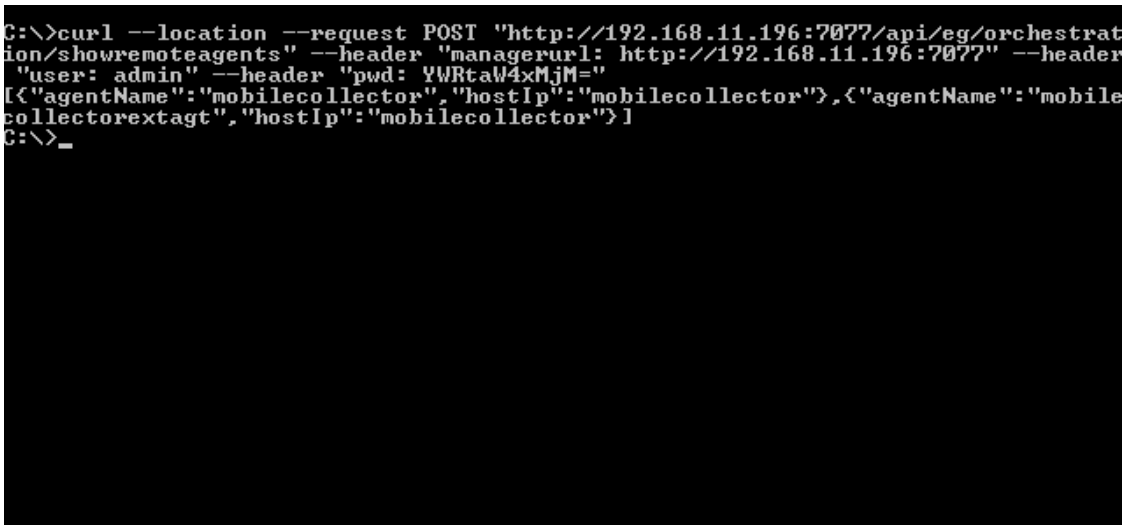

Figure 2.68: Displaying all the remote agents in the target environment using cURL

2.35 Displaying Maintenance Policies

Administrators can use this API to display all the maintenance policies configured in the target environment.

URL: http://192.168.8.206:7077/api/eg/orchestration/showmaintenancepolicynames

Method: POST

Inputs to be Specified

Parameters	Key values	Example
Header	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username	Not Applicable

Parameters	Key values	Example
	pwd: Base64 encoded password	

Success Response

Type	Code	Content

JSON	200	{
		"policyNames": [
		"esx_maintenance",
		"Manual_restart",
		"VDI_maintenance"
		]
		}

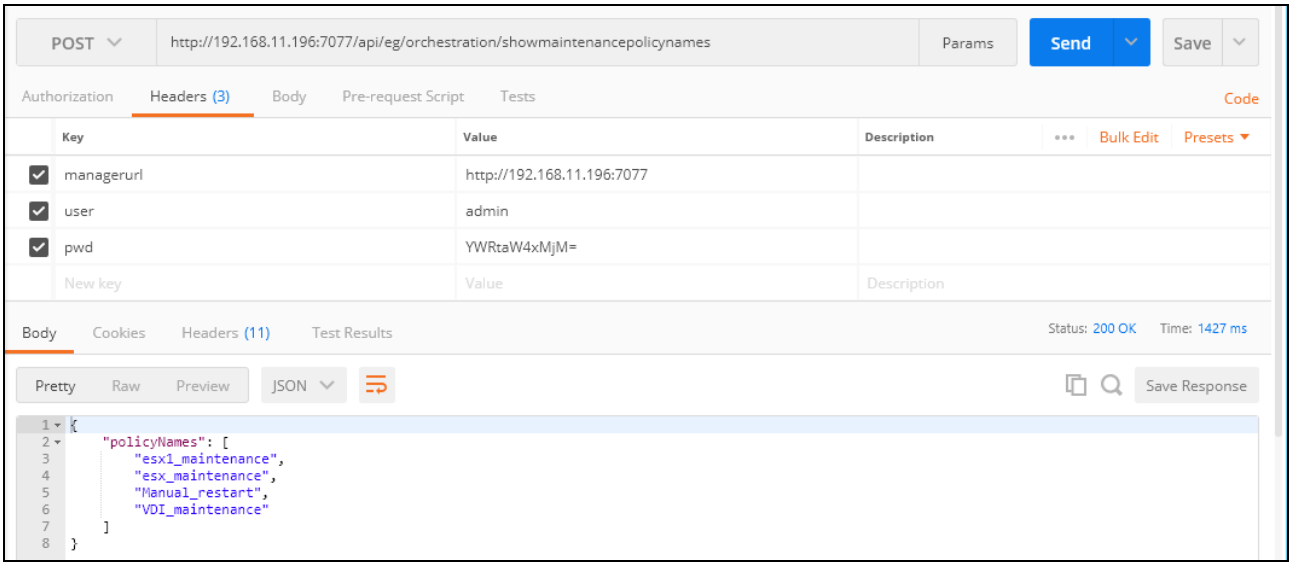

Figure 2.69: Displaying the Maintenance Policies configured in the target environment using Postman REST Client

2.35.1 Displaying Maintenance Policies using cURL

To display all the maintenance policies configured in the target environment through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/showmainteanncepolicynames" -H "managerurl:http://<eG
Manager IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded
password"
```

Figure 2.70 shows an example of displaying the maintenance policies configured in the target environment using cURL.

Figure 2.70: Displaying the maintenance policies in the target environment using cURL

2.36 Displaying Details of Maintenance Policies

Use this API to display all the details of the Maintenance Policies configured in the target environment.

URL: http://192.168.8.206:7077/api/eg/orchestration/showmaintenancepolicydetails

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"policyname":"QMP1, QMP2"
		}

Body	**Default:**
	{
	"policyname":"comma-separated list of Maintenance Policies"
	}	

Success Response

Type	Code	Content

JSON	200	[
		{
		"policyName": "VDI_maintenance",
		"policyStatus": "Deactive",
		"policySchedule": ["Last Day of Month=00:00-23:59"],
		"nextScheduleDate": "Sep 30, 2020 0:00-23:59",
		"associatedElements": {
		"component": ["Vdi_113"]
		}
		},

Type	Code	Content
		. . .]

Failure Response

Type	Content

JSON	{
	"Error": "One or more maintenance policy does/do not exist."
	}

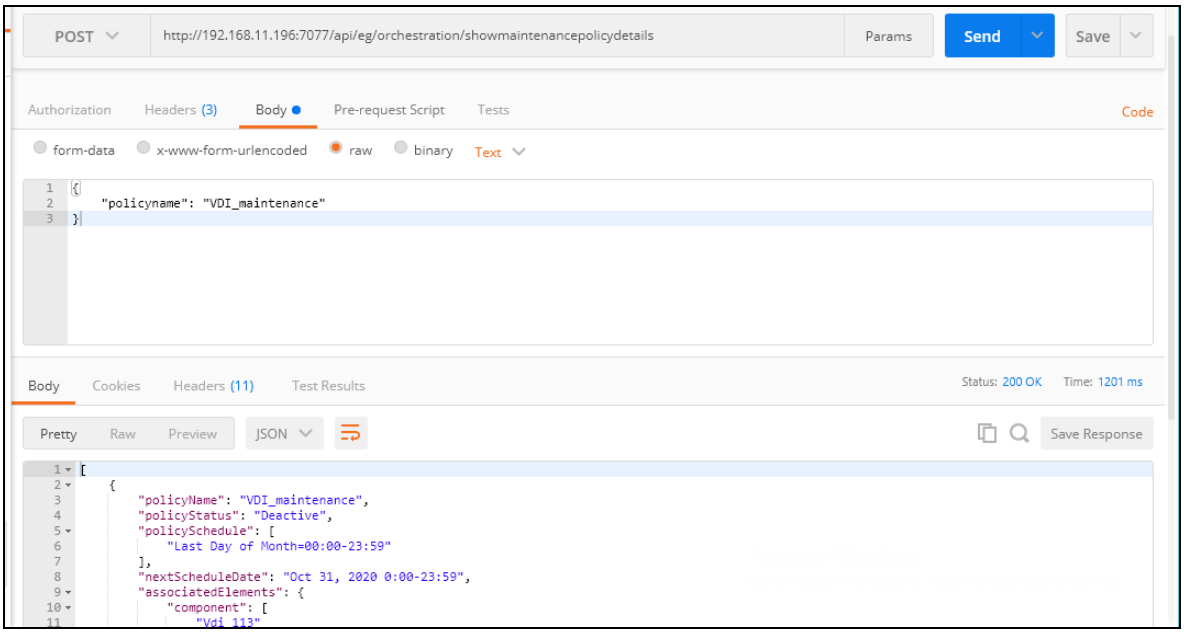

Figure 2.71: Displaying the details of the Maintenance Policies in the target environment using Postman REST Client

2.36.1 Displaying Details of Maintenance Policies using cURL

To display the details of the Maintenance Policies in the target environment through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/showmaintenancepolicydetails" -H "managerurl:http://<eG
Manager IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded
password" -H "Content-Type: application/json" --data-raw "{'policyname':'comma-separated
list of Maintenance Policies'}"
```

Figure 2.72 shows an example of displaying the details of the Maintenance Policies using cURL.

Figure 2.72: Displaying the details of the Maintenance Policies in the target environment using cURL

2.37 Displaying the Hosts Managed in the Target Environment

Use this API to display the hosts that are managed in the target environment.

URL: http://192.168.8.206:7077/api/eg/orchestration/showmanagedhosts

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>	Example with Default Key values:
		{
		"agentname":"192.168.11.136"
		}

Parameters	Key values	Example

	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default and Optional Key values:
		{
		"agentname":"ext12"
		"agenttype":" External agent"
		}

Body	**Default:**
	{
	"agentname":"Agent name"
	}	

	Optional:
	{
	"agenttype":"External agent/Remote agent"
	}	

Note that the agentname key value is case-sensitive.

Success Response

Type	Code	Content

JSON	200	{
		"managedHosts": [
		"Esx_14",
		"mobilecollector",
		"network_10",
		"Rds_196",
		"Vdi_113",
		"windows1"
		]
		}

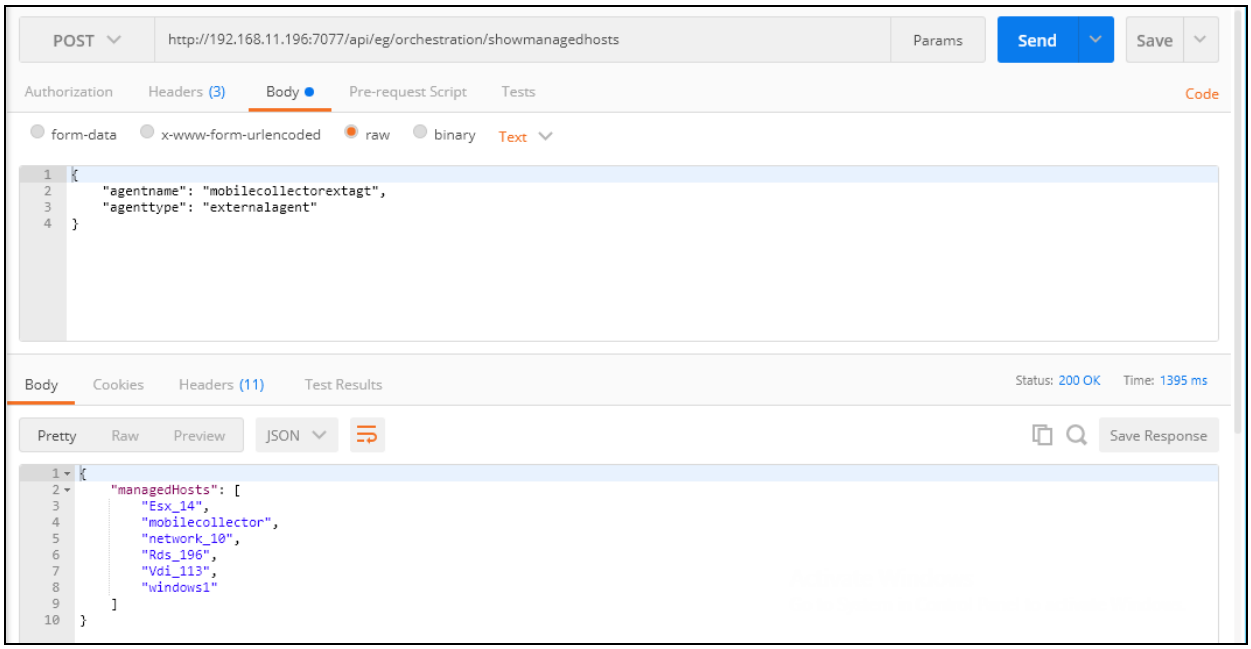

Figure 2.73: Displaying the hosts managed in the target environment using Postman REST Client

2.37.1 Displaying the Hosts Managed in the Target Environment using cURL

To display the hosts managed in the target environment through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/showmanagedhosts" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'agentname':'Agent name',
'agenttype':'External agent/Remote agent'}"
```

Note that the command specified above contains both the Default and Optional key values. Figure 2.74 shows an example of displaying the hosts managed in the target environment using cURL.

Figure 2.74: Displaying the hosts managed in the target environment using cURL

2.38 Displaying the Details of the Tests

Use this API to display the details of a test pertaining to a chosen Component Type.

URL: http://192.168.8.206:7077/api/eg/orchestration/showtestsdetails

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Optional Key values:
		{
		"componenttype": "Citrix NetScaler VPX/MPX",
		"componentname": "Netscaler176:NULL",
		"testtype": "Performance",
		"testname": "Application Flows"
		}

Body	Optional: { "componenttype":"Component type",	

Parameters	Key values	Example
	"componentname":"Component name", "testtype":"Performance / Configuration", "testname":"Test name" }	

Success Response

Type	Code	Content

JSON	200	[
		{
		"componentType": "Citrix NetScaler VPX/MPX",
		"componentName": "Netscaler176:NULL",
		"testType": {
		"performance": [
		{
		"testName": "Application Flows",
		"details": {
		"TESTPERIOD": "5 mins",
		"HOST": "192.168.10.176",
		"NETSCALER USERNAME": "$unconfigured",
		"NETSCALER PASSWORD": "$unconfigured",
		"SSL": "No",
		"AGENTLESS": "y",
		"OS": "win7",
		"SSHPORT": "22"
		}

Type	Code	Content
		}], "configuration": [] } }]

Failure Response

Type	Content

JSON	{
	"Error": "The test is not available for this component type."
	}

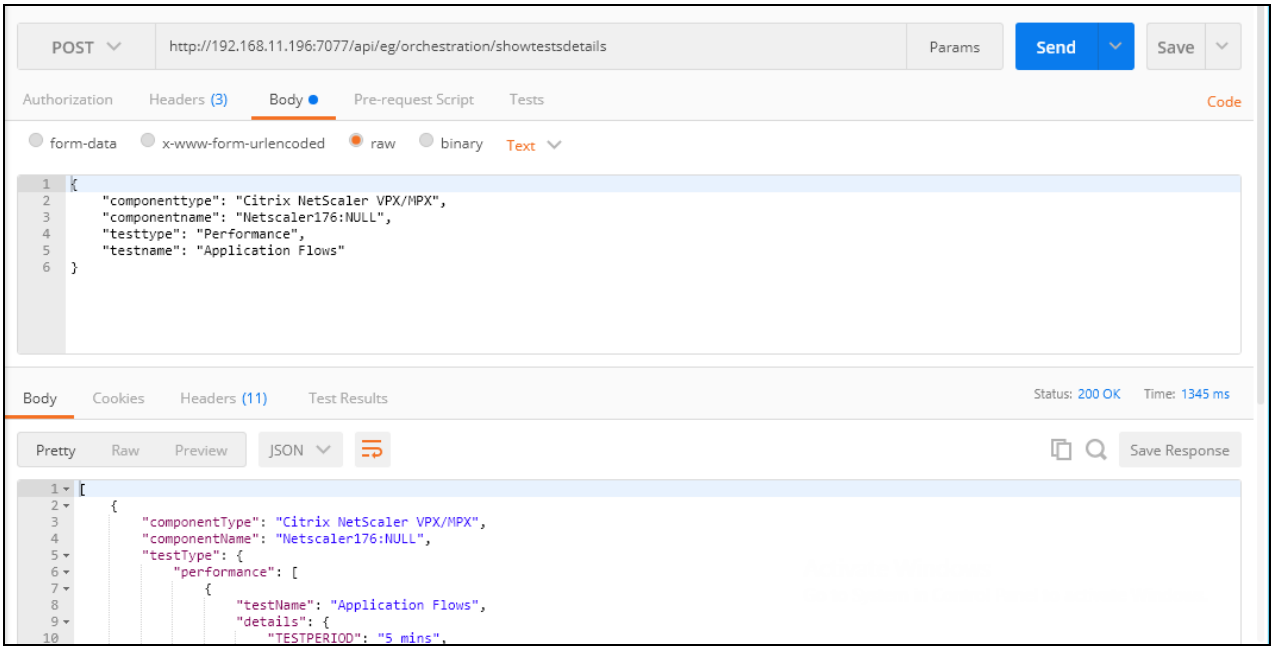

Figure 2.75: Displaying the details of a test using Postman REST Client

2.38.1 Displaying the Details of the Tests using cURL

To display the details of a test through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/showtestsdetails" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'componenttype':'Component type',
'componentname':'Component name', 'testtype':'Performance / Configuration',
'testname':'Test name'}"
```

Note that the command specified above contains both the Default and Optional key values. Figure 2.76 shows an example of displaying the details of a test using cURL.

Figure 2.76: Displaying the details of a test using cURL

2.39 Displaying Test Names for a Component Type

Administrators can use this API to obtain all the performance/configuration tests pertaining to a chosen Component Type.

URL: http://192.168.8.206:7077/api/eg/orchestration/showtests

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with Default key values:
		{
		"componenttype:"oracle database"
		}
		Example with Default and Optional Key values:
		{
		"componenttype":"oracle database",
		"category":"enabled",
		"testtype":"performance"
		}

Body	**Default:**
	{
	"componenttype":"Component type"
	}	

	Optional:
	{
	"category":"Enabled/Disabled/All",
	"testtype":"Performance/Configuration"
	}	

Success Response

Type	Code	Content
JSON	200	{ "enabledTests": ["Drives", "Drives Capacity", "Environment Entries", "Hotfix/Patch", "IP Settings Configuration", "IPC Semaphores Configuration",

Type	Code	Content
		"IPC Shared Memory Configuration", "Network Adapters Configuration", "Operating System", "Oracle Audit", "Oracle Automatic Storage Management", "Oracle Backup", . . .], "disabledTests": ["File Information"] }

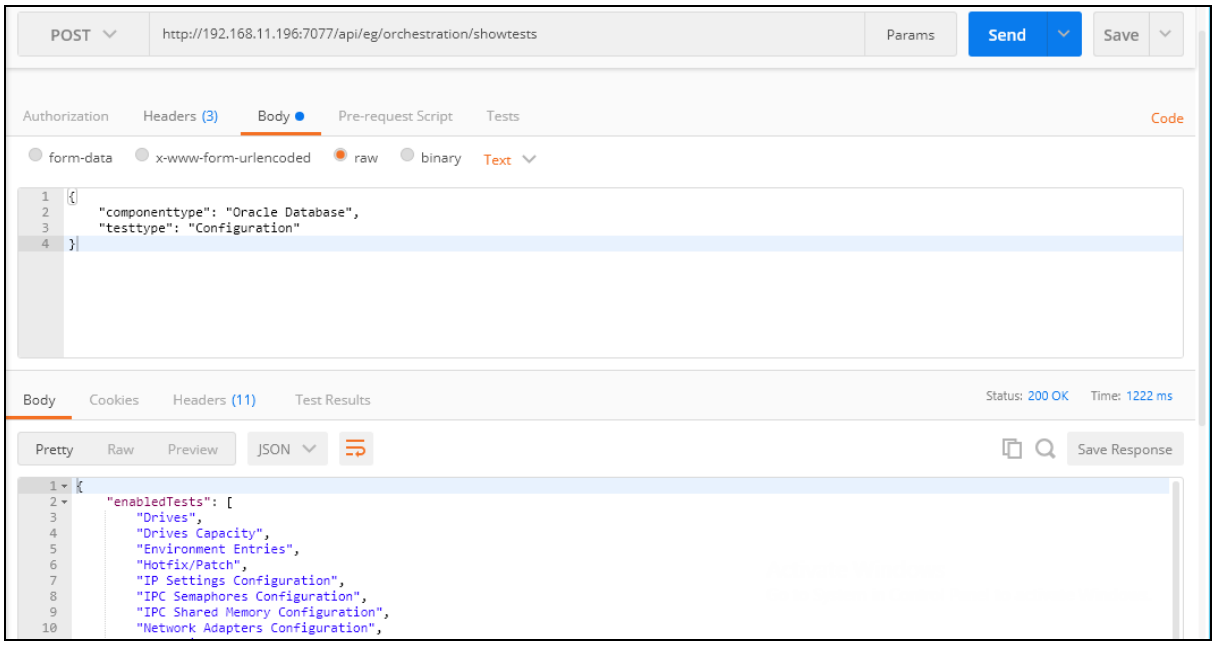

Figure 2.77: Displaying the tests for a chosen Component Type using Postman REST Client

2.39.1 Displaying Test Names for a Component Type using cURL

To display the tests for a chosen Component Type through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/showtests" -H "managerurl:http://<eG Manager IP:Port>"-H
"user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "Content-
Type: application/json" --data-raw "{'componenttype':'Component type',
'category':'Enabled/Disabled/All', 'testtype':'Performance / Configuration'}"
```

Note that the command specified above contains both the Default and Optional key values. Figure 2.78 shows an example of displaying the tests for a chosen Component Type using cURL.

Figure 2.78: Displaying the tests for a chosen Component Type using cURL

2.40 Disassociating Agents from Managers in a Redundant Setup

Use this API to unassign agents from the eG managers in a redundant setup.

URL: http://192.168.8.206:7077/api/eg/orchestration/unassignagents

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"managerip:"192.168.8.104",
		"agents":"gen1,ora8"
		}

Body	**Default:**
	{
	"managerip":"IP of the eG manager from which agents are to be delinked",
	"agents":"Comma-separated list of agents"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "one or more agents unassigned successfully."
		}

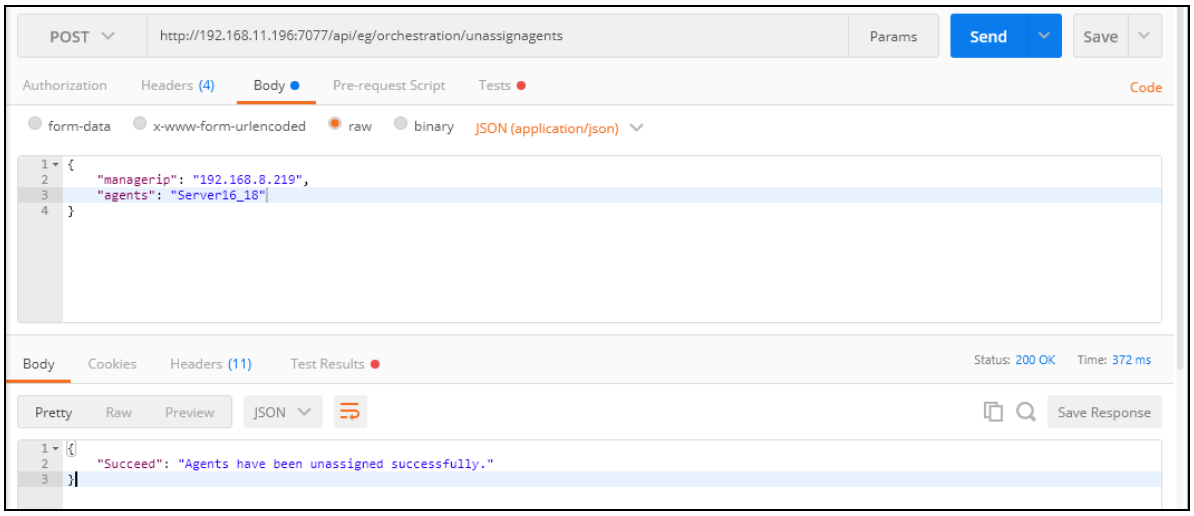

Figure 2.79: Unassign agents from the eG managers in a redundant setup using Postman REST Client

2.40.1 Disassociating Agents from Managers in a Redundant Setup using cURL

To unassign agents from the eG managers in a redundant setup through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/unassignagents" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'managerip':'IP of the eG manager from
which agents are to be delinked', 'agents':'Comma-separated list of agents'}"
```

Figure 2.80 shows an example of unassigning agents from the eG managers in a redundant setup using cURL.

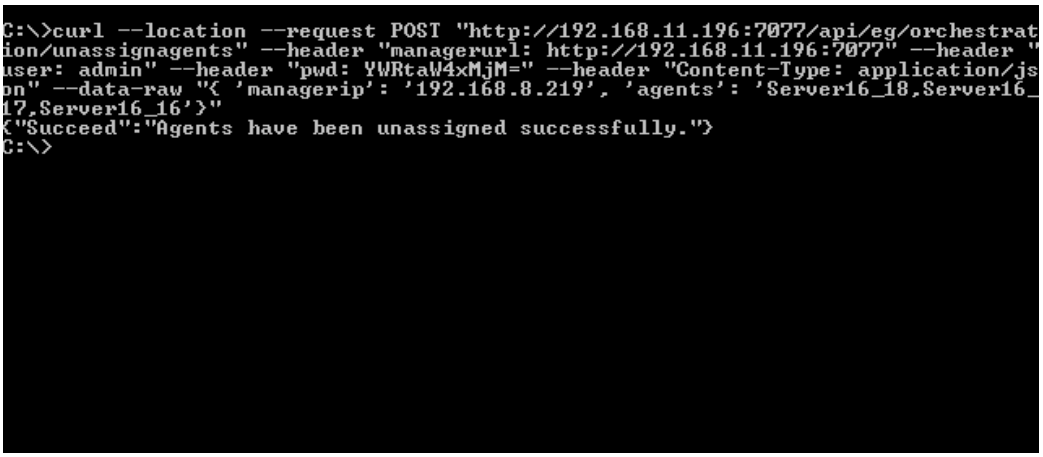

Figure 2.80: Unassigning agents from the eG managers in a redundant setup using cURL

2.41 Unmanaging a Component

To unmanage a component from the target environment, use this API.

URL: http://192.168.8.206:7077/api/eg/orchestration/unmanagecomponent

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		Example with both Default and Optional Key Values:
		{
		"componenttype":"Oracle Database",
		"componentname":"oracleDB",
		"port":"1521",
		"sid":"egurkha"
		}
		Example with Default key values:
		{
		"componenttype":"Microsoft SQL",

Parameters	Key values	Example

Body	**Default:**
	{
	"componenttype":"component type",
	"componentname":"Nick name",
	"port":"Port",
	}
	Optional:
	{
	"sid":"SID"
	}
		"componentname":"MSSQL",
		"port":"1433"
		}

Note:

If an Oracle Database server is added with multiple SIDs, then the eG Enterprise system will monitor each SID as a different Oracle Database server. Therefore, while unmanaging an Oracle Database server that supports multiple SIDs, you cannot issue a single command to remove all the SIDs at one shot. Instead, this command should be invoked separately for each SID.

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component has been unmanaged successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Please enter the SID of the component."
	}

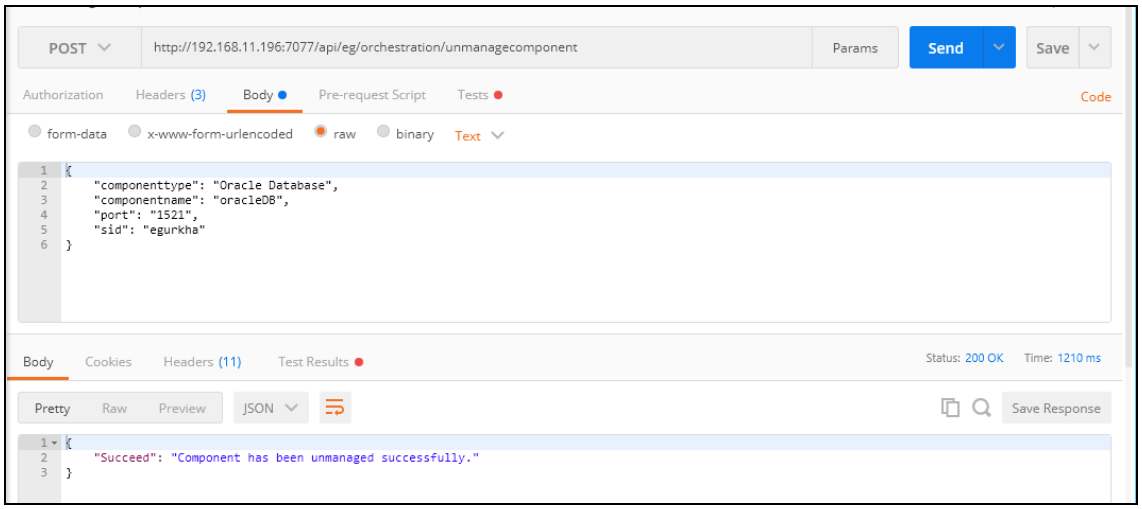

Figure 2.81: Unmanaging a component using Postman REST Client

2.41.1 Unmanaging a Component using cURL

To unmanage a component through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/unmanagecomponent" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: application/json" --data-raw "{'componenttype':'component type',
'componentname':'Nick name', 'port':'Port', 'sid':'SID'}"
```

Note that the command specified above contains both the Default and Optional key values. Figure 2.82 shows an example of unmanaging a component using cURL.

Figure 2.82: Unmanaging a component using cURL

Chapter 3: Performing Operations in Bulk Using eG REST API

One of the key benefits of the eG REST API is that, administrators are allowed to process commands in bulk, thus allowing to perform multiple tasks simultaneously without logging into the eG console - for instance, you can add multiple components at one shot using the eG REST API. This saves administrators the time and trouble involved in performing redundant tasks.

To execute commands in bulk, the eG REST API requires a CSV file that contains the details of the operations to be performed. This file (CSV) should be created on the eG Manager on which the operations are to be performed.

Once the file is created, invoke the relevant eG REST API command from the RESTClient by providing the manager ID and the full path to the CSV file. The command will then read the instructions defined in the CSV file and then execute them on the specified manager to perform the operation.

Note:

To ensure that the CSV file supports multi-byte component names and agent nick names, ensure that you save the CSV file in the UTF-8 mode.

The sections below discuss how the CSV file can be used for performing multiple administrative operations on an eG manager.

3.1 Adding Components in Bulk

For every administrative operation to be performed in bulk by the eG REST API, separate CSV files should be created. This implies that for adding new components to the eG Enterprise system, a dedicated CSV file is required.

To configure a CSV file with the details of the components to be added, entries of the following format should be included in that file:

```
Element,action
Component,add
componenttype,hostip/hostname,componentname,port,externalagents
<Details of component1>
<Details of component2>
.
.
.
```

For example, if you want to add 3 IIS web servers to the eG Enterprise system using the host IP, do the following:

```
Element,action
Component,add
componenttype,hostip,componentname,port,mtsenabled,externalagents
IIS web,192.168.10.96,iis96,80,no,ext43
IIS web,192.168.10.173,iis173,7077,no,ext173
IIS web,192.168.10.90,web90,80,no,ext85
```

If you want to add 3 IIS web servers to the eG Enterprise system using the host name of the component, do the following:

```
Element,action
Component,add
componenttype,hostname,componentname,port,mtsenabled,externalagents
IIS web,egurkha25,iis96,80,no,ext43
IIS web,egurkha26,iis173,7077,no,ext173
IIS web,egurkha27,web90,80,no,ext85
```

Note that the column names (componenttype, hostip, etc.) used here are the same as the input parameters of the 'addcomponent' command supported by eG REST API (see Section **2.1** of this document). These column names cannot be changed. Also, while providing the details of the components to be added, ensure that you follow the same order of the column names.

While adding components of different types or those which support different parameter sets, make sure that you leave the columns not applicable for a component specification, blank. At the same time, ensure that the column names you specify in the CSV file are a super-set of the parameters supported by all the components that are being added. In other words, the column names provided in the CSV file should correspond to the following:

- the parameters that are common across all the components to be added, and;
- the parameters that are distinct/unique for each of the components being added;

For instance, you can add an IIS web server and a Microsoft Windows component using the same CSV file, with the help of the following specification:

```
Element,action
Component,add
componenttype,hostip,componentname,port,mtsenabled,externalagents IIS
web,192.168.10.96,iis96,80,no,ext43
Microsoft Windows,192.168.10.173, win173,,,ext180
```

In this case, note that the columns componenttype, hostip, componentname, and externalagents are common for both the IIS web server and the Microsoft Windows server, but the port and mtsenabled columns are supported only by the IIS web server. Moreover, since the Microsoft Windows server is a non-port-based component and does not support the mtsenabled parameter, the columns port and mtsenabled have been left blank in the specification that corresponds to the Microsoft Windows server.

If you want to say, associate multiple external agents with a component, then your specification should include a comma-separated list of external agents provided within double-quotes:

```
Element,action
Component,add
componenttype,hostip,componentname,port,mtsenabled,externalagents
IIS web,192.168.10.96,iis96,80,no,"ext43,ext60"
IIS web,192.168.10.173,iis173,7077,no,ext173
```

Similarly, you can add an Oracle database sever with multiple SIDs.

Note:

If an Oracle database server with multiple SIDs is added to the eG Enterprise system, then each SID will be registered as a separate Oracle database server in the eG Enterprise system.

Once all the required entries are defined in the CSV file (let's say, the name of the file is *addcomponent.csv*), execute the command specified in the URL of the table below to extract the component information from the file, connect to the required eG manager, and add the specified components to the eG Enterprise system.

Note:

A few key values of the Body parameter are optional. These optional key values are mentioned separately in the below table.

URL: http://192.168.8.206:7077/api/eg/orchestration/addcomponent/bulk

Method: POST

Content-Type: multipart/form-data

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"file": G:\addcomponent.csv
		}

Body	**Default:**
	{
	"file" : "Full path to the CSV file"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component has been added successfully."
		}

Failure Response

Type	Content
JSON	{ "Error": "Following component(s) could not be configured.", "Description": [

Type	Content
	"Component already exist under this type. Component type :<Component Type>,Component name :<hostname of the Component>",] }

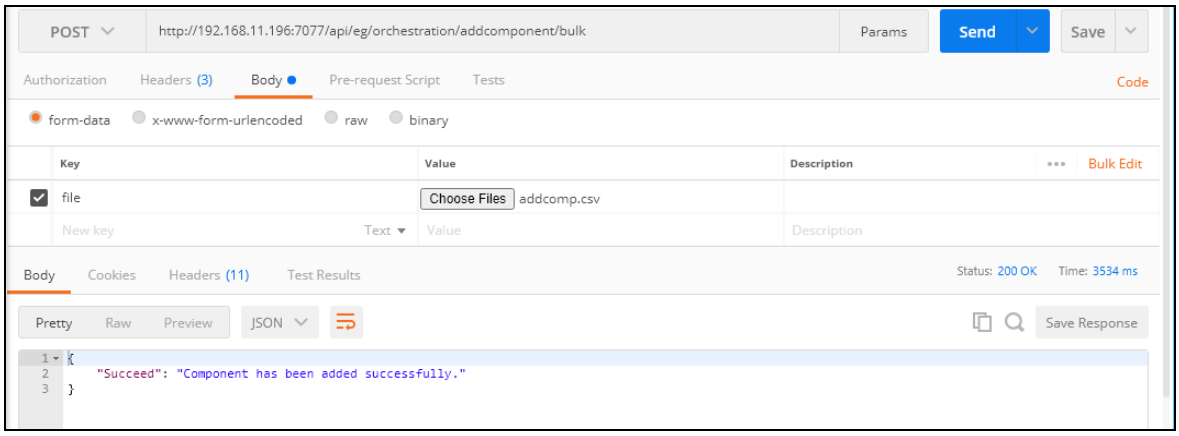

Figure 3.1: Example to add components in bulk using Postman REST Client

3.1.1 Adding Components in Bulk using cURL

To add components in bulk through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/addcomponent/bulk" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: multipart/form-data" --form "file=@ Full path to the CSV file"
```

Figure 3.2 shows an example of adding components in bulk using cURL.

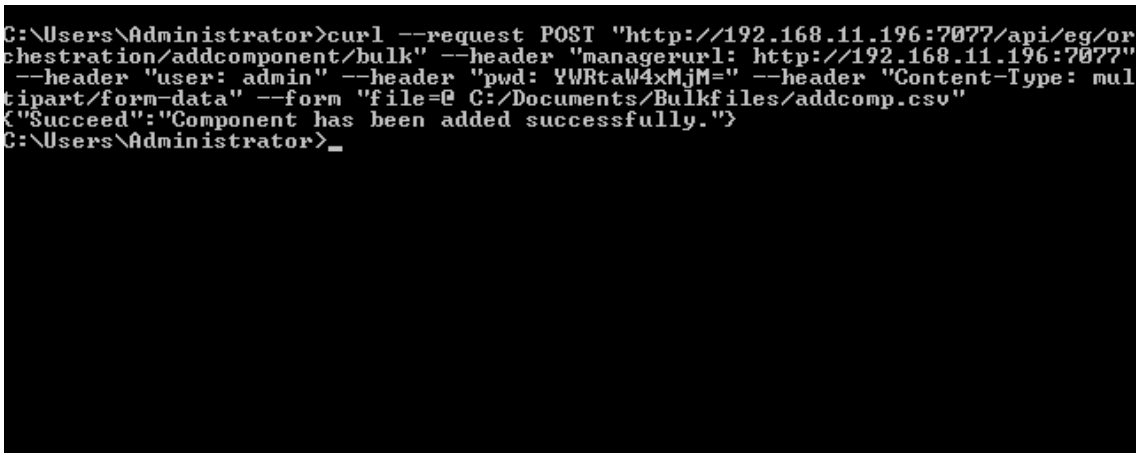

Figure 3.2: Adding components in bulk using cURL

3.2 Managing Components in Bulk

To manage components in bulk, you can create a dedicated CSV file for this purpose and configure it with entries related to each component to be managed. Given below is the format of the entries in such a file:

```
Element,action
Component,manage
componenttype,componentname,port,sid
<Component1 to be managed>
<Component2 to be managed>
.
.
.
```

For instance:

```
Element,action
Component,manage
componenttype,componentname,port,sid
Microsoft Windows,win1,,
AGate,agte10,3900,
Oracle Database,ora55,1521,multi
```

Note that the column names used in the CSV file are the same as the input parameters of the 'ManageComponent' command supported by the eG REST API.

Note:

- If an Oracle database server with multiple SIDs is to be managed, then the entry for the Oracle server in your CSV file should not include a comma-separated list of SIDs; instead, you should provide a separate entry for each SID to be managed.
- If one/more column names in your CSV file are not applicable to a component specification, then make sure that such columns are left empty in the specification.

Once the CSV file (let's say, the name of the file is *managecomponent.csv*) is created on the eG manager, invoke the URL command mentioned in the below table from the eG REST API Client to update the eG manager with all the modifications contained in the CSV file.

URL: http://192.168.8.206:7077/api/eg/orchestration/managecomponent/bulk

Method: POST

Content-Type: multipart/form-data

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"file": G:\managecomponent.csv
		}

Body	Default:
	{
	"file" : "Full path to the CSV file"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component has been managed successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Following component(s) could not be managed.",
	"Description": [
	"The selected component does not exist. Component type :<Component TYpe>,Component name :<hostname of the component>",
	]
	}

3.2.1 Managing Components in Bulk using cURL

To manage components in bulk through the REST API using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/addcomponent/bulk" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: multipart/form-data" -F "file=@Full path to the CSV file"
```

Figure 3.4 shows an example of managing components in bulk using cURL.

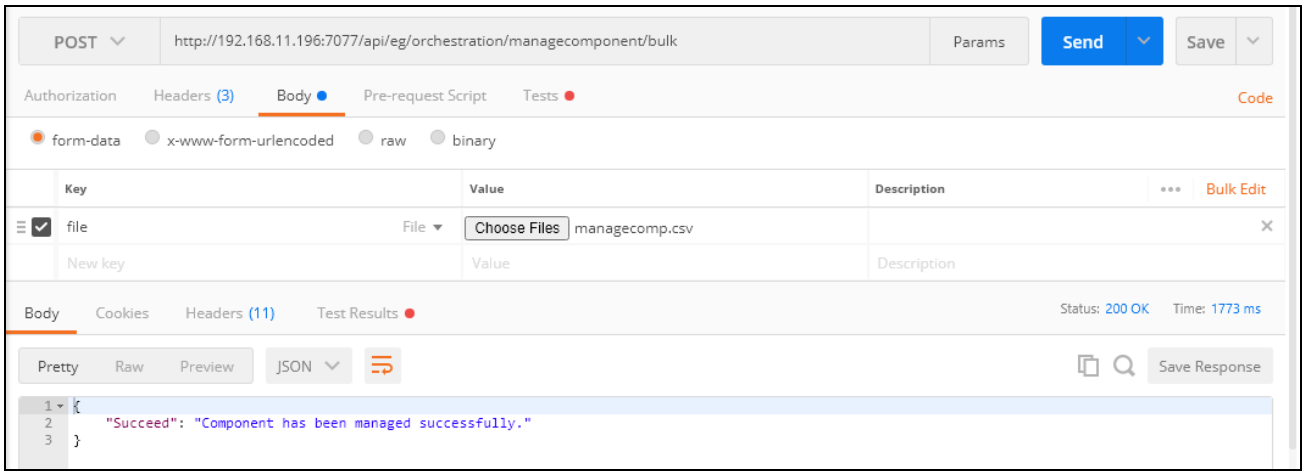

Figure 3.3: Managing components in bulk using cURL

3.2.2 Managing Components in Bulk using cURL

To manage components s in bulk through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/managecomponent/bulk" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: multipart/form-data" --form "file=@ Full path to the CSV file"
```

Figure 3.4 shows an example of managing components in bulk using cURL.

Figure 3.4: Managing Components in bulk using cURL

3.3 Modifying Components in Bulk

As already mentioned, an exclusive CSV file should be created to handle bulk modifications to component details.

Entries in this file should be of the following format:

```
Element,action
Component,modify
componenttype,hostip/hostname,componentname,port,externalagents
<Modification to component1>
<Modification to component2>
.
.
.
```

For example, if you want to modify the port numbers of 2 IIS web servers, do the following:

```
Element,action
Component,modify
componenttype,hostip,componentname,oldport,Newport,mtsenabled,externalagents
IIS web,192.168.10.96,iis96,7077,8088,no,ext43
IIS web,192.168.10.173,iis173,7077,7078,no,ext173
```

The CSV file can also be used to modify the details of components of multiple types at one shot. While doing so, make sure that you leave the columns not applicable for a component specification, blank. At the same time, ensure that the column names you specify in the CSV file are a super-set of the parameters supported by all the components that are being modified. In other words, the column names provided in the CSV file should correspond to the following:

- the parameters that are common across all the components to be modified, and;
- the parameters that are distinct/unique for each of the components being modified;

For instance, say you want to modify the nick name of an Oracle database server, and want to change the monitoring mode of an MS SQL server from agent-based to agentless. The specification in this case will be, as follows:

```
Element,action
Component,modify
componenttype,hostip,componentname,oldcomponentname,
newcomponentname,port,sid,agentless,mode,os,externalagents,remoteagent
Oracle database,192.168.10.8,,ora8,ora08,1521,egora,,,,ext125,
Microsoft SQL,192.168.10.63, sql63,,,1433,,yes,perfmon,Windows2012,ext173,rem12
```

In the above specification, you can find that the column list includes the following:

- parameters such as componenttype, hostip/hostname, port, and externalagents that are common to both the Oracle and MS SQL servers
- the oldcomponentname, newcomponentname, and sid parameter that are available only for the Oracle component
- the componentname, agentless, mode, OS, and remoteagent parameters that are relevant for only the MS SQL server being modified

From the above specification, it is also evident that columns not applicable to a component specification have been left blank.

If say, you want to add multiple SIDs to an Oracle database server, your specification should be as follows:

```
Element,action
Component,modify
componenttype,hostip,componentname,port,sid,externalagents
Oracle database,192.168.10.8,ora08,1521,"egora,egoracle",ext125
```

Note:

If an Oracle database server with multiple SIDs is added to the eG Enterprise system, then each SID will be registered as a separate Oracle database server in the eG Enterprise system.

Once the CSV file (let's say, the name of the file is *modifycomponent.csv*) is created on the eG manager, invoke the URL command mentioned below from the eG REST API Client to update the eG manager with all the modifications contained in the CSV file.

URL: http://192.168.8.206:7077/api/eg/orchestration/modifycomponent/bulk

Method: POST

Content-Type: multipart/form-data

Inputs to be Specified

Parameters	Key values	Example

Header	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>	{
		"file": G:\modifycomponent.csv
		}

Parameters	Key values	Example
	user: eG username or domain/eG username pwd: Base64 encoded password	

Body	**Default:**
	{
	"file" : "Full path to the CSV file"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component has been modified successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Following component(s) could not be modifed.",
	"Description": [
	"The old component name or port does not exist. Component type :<Component Type>,Component name :<Component name>",
	"The old component name or port does not exist. Component type :<Component Type>,Component name :<Component name>",
	]
	}

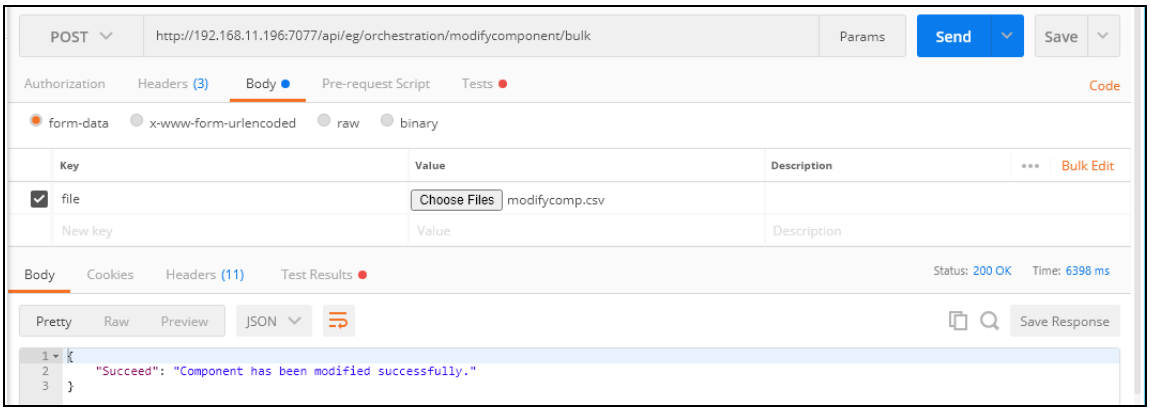

Figure 3.5: Modifying Components in bulk using Postman REST Client

3.3.1 Modifying Components in Bulk using cURL

To modify components in bulk through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/modifycomponent/bulk" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: multipart/form-data" --form "file=@ Full path to the CSV file"
```

Figure 3.6 shows an example of modifying components in bulk using cURL.

Figure 3.6: Modifying Components in bulk using cURL

3.4 Deleting Components in Bulk

The components to be deleted simultaneously from the eG Enterprise system should be included in a CSV file that is created exclusively for this purpose.

Such a CSV file should contain entries of the following format:

```
Element,action
Component,delete
componenttype,hostip/hostname,componentname,port
<Component1 to be deleted>
<Component2 to be deleted>
.
.
.
```

For example, if you want to delete an Oracle and an MS SQL server together using their respective host IPs, then your CSV file should include the following entries:

```
Element,action
Component,delete
componenttype,hostip,componentname,port,sid
Oracle database,192.168.10.96,ora96,1521,egora
Microsoft SQL,192.168.10.173,sql173,1433,
```

If you want to delete an Oracle and an MS SQL server together using their respective host names, then your CSV file should include the following entries:

```
Element,action
Component,delete
componenttype,hostname,componentname,port,sid
Oracle database,egurkha25,ora96,1521,egora
Microsoft SQL,egurkha26,sql173,1433,
```

As already stated, if an Oracle database server is added with multiple SIDs, then the eG Enterprise system will monitor each SID as a different Oracle server. Therefore, while removing an Oracle database server that supports multiple SIDs, each SID should be treated as a different Oracle server, and a separate specification for each SID should be included in the CSV file. For example, say, an Oracle database server has been added with the following SIDs: egdemo,egora. To remove this Oracle server completely, your CSV file should contain the following entries:

```
Element,action
Component,delete
componenttype,hostip,componentname,port,sid
Oracle database,192.168.10.96,ora96,1521,egora
Oracle database,192.168.10.96,ora96,1521,egdemo
```

Note:

Components included in a zone, segment, or service cannot be deleted.

Once the CSV file (let's say, the name of the file is *deletecomponent.csv*) is created on the eG manager, invoke the URL command mentioned below from the eG REST API Client to update the eG manager with all the modifications contained in the CSV file.

URL: http://192.168.8.206:7077/api/eg/orchestration/deletecomponent/bulk

Method: POST

Content-Type: multipart/form-data

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"file": G:\deletecomponent.csv
		}

Body	Default:
	{
	"file" : "Full path to the CSV file"
	}	

Success Response

Type	Content
JSON	{ "Succeed": "Component has been removed successfully."

Type	Content
	}

Failure Response

Type	Content

JSON	{
	"Error": "Following component(s) could not be deleted.",
	"Description": [
	"The selected component does not exist. Component type :<Component Type>,Component name :<nick name of the component>",
	"The selected component does not exist. Component type :<Component Type>,Component name :<nick name of the component>",
	.
	.
	.
	]
	}

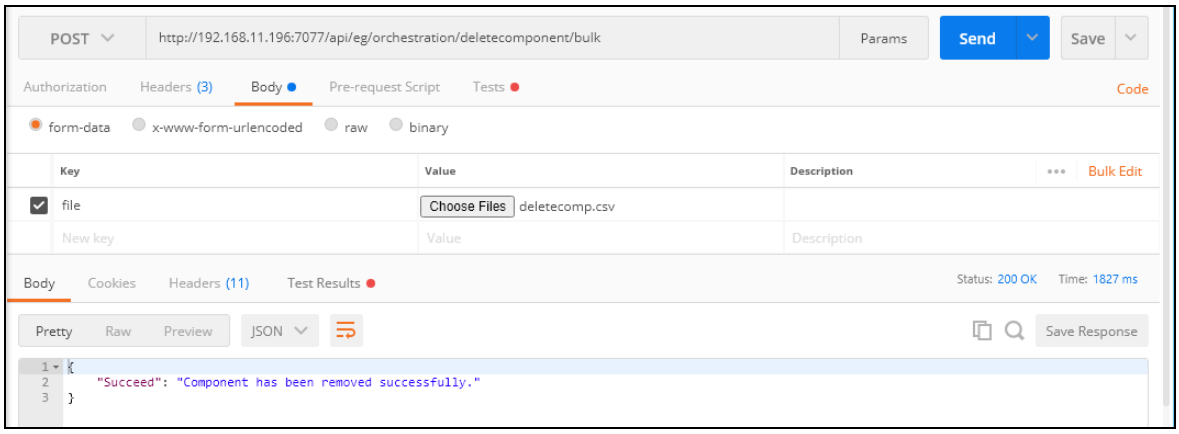

Figure 3.7: Deleting Components in bulk using Postman REST Client

3.4.1 Deleting Components in Bulk using cURL

To delete components in bulk through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/deletecomponent/bulk" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: multipart/form-data" --form "file=@ Full path to the CSV file"
```

Figure 3.8 shows an example of deleting components in bulk using cURL.

Figure 3.8: Deleting components in bulk using cURL

3.5 Unmanaging Components in Bulk

The components to be unmanaged should be included in a CSV file that is created exclusively for this purpose.

Such a CSV file should contain entries of the following format:

```
Element,action
Component,unmanage
componenttype,componentname,port,sid
<Component1 to be managed>
<Component2 to be managed>
.
.
.
```

For instance:

```
Element,action
Component,unmanage
componenttype,componentname,port,sid
Microsoft Windows,win1,,
AGate,agte10,3900,
Oracle Database,ora55,1521,multi
```

Note that the column names used in the CSV file are the same as the input parameters of the 'unmanagecomponent' command supported by the eG REST API.

Note:

- Components included in a zone, segment, or service cannot be unmanaged.
- If an Oracle database server with multiple SIDs is to be managed, then the entry for the Oracle server in your CSV file should not include a comma-separated list of SIDs; instead, you should provide a separate entry for each SID to be managed.
- If one/more column names in your CSV file are not applicable to a component specification, then make sure that such columns are left empty in the specification.

Once the CSV file (let's say, the name of the file is *unmanagecomponent.csv*) is created on the eG manager, invoke the URL command mentioned in the below table from the eG REST API Client to update the eG manager with all the modifications contained in the CSV file.

URL: http://192.168.8.206:7077/api/eg/orchestration/unmanagecomponent/bulk

Method: POST

Content-Type: multipart/form-data

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"file": G:\unmanagecomponent.csv
		}

Parameters	Key values	Example

Body	**Default:**
	{
	"file" : "Full path to the CSV file"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Component has been unmanaged successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Following component(s) could not be unmanaged.",
	"Description": [
	"The selected component does not exist. Component type :<Component Type>,Component name :<host name of the Component>",
	"The selected component does not exist. Component type <Component Type>,Component name :<host name of the Component>",
	"The selected component does not exist. Component type :<Component Type>,Component name :<host name of the Component>"
	]
	}

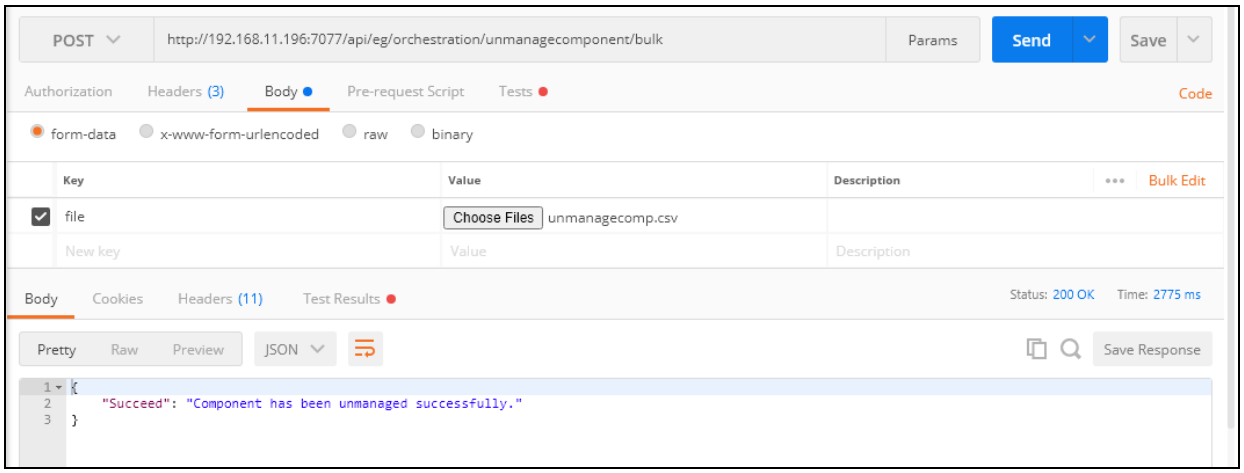

Figure 3.9: Unmanaging Components in bulk using Postman REST Client

3.5.1 Unmanaging Components in Bulk using cURL

To unmanage components in bulk through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/unmanagecomponent/bulk" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: multipart/form-data" --form "file=@ Full path to the CSV file"
```

Figure 3.10 shows an example of unmanaging components in bulk using cURL.

Figure 3.10: Unmanaging components in bulk using cURL

3.6 Adding Remote Agents in Bulk

The CSV file created specifically for adding multiple remote agents to the eG Enterprise system, should contain the following entries:

```
Element,action
RemoteAgent,add
hostip,agentname
<Details of remoteagent1>
<Details of remoteagent2>
<Details of remoteagent3>
.
.
.
```

For instance:

```
Element,action
RemoteAgent,add
hostip,agentname
192.168.10.8,rem8
192.168.10.10,rem10
192.168.10.12,lin12
```

Note that the column names used in the CSV file are the same as the input parameters of the 'addremoteagent' command supported by the eG REST API.

Once the CSV file (let's say, the name of the file is *addremagent.csv*) is created on the eG manager, invoke the URL command mentioned in the below table from the eG REST API Client to update the eG manager with all the modifications contained in the CSV file.

URL: http://192.168.8.206:7077/api/eg/orchestration/addremoteagent/bulk

Method: POST

Content-Type: multipart/form-data

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e.,	{ "file": G:\addremagent.csv

Parameters	Key values	Example
	http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	}

Body	**Default:**
	{
	"file" : "Full path to the CSV file"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Remote agent has been added successfully."
		}

Failure Response

Type	Content
JSON	{ "Error": "Following remote agent(s) could not be added.", "Description": ["The agent name you are trying to add already exists. Please use another agent name. Agent name :<name of the remote agent>", "The agent name you are trying to add already exists. Please use another agent name. Agent name :<name of the remote agent>"

Type	Content
	] }

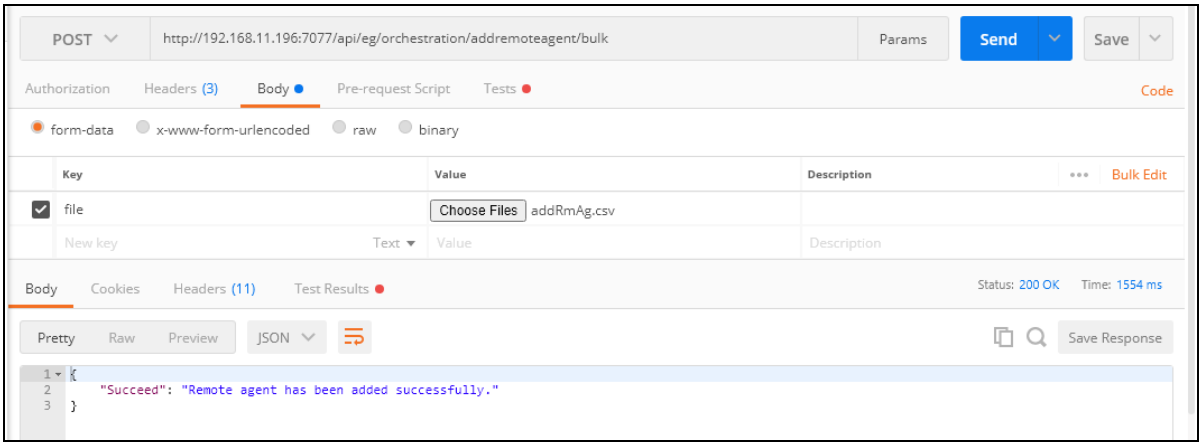

Figure 3.11: Example to add remote agents in bulk

3.6.1 Adding Remote Agents in Bulk using cURL

To add remote agents in bulk through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/addremoteagent/bulk" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: multipart/form-data" --form "file=@ Full path to the CSV file"
```

Figure 3.12 shows an example of adding remote agents in bulk using cURL.

Figure 3.12: Adding remote agents in bulk using cURL

3.7 Adding External Agents in Bulk

The CSV file created specifically for adding multiple remote agents to the eG Enterprise system, should contain the following entries:

```
Element,action
ExternalAgent,add
Hostip/hostname,agentname
Details of extenalagent1>
<Details of extenalagent2>
<Details of extenalagent3>
.
.
.
```

For instance:

```
Element,action
ExternalAgent,add
hostip,agentname
192.168.10.8,ext8
192.168.10.10,ext10
192.168.10.12,lin12
```

If you use the host name instead of hostip, then your specification should be:

```
Element,action
ExternalAgent,add
hostname,agentname
egurkha25,ext8
egurkha26,ext10
egurkha27,lin12
```

Note that the column names (hostip,hostname,etc.) used in the CSV file are the same as the input parameters of the 'addexternalagent' command supported by the eG REST API.

If the eG license enables the client emulation capability, then the CSV file should include an additional clientemulation column. Therefore, if you want to add two external agents - one to be used for client emulation and another that is not used for client emulation - then, your CSV specification should be as follows:

```
Element,action
ExternalAgent,add
hostip,agentname,clientemulation
192.168.10.8,ext8,yes
192.168.10.10,ext10,no
```

Once the CSV file (let's say, the name of the file is *addextagent.csv*) is created on the eG manager, invoke the URL command mentioned in the below table from the eG REST API Client to update the eG manager with all the modifications contained in the CSV file.

URL: http://192.168.8.206:7077/api/eg/orchestration/addexternalagent/bulk

Method: POST

Content-Type: multipart/form-data

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"file": G:\addextagent.csv
		}

Body	**Default:** { "file" : "Full path to the CSV file"	

Parameters	Key values	Example
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "External agent has been added successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Following external agent(s) could not be add.",
	"Description": [
	"The agent name you are trying to add already exists. Please use another agent name. Agent name :<name of the external agent>",
	"The agent name you are trying to add already exists. Please use another agent name. Agent name :<name of the external agent>"
	]
	}

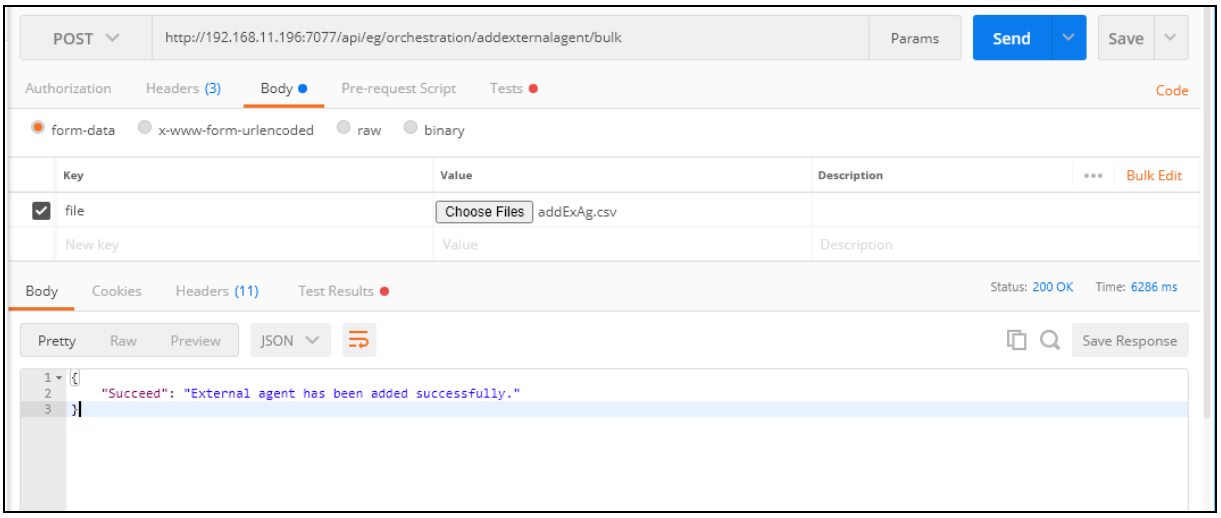

Figure 3.13: Example to add external agents in bulk using Postman REST Client

3.7.1 Adding External Agents in Bulk using cURL

To add external agents in bulk through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/addexternalagent/bulk" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: multipart/form-data" --form "file=@ Full path to the CSV file"
```

Figure 3.14 shows an example of adding external agents in bulk using cURL.

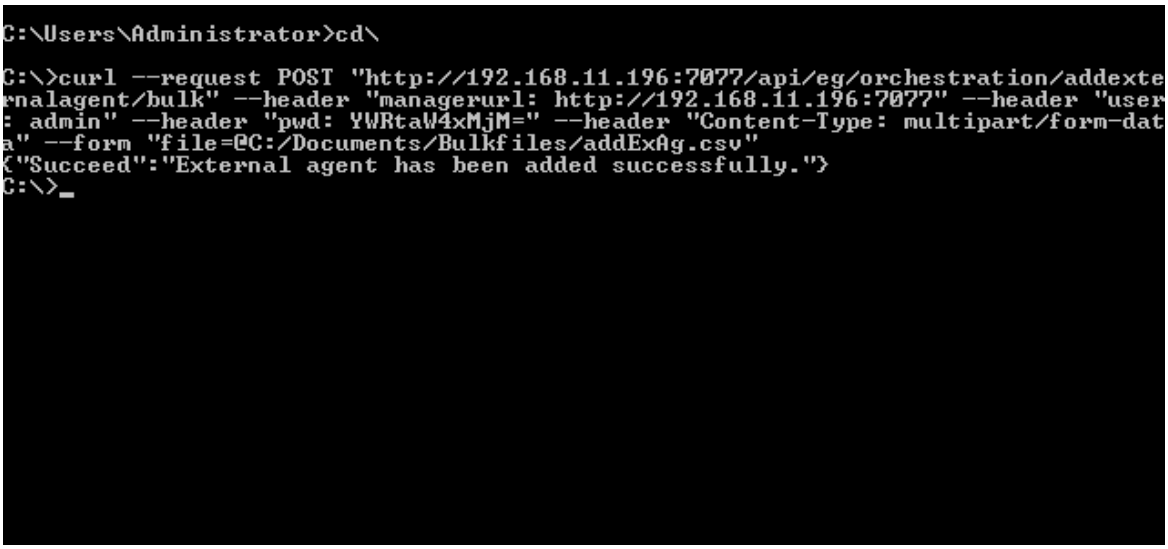

Figure 3.14: Adding external agents in bulk using cURL

3.8 Deleting Remote Agents in Bulk

The CSV file that is specifically created for deleting a number of remote agents in bulk, should contain the following entries:

```
Element,action
RemoteAgent,delete
agentname
<Nickname of remote agent1>
<Nickname of remote agent2>
<Nickname of remote agent3>
.
.
.
```

For instance:

```
Element,action
RemoteAgent,delete
agentname
rem8
rem10
external12
```

Once the CSV file (let's say, the name of the file is *deleteremagent.csv*) is created on the eG manager, invoke the URL command mentioned below from the eG REST API Client to update the eG manager with all the modifications contained in the CSV file.

URL: http://192.168.8.206:7077/api/eg/orchestration/deleteremoteagent/bulk

Method: POST

Content-Type: multipart/form-data

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or
		{
		"file": G:\deleteremagent.csv
		}

Parameters	Key values	Example
	domain/eG username pwd: Base64 encoded password	

Body	**Default:**
	{
	"file" : "Full path to the CSV file"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "Remote agent has been deleted successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Following remote agent(s) could not be deleted.",
	"Description": [
	"The remote agent you are trying to delete does not exist. Agent name :<name of the remote agent>",
	"The remote agent you are trying to delete does not exist. Agent name :<name of the remote agent>"
	]
	}

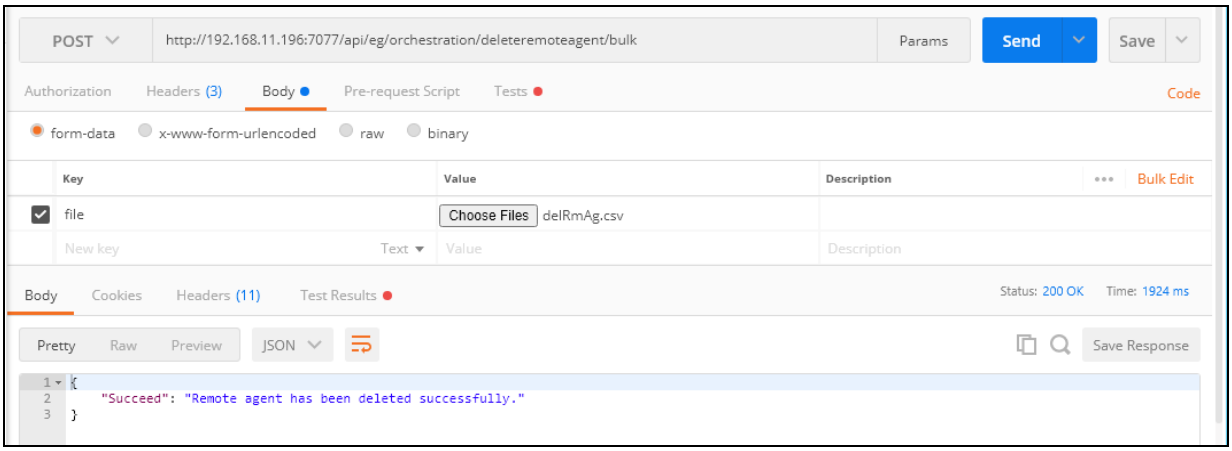

Figure 3.15: Deleting remote agents in bulk using Postman REST Client

3.8.1 Deleting Remote Agents in Bulk using cURL

To delete remote agents in bulk through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/deleteremoteagent/bulk" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: multipart/form-data" --form "file=@ Full path to the CSV file"
```

Figure 3.16 shows an example of deleting remote agents in bulk using cURL.

Figure 3.16: Deleting remote agents in bulk using cURL

3.9 Deleting External Agents in Bulk

The CSV file that is specifically created for deleting a number of remote agents in bulk, should contain the following entries:

```
Element,action
ExternalAgent,delete
agentname
<Nickname of external agent1>
<Nickname of external agent2>
<Nickname of external agent3>
.
.
.
```

For instance:

```
Element,action
ExternalAgent,delete
agentname
ext8
ext10
ext12
```

Once the CSV file (let's say, the name of the file is *deleteexternalagent.csv*) is created on the eG manager, invoke the URL command mentioned below from the eG REST API Client to update the eG manager with all the modifications contained in the CSV file.

URL: http://192.168.8.206:7077/api/eg/orchestration/deleteexternalagent/bulk

Method: POST

Content-Type: multipart/form-data

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or
		{
		"file": G:\deleteexternalagent.csv
		}

Parameters	Key values	Example
	domain/eG username pwd: Base64 encoded password	

Body	**Default:**
	{
	"file" : "Full path to the CSV file"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Succeed": "External agent has been deleted successfully."
		}

Failure Response

Type	Content

JSON	{
	"Error": "Following external agent(s) could not be deleted.",
	"Description": [
	"The external agent you are trying to delete does not exist. Agent name :<name of the external agent>",
	"The external agent you are trying to delete does not exist. Agent name :<name of the external agent>"
	]
	}

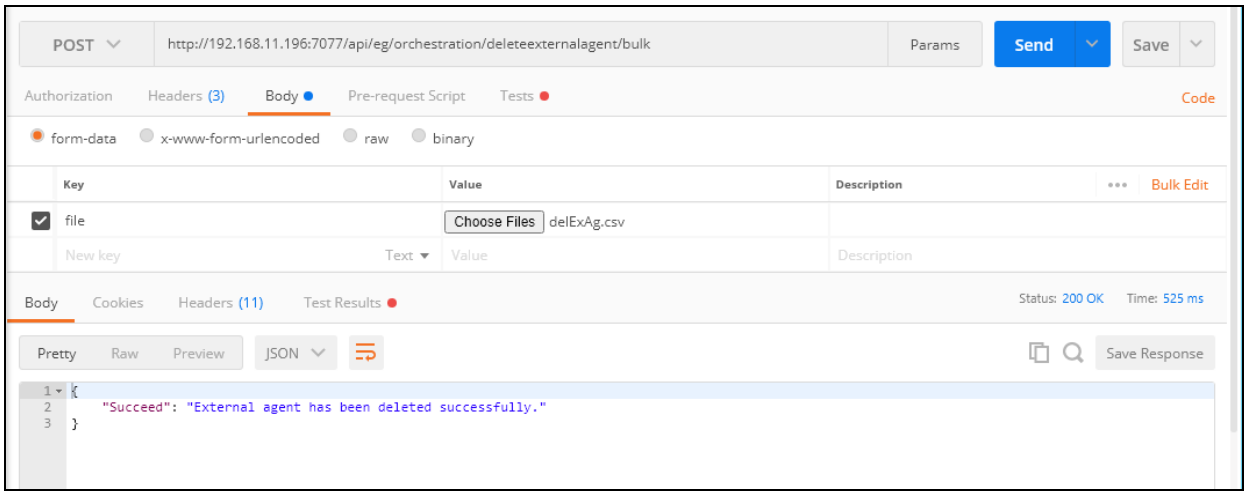

Figure 3.17: Example to delete external agents in bulk using Postman REST Client

3.9.1 Deleting External Agents in Bulk using cURL

To delete external agents in bulk through the REST API using cURL, the command should be specified in the following format:

```
curl --location --request POST "http://<eG Manager
IP:Port>/api/eg/orchestration/deleteexternalagent/bulk" -H "managerurl:http://<eG Manager
IP:Port>"-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -
H "Content-Type: multipart/form-data" --form "file=@ Full path to the CSV file"
```

Figure 3.18 shows an example of deleting external agents in bulk using cURL.

Figure 3.18: Deleting external agents in bulk using cURL

Chapter 4: Retrieving Analytical Data from eG Manager Using eG REST API

An important aspect of the eG REST API is that, apart from performing administrative activities on the eG manager, administrators are allowed to retrieve analytical data from the eG manager (for e.g., alarms raised in the target environment, the detailed diagnosis data of a chosen measure, health of the components in the target environment). Administrators can export this data to other management portals to provide a seamless user interface. This data can also be used by the administrators for different purposes such as creating more powerful dashboards, consolidation with asset / configuration tracking systems etc.

The sections below will discuss in detail on how to retrieve analytical data from the eG manager.

4.1 Retrieving Count of Alarms Raised in the Target Environment

By default, using the eG REST API, administrators can retrieve the count of alarms raised in the eG manager. The URL to retrieve the count of alarms should be in the following format:

URL:http://<eG manager IP:port>/api/eg/analytics/getAlarmCount

URL: http://192.168.8.206:7077/api/eg/analytics/getAlarmCount

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	Not Applicable

Success Response

Type	Code	Example Response

JSON	200	{
		"TOTAL": 43,
		"CRITICAL": 7,
		"MAJOR": 13,
		"MINOR": 23
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

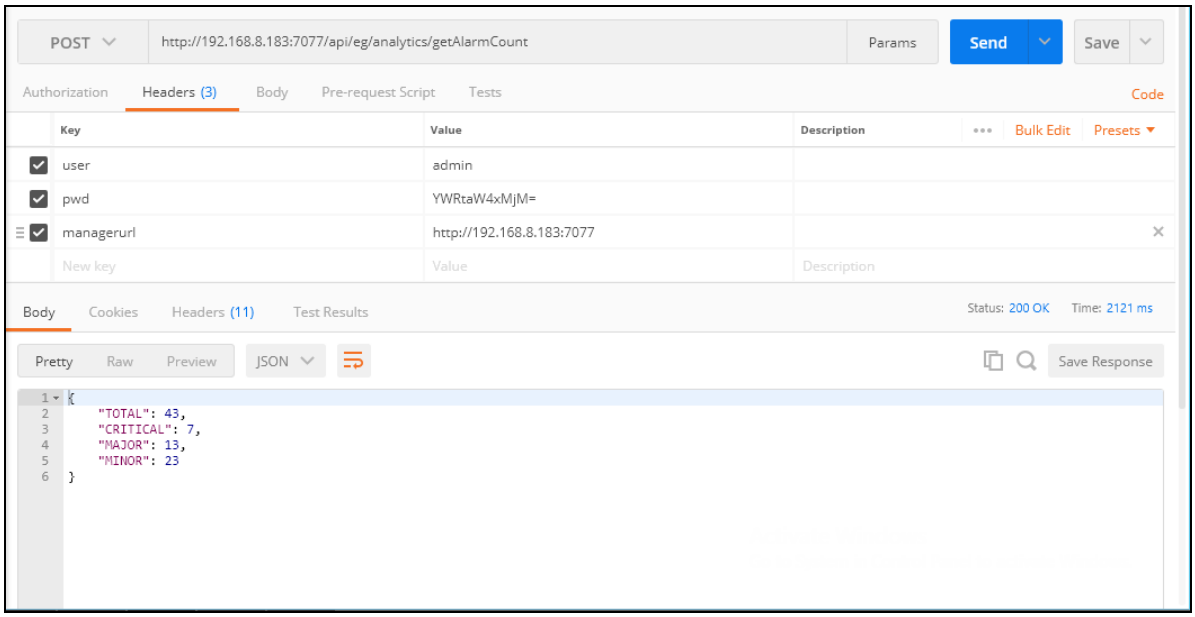

Figure 4.1: Example to retrieve current alarm count using Postman REST Client

4.1.1 Retrieving Count of Alarms Raised in the Target Environment using cURL

To retrieve the count of alarms in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager IP:Port>/api/eg/analytics/getAlarmCount"
-H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H
"managerurl:http://<eG Manager IP:Port>" --data-raw ""
```

Figure 4.2 shows an example of retrieving the count of alarms raised in the target environment using cURL.

Figure 4.2: Retrieving current alarm count in the target environment using cURL

4.2 Retrieving Live Measures of a Component

Using the eG REST API, the measures reported by executing the tests of a component during the current measurement period can be retrieved.

URL: http://<eG manager IP:port>/api/eg/analytics/getLiveMeasure

URL: http://192.168.8.206:7077/api/eg/analytics/getLiveMeasure

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"servername":"windows_136",
		"servertype":"Microsoft Windows"
		}
		Example for retrieving current measures from an Oracle Database Server:
		{
		"servername":"Oractest:1521:egora",
		"servertype":"Oracle Database Server"
		}

Body	{
	"servername":"Hostname of the component:Port",
	"servertype":"Component Type"
	}
	If current measures of the Oracle Database server is to be retrieved, then the Key values should be specified as follows:
	{
	"servername":"Hostname of the component:Port:SID",
	"servertype":"Component Type"
	}	

Success Response

Type	Code	Content

JSON	200	[{"Measure Details for windows_136:Microsoft Windows": [{
		"Disk Space": {
		"State": "GOOD",
		"Last Measurement Time": "Aug 17, 2020 00:30:36",
		"Total capacity": [{
		"State": "GOOD",
		"Value": "51098",
		"Unit": "MB"
		}]
		}
		}]
		}]

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

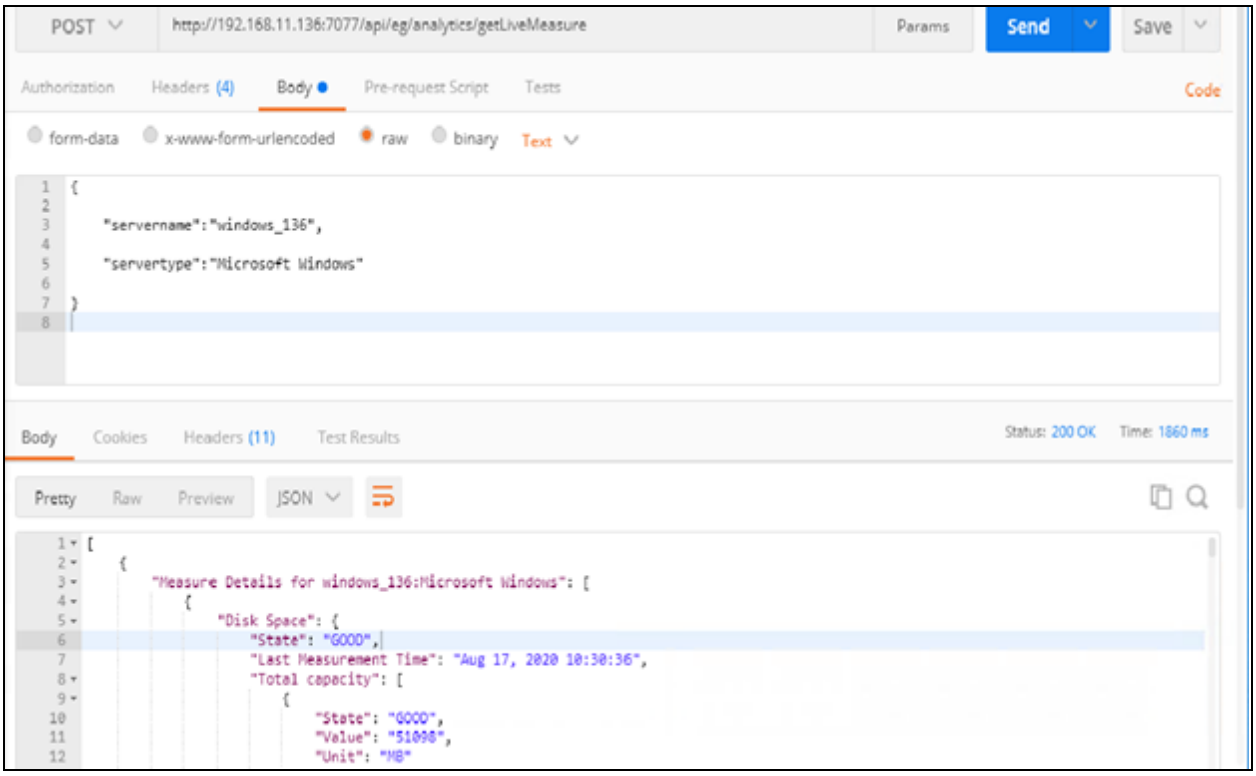

Figure 4.3: Example to retrieve current measures of a component using Postman REST Client

4.2.1 Retrieving Live Measures of a Component using cURL

To retrieve the measures of a component during the current measurement period through the REST API using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getLiveMeasure" -H "user:<eG username or domain/eG username>" -
H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -d"
{\"servertype\":\Microsoft Windows\",\servername\":\"name of the component\"}" -
H"Content-Type:application/json" -s
```

Figure 4.4 shows an example of retrieving the current measures of a component using cURL.

Figure 4.4: Retrieving current measures of a component using cURL

4.3 Retrieving Historical Data of a Measure

To figure out whether the target environment is functioning without any glitches, more often than not, administrators tend to monitor the performance of certain key measures over a period of time. Using the eG REST API, administrators are befitted in monitoring the performance of the measures over a period of time without logging into the eG console. The table below specifies the parameters that should be used to retrieve the historical data of the measures.

URL: http://192.168.8.206:7077/api/eg/analytics/getHistoricalData

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>	{ "timeline":"1 hour", "server":"Microsoft Windows:win112:NULL", "test":"Network",

Parameters	Key values	Example

	user: eG username or domain/eG username
	pwd: Base64 encoded password
		"measure":"packet loss"
		}
		Example for retrieving historical data of a measure pertaining to an Oracle Database Server:
		{
		"timeline":"1 day",
		"server":"Oracle Database Server:Oradb:1521:egora",
		"test":"Oracle Sessions",
		"measure":"Active Sessions"
		}

Body	**Default:**
	{
	"timeline":"Timeline for retrieving the measure data (in hours/days/weeks)",
	"server":"Component Type:Component name:Port/Null",
	"test":"Test name",
	"measure":"Measure name"
	}
	If current measures of the Oracle Database server is to be retrieved, then the Key values should be specified as follows:
	{
	"timeline":"Timeline for retrieving the measure data (in hours/days/weeks)",
	"server":"Component Type:Component name:Port:SID",
	"test":"Test name",
	"measure":"Measure name"	

Parameters	Key values	Example
	}	

Success Response

Type	Code	Example Response

JSON	200	{
		"NetworkTest": [
		{
		"Date": "29/09/2020 05:33:13",
		"Value": "0.0"
		},
		{
		"Date": "29/09/2020 05:38:40",
		"Value": "0.0"
		},
		.
		.
		.
		]
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

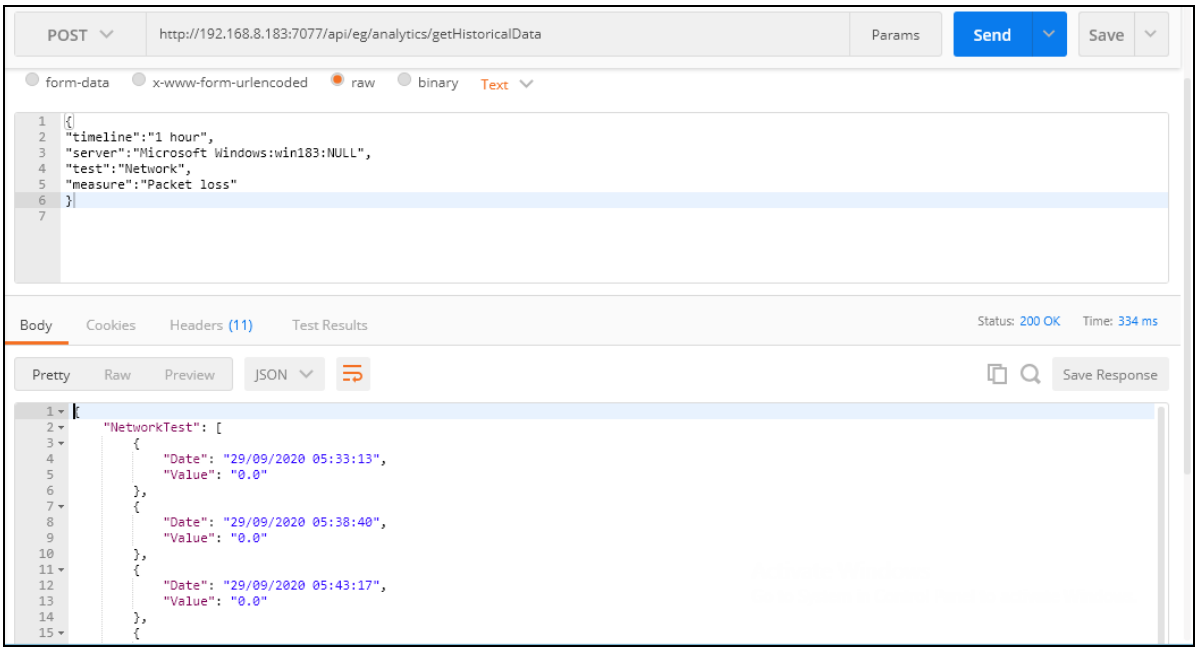

Figure 4.5: Retrieving historical data of a measure using Postman REST Client

4.3.1 Retrieving Historical Data of a Measure using cURL

To retrieve the historical data of a measure using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getHistoricalData" -H "user:<eG username or domain/eG
username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -
H"Content-Type:application/json" --data-raw "{\"timeline\":\"Timeline for retrieving the
measure data (in hours/days/weeks)\",\"server\":\"Component Type:Component
name:Port\",\"test\":\"Test name\",\"measure\":\"Measure name\"}"
```

Figure 4.6 shows an example of retrieving the current measures of a component using cURL.

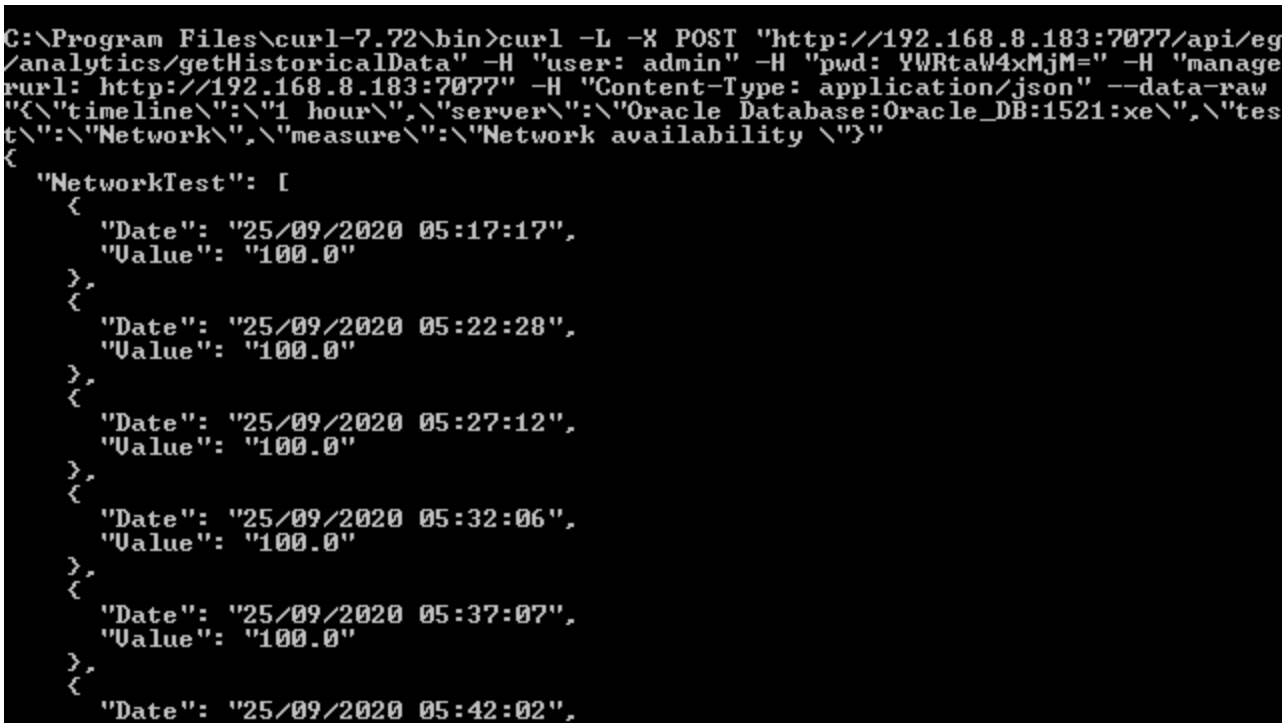

Figure 4.6: Retrieving historical data of a measure using cURL

4.4 Retrieving Detailed Diagnosis of a Measure

More often than not, administrators may want additional information on a key performance measure. Such additional information is provided as part of detailed diagnosis by eG Enterprise. Using eG REST API, administrators may be able to retrieve the detailed diagnosis of a measure without logging into the eG console. The table below specifies the URL and the parameters that should be used to retrieve the detailed diagnosis of a measure.

URL: http://192.168.8.206:7077/api/eg/analytics/getDiagnosisData

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Header	managerurl: Base URL of the eG Manager i.e., http://<IP address of the	{ "timeline":"1 hour",

Parameters	Key values	Example
	eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	"server":"Microsoft Windows:win112:NULL", "test":"Network", "measure":"packet loss", "descriptor":"" }

Body	**Default:**
	{
	"timeline"="Timeline for retrieving the measure data (in hours/days/weeks)",
	"server"="Component Type:Component name:Port/Null",
	"test"="Test name",
	"measure"="Measure name",
	"descriptor"="Descriptor name"
	}
	For an Oracle Database server, the Key values should be specified as follows:
	{
	"timeline"="Timeline for retrieving the measure data (in hours/days/weeks)",
	"server"="Component Type:Component name:Port:SID",
	test="Test name",
		Example to retrieve Detailed Diagnosis for a descriptor of a measure:
		{
		"timeline":"1 hour",
		"server":"Windows_server:win112:NULL",
		"test":"Disk Activity",
		"descriptor":"Disk0 C:",
		"measure":"Disk busy"
		}

Parameters	Key values	Example
	measure="Measure name" }	

Note:

The "descriptor" key value is mandatory for non-descriptor based tests too. In such case, the key value should be blank as shown in the example.

Success Response

Type	Code	Content
JSON	200	[{ "PROCESS ID": "4196", "APPLICATION NAME": "Symantec Service Framework", "PROCESSNAME": "C:\\Program Files (x86)\\Symantec\\Symantec Endpoint Protection\\14.2.5587.2100.105\\Bin\\ccSvcHst.exe /s Symantec Endpoint Protection /m C:\\Program Files (x86)\\Symantec\\Symantec Endpoint Protection\\14.2.5587.2100.105\\Bin\\sms.dll /prefetch:1", "IO RATE(KB/SEC)": "59.44", "IO READ RATE(KB/SEC)": "59.44", "IO READ OPS RATE(OPS/SEC)": "2.99", "IO WRITE RATE(KB/SEC)": "0", "IO WRITE OPS RATE(OPS/SEC)": "0", "FILE NAME": "-", "FILE IO READ RATE(KB/SEC)": "-", "FILE IO WRITE RATE(KB/SEC)": "-", "TOTAL FILE IO RATE(KB/SEC)": "-", "RESPONSE TIME(SECS)": "-"

Type	Code	Content
		}, . . . }]

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

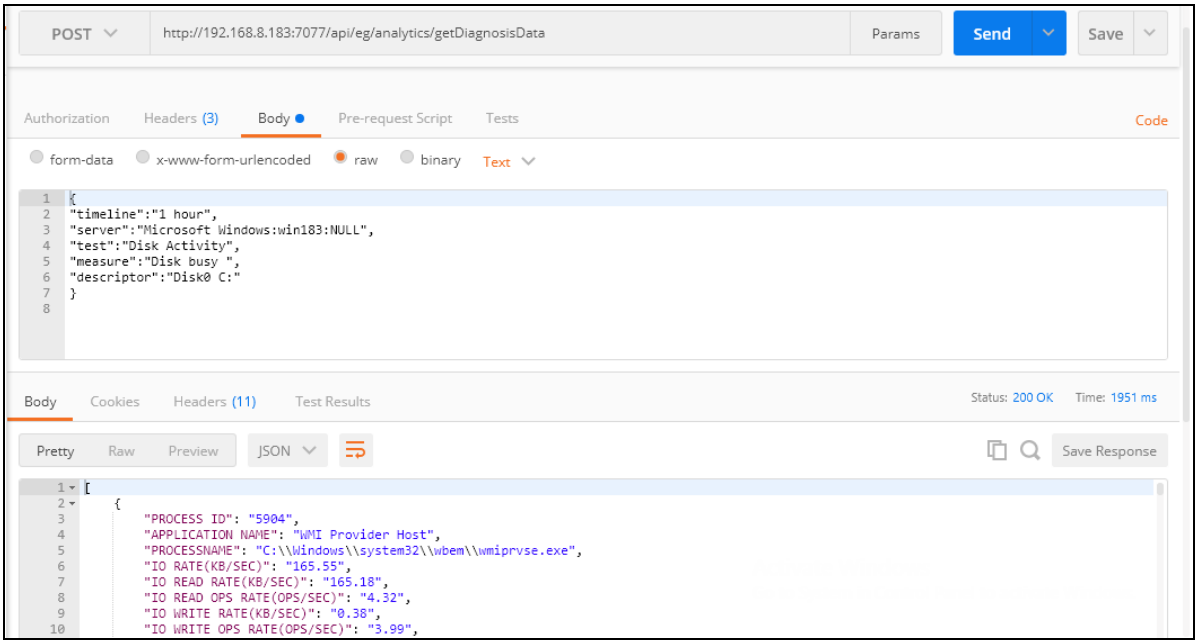

Figure 4.7: Retrieving detailed diagnosis of a measure using Postman REST Client

4.4.1 Retrieving Detailed diagnosis of a Measure using cURL

To retrieve the detailed diagnosis of a measure using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getDiagnosisData" -H "user:<eG username or domain/eG username>"
-H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -H"Content-
Type:application/json" --data-raw "{\"timeline\":\"Timeline for retrieving the detailed
diagnosis data (in hours/days/weeks)\",\"server\":\"Component Type:Component
name:Port\",\"test\":\"Test name\",\"measure\":\"Measure
name\",\"descriptor\":\"Descriptor name\"}"
```

Figure 4.8 shows an example of retrieving the detailed diagnosis of a measure using cURL.

Figure 4.8: Retrieving Detailed diagnosis of a measure using cURL

4.5 Retrieving Top-N Analysis Data

To identify the best/worst players in a particular performance area, administrators need to rank components/descriptors for every metric collected by the eG Enterprise. For such ranking, administrators need to figure out the Top-N Analysis data offered by eG Enterprise. Using the eG REST API, administrators can figure out the Top-N data of the components/descriptors of a measure reported by eG Enterprise without logging into the eG console. The table below specifies the parameters that should be used to retrieve the health of the infrastructure.

URL: http://192.168.8.206:7077/api/eg/analytics/getTopNData

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"timeline":"1 hour",
		"server":"win112:NULL:Microsoft Windows",
		"test":"Disk Activity",
		"descriptor":"Disk0 C:",
		"measure":"Disk busy"
		}

Body	**Default:**
	{
	"timeline":"Timeline for retrieving the measure data (in hours/days/weeks)",
	"server":"Component Type:Component name:Port/Null",
	"test":"Test name",
	"measure":"Measure name",
	"descriptor":"Descriptor name"
	}	

Success Response

Type	Code	Content

JSON	200	[
		{
		"Name": "win183 {Disk0 C: D:}",
		"Value": "7.5"
		}
		]

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

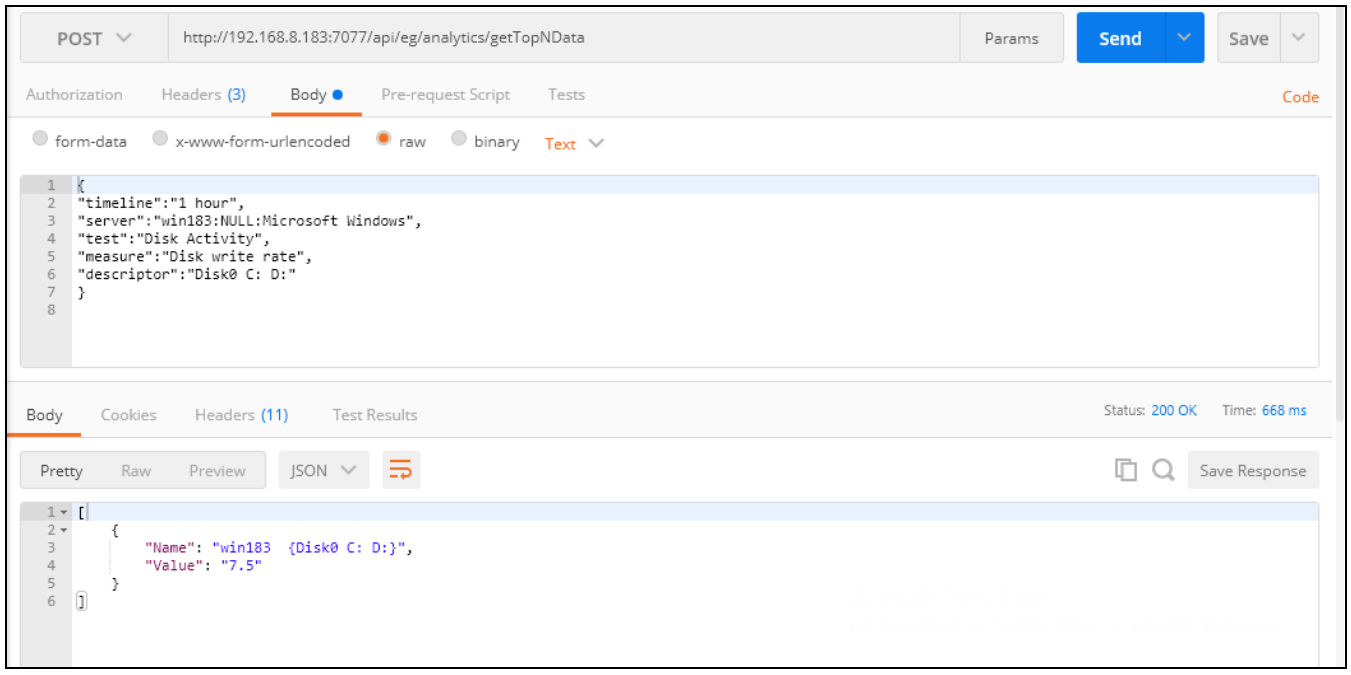

Figure 4.9: Retrieving Top-N Analysis Data using Postman REST Client

4.5.1 Retrieving Top-N Analysis Data using cURL

To retrieve the Top-N Data of components/descriptors using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager IP:Port>/api/eg/analytics/getTopNData" -H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -H"Content-Type:application/json" --data-raw "{\"timeline\":\"Timeline for retrieving the Top-N data (in hours/days/weeks)\",\"server\":\"Component Type:Component name:Port\",\"test\":\"Test name\",\"measure\":\"Measure name\",\"descriptor\":\"Descriptor name\"}"
```

Figure 4.10 shows an example of retrieving the Top-N Analysis Data using cURL.

Figure 4.10: Retrieving Top-N Analysis Data using cURL

4.6 Retrieving Test Data

Using the eG REST API, administrators can retrieve the measurement data collected upon execution of tests across all relevant component types. The table below specifies the parameters that should be used to retrieve the measures of the tests.

URL: http://192.168.8.206:7077/api/eg/analytics/getTestData

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"test":"TCP Port Status",
		"host":"sql",
		"port":"1433",
		"lastmeasure":"true",
		"startDate":"2020-01-29 18:00:26",
		"endDate":"2020-01-29 18:15:03"
		}

Body	**Default:**
	{
	"test":"Test name",
	"host":"Host name",
	"port":"Port",
	"info":"info"
	}	

	Optional: { "lastmeasure":"true/false", "startDate":"start date", "endDate":"End date", "measures":"comma-separated list of measures", "msmthost":"Measurement Host", "type":"dd", "segment":"Segment name", "service":"Service name", "searchhost":"Search Host", "searchinfo":"Search info", "groupby":"measure", "orderby":"Ascending/Descending",	

Parameters	Key values	Example
	"dateformat":"Date formant", }	

Success Response

Type	Code	Content
JSON	200	["TRGT_HOST PORT_NO SITE_NAME INFO MSMT_HOST MSMT_TIME AVAILABILITY AVAILABILITY_ST RESPONSETIME RESPONSETIME_ST ", "", "cvadddc7v1912 80 NULL +80 eGDP169 2020-09-17 15:34:58 100.0000 GOOD 0.0030 GOOD", "cvadddc7v1912 80 NULL +80 eGDP169 2020-09-17 15:39:48 100.0000 GOOD 0.0030 GOOD", "cvadddc7v1912 80 NULL +80 eGDP169 2020-09-17 15:44:41 100.0000 GOOD 0.0040 GOOD", "cvadddc7v1912 80 NULL +80 eGDP169 2020-09-17 15:49:39 100.0000 GOOD 0.0070 GOOD", . . .]

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

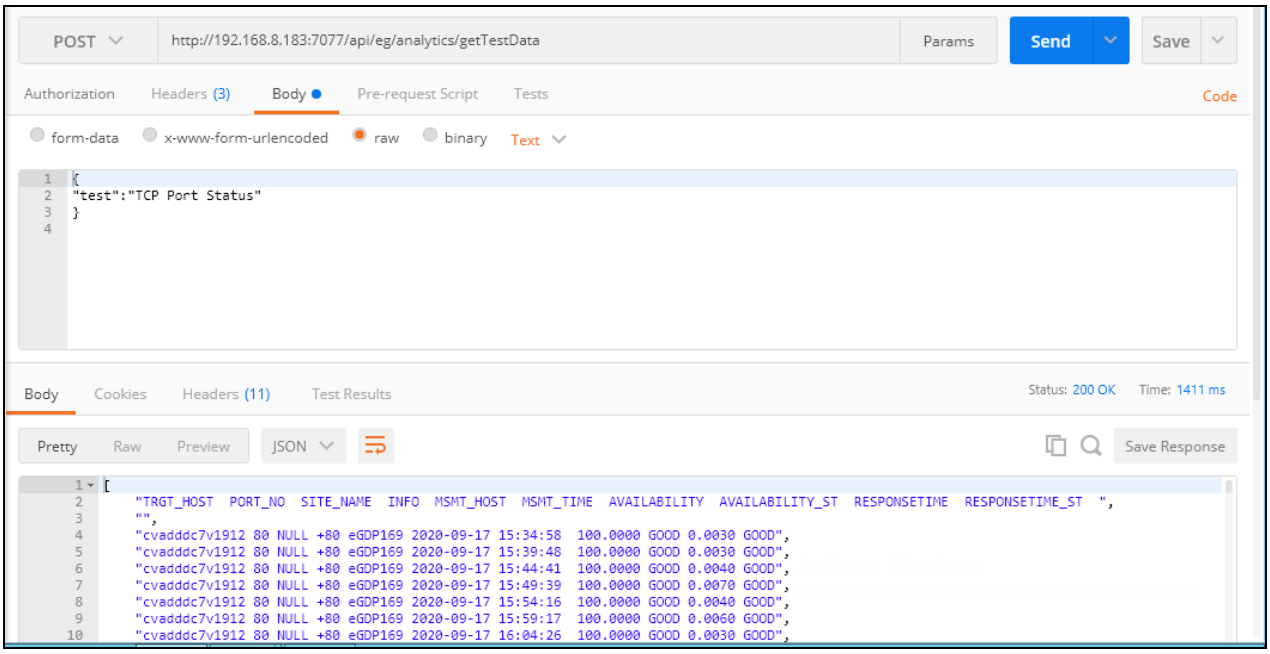

Figure 4.11: Retrieving measurement data of a test using Postman REST Client

4.6.1 Retrieving Test Data using cURL

To retrieve the measurement data collected upon execution of tests across all relevant component types using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager IP:Port>/api/eg/analytics/getTestData" -H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -H"Content-Type:application/json" --data-raw "{\"test\":\"test name\"}
```

Figure 4.12 shows an example cURL command for retrieving the measurement data that is reported by eG Enterprise by monitoring all the components in the target environment.

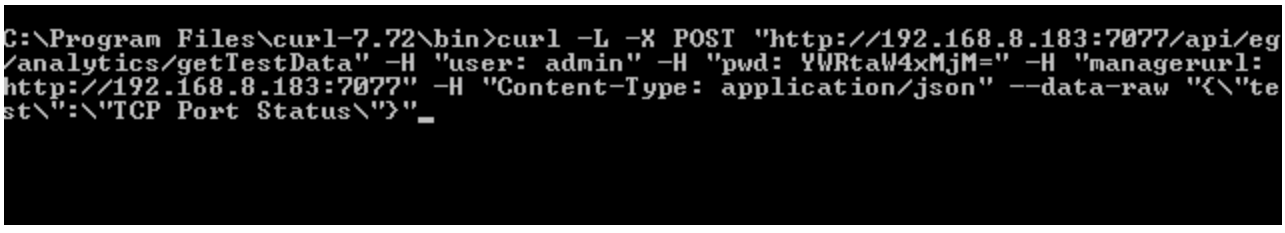

Figure 4.12: An example cURL command to retrieve the measurement data of the test

Figure 3 shows a sample output that retrieves the measurement data of a chosen test reported by eG Enterprise using cURL.

Figure 4.13: Sample output with the measurement data of a test across all monitored component types

4.7 Retrieving Trend Data

Using the eG REST API, administrators can retrieve the trend data of the tests across all relevant component types. The table below specifies the URL and the parameters that should be used to retrieve the measures of the tests.

URL: http://192.168.8.206:7077/api/eg/analytics/getTrendData

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	{ "test":"Oracle Latches", "host":"Oracle", "port":"1521", "info":"egurkha+redo allocation", "type":"Trend"

Parameters	Key values	Example

Body	**Default:**
	{
	"test":"Test name",
	"host":"Host name",
	"port":"Port",
	"info":"info"
	}
		}

	Optional:
	{
	"startDate":"start_date",
	"endDate":"end_date",
	"measure":"comma-separated list of measures",
	"msmthost":"Measurement Host",
	"type":"trend",
	"segment":"Segment Name",
	"service":"Service Name",
	"searchhost":"Search Host",
	"searchinfo":"Info",
	"groupby":"measure",
	"orderby":"Ascending/Descending",
	}	

Success Response

Type	Code	Content
JSON	200	[

Type	Code	Content
		"TRGT_HOST PORT_NO SITE_NAME INFO MSMT_HOST MSMT_TIME ", "", "eGDP169 NULL NULL +Disk0 C: D: eGDP169 2020-08-01 00:00:00 ", "HISELKVPMAS01 NULL NULL +dm-0 eGDP169 2020-08-01 00:00:00 ", "HISELKVPMAS01 NULL NULL +dm-1 eGDP169 2020-08-01 00:00:00 ", "HISELKVPMAS01 NULL NULL +sda eGDP169 2020-08-01 00:00:00 ", "java183 NULL NULL +Disk0 C: D: java183 2020-08-01 00:00:00 ", "win183 NULL NULL +Disk0 C: D: win183 2020-08-01 00:00:00 ", . . .]

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

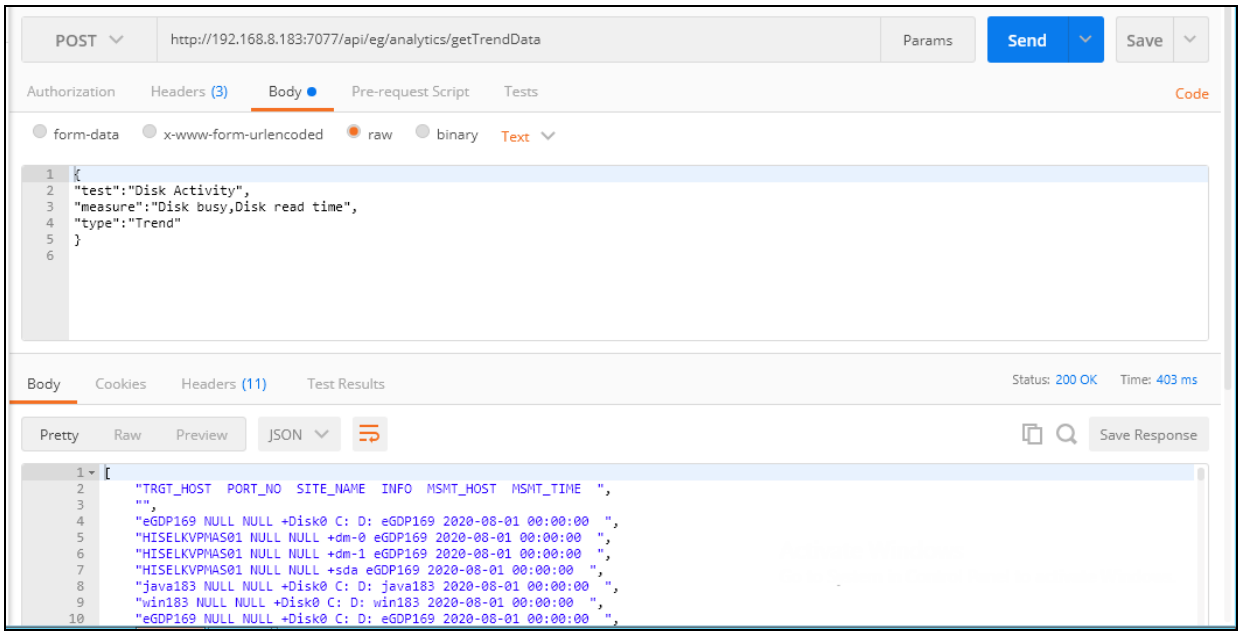

Figure 4.14: Retrieving trend data of a chosen measure using Postman REST Client

4.7.1 Retrieving Trend Data using cURL

To retrieve trend data of the tests using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager IP:Port>/api/eg/analytics/getTrendData" -H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -H"Content-Type:application/json" --data-raw "{\"test\":\"Test name\",\"measure\":\"comma-separated list of measures\",\"type\":\"Trend\"}"
```

Figure 4.15 shows an example to retrieve the trend data for the measures of a chosen test using cURL.

Figure 4.15: An example cURL command to retrieve the trend data for the measures

Figure 3 shows a sample output that retrieves the trend data for the chosen measures of a chosen test reported by eG Enterprise using cURL.


sk0 C: D: java183 2020-09-23 16:00:00 ","win183 NULL NULL +Disk0 C: D: win183 2
020-09-23 16:00:00 ","eGDP169 NULL NULL +Disk0 C: D: eGDP169 2020-09-23 17:00:0
0 ","HISELKVPMAS01 NULL NULL +dm-0 eGDP169 2020-09-23 17:00:00 ","HISELKVPMAS0
1 NULL NULL +dm-1 eGDP169 2020-09-23 17:00:00 ","HISELKVPMAS01 NULL NULL +sda e
GDP169 2020-09-23 17:00:00 ","java183 NULL NULL +Disk0 C: D: java183 2020-09-23
17:00:00 ","win183 NULL NULL +Disk0 C: D: win183 2020-09-23 17:00:00 ","eGDP1
69 NULL NULL +Disk0 C: D: eGDP169 2020-09-23 17:00:00 ","HISELKVPMAS01 NULL NUL
L +dm-0 eGDP169 2020-09-23 17:00:00 ","HISELKVPMAS01 NULL NULL +dm-1 eGDP169 20
20-09-23 17:00:00 ","HISELKVPMAS01 NULL NULL +sda eGDP169 2020-09-23 17:00:00
","java183 NULL NULL +Disk0 C: D: java183 2020-09-23 17:00:00 ","win183 NULL NU
LL +Disk0 C: D: win183 2020-09-23 17:00:00 ","eGDP169 NULL NULL +Disk0 C: D: eG
DP169 2020-09-23 18:00:00 ","HISELKVPMAS01 NULL NULL +dm-0 eGDP169 2020-09-23 1
8:00:00 ","HISELKVPMAS01 NULL NULL +dm-1 eGDP169 2020-09-23 18:00:00 ","HISELK
VPMAS01 NULL NULL +sda eGDP169 2020-09-23 18:00:00 ","java183 NULL NULL +Disk0
C: D: java183 2020-09-23 18:00:00 ","win183 NULL NULL +Disk0 C: D: win183 2020-
09-23 18:00:00 ","eGDP169 NULL NULL +Disk0 C: D: eGDP169 2020-09-23 18:00:00 "
,"HISELKVPMAS01 NULL NULL +dm-0 eGDP169 2020-09-23 18:00:00 ","HISELKVPMAS01 NU
LL NULL +dm-1 eGDP169 2020-09-23 18:00:00 ","HISELKVPMAS01 NULL NULL +sda eGDP1
69 2020-09-23 18:00:00 ","java183 NULL NULL +Disk0 C: D: java183 2020-09-23 18:
00:00 ","win183 NULL NULL +Disk0 C: D: win183 2020-09-23 18:00:00 ","eGDP169 N
ULL NULL +Disk0 C: D: eGDP169 2020-09-23 19:00:00 ","HISELKVPMAS01 NULL NULL +d
m-0 eGDP169 2020-09-23 19:00:00 ","HISELKVPMAS01 NULL NULL +dm-1 eGDP169 2020-0
9-23 19:00:00 ","HISELKVPMAS01 NULL NULL +sda eGDP169 2020-09-23 19:00:00 ","j
ava183 NULL NULL +Disk0 C: D: java183 2020-09-23 19:00:00 ","win183 NULL NULL +
Disk0 C: D: win183 2020-09-23 19:00:00 ","eGDP169 NULL NULL +Disk0 C: D: eGDP16
9 2020-09-23 19:00:00 ","HISELKVPMAS01 NULL NULL +dm-0 eGDP169 2020-09-23 19:00
:00 ","HISELKVPMAS01 NULL NULL +dm-1 eGDP169 2020-09-23 19:00:00 ","HISELKVPMA
S01 NULL NULL +sda eGDP169 2020-09-23 19:00:00 ","java183 NULL NULL +Disk0 C: D
: java183 2020-09-23 19:00:00 ","win183 NULL NULL +Disk0 C: D: win183 2020-09-2
3 19:00:00 ","eGDP169 NULL NULL +Disk0 C: D: eGDP169 2020-09-23 20:00:00 ","HI

Figure 4.16: Sample output with the trend data for the chosen measures of a chosen test

4.8 Retrieving Threshold Data

Using the eG REST API, administrators can retrieve the threshold configured for the measures of a chosen test. The table below specifies the URL and the parameters that should be used to retrieve the measures of the tests.

URL: http://192.168.8.206:7077/api/eg/analytics/getThresholdData

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	{ "test":"Oracle Latches", "host":"Oracle", "port":"1521", "info":"egurkha+redo allocation",

Parameters	Key values	Example

Body	**Default:**
	{
	"test":"Test name",
	"host":"Host name",
	"port":"Port",
	"info":"info"
	}
		"type":"Threshold"
		}

	Optional: "type":"Threshold", "measure":"comma-separated list of measures", "searchhost":"Search Host", "searchinfo":"Search info", "groupby":"TRGT_HOST", "orderby":"Ascending/Descending", }	

Success Response

Type	Code	Content
JSON	200	["TRGT_HOST PORT_NO SITE_NAME INFO MSMT_HOST MSMT_TIME_START MSMT_TIME_END TOTAL_ CAPACITY_MIN TOTAL_CAPACITY_MAX USED_SPACE_ MIN USED_SPACE_MAX FREE_SPACE_MIN FREE_SPACE_ MAX PERCENT_USAGE_MIN PERCENT_USAGE_MAX DRIVE_AVAIL_MIN DRIVE_AVAIL_MAX ", "", "win183 NULL NULL +D win183 2020-09-12 19:30:00

Type	Code	Content
		2020-09-12 20:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1", "win183 NULL NULL +D win183 2020-09-12 20:30:00 2020-09-12 21:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1", "win183 NULL NULL +D win183 2020-09-12 21:30:00 2020-09-12 22:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1", "win183 NULL NULL +D win183 2020-09-12 22:30:00 2020-09-12 23:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1", "win183 NULL NULL +D win183 2020-09-12 23:30:00 2020-09-13 00:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1", . . .]

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

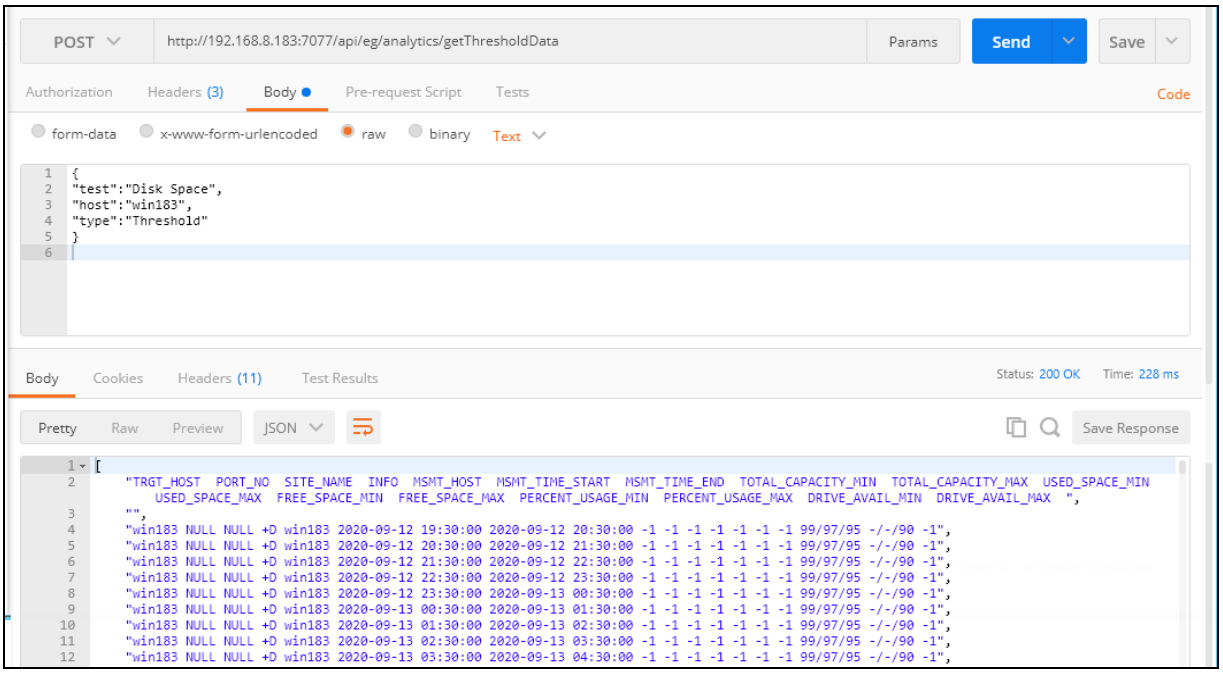

Figure 4.17: Retrieving Threshold data configured for the measures using Postman REST Client

4.8.1 Retrieving Threshold Data using cURL

To retrieve the threshold configured for the measures of a chosen test using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getThresholdData" -H "user:<eG username or domain/eG username>"
-H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -H"Content-
Type:application/json" --data-raw "{\"test\":\"Test name\",\"host\":\"Host
name\",\"type\":\"Threshold\"}"
```

Figure 4.18 shows an example to retrieve the threshold configured for the measures of a chosen test using cURL.

Figure 4.18: An example cURL command to retrieve the threshold configured for the measures

Figure 3 shows a sample output that retrieves the threshold configured for the measures of a chosen test reported by eG Enterprise using cURL.


"win183 NULL NULL +D win183 2020-09-18 20:30:00 2020-09-18 21:30:00 -1 -1 -1 -1
-1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NULL +D win183 2020-09-18 21:30:00 202
0-09-18 22:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NULL +D w
in183 2020-09-18 22:30:00 2020-09-18 23:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/
90 -1","win183 NULL NULL +D win183 2020-09-18 23:30:00 2020-09-19 00:30:00 -1 -1
-1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NULL +D win183 2020-09-19 00:30
:00 2020-09-19 01:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NU
LL +D win183 2020-09-19 01:30:00 2020-09-19 02:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/
95 -/-/90 -1","win183 NULL NULL +D win183 2020-09-19 02:30:00 2020-09-19 03:30:0
0 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NULL +D win183 2020-09-1
9 03:30:00 2020-09-19 04:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183
NULL NULL +D win183 2020-09-19 04:30:00 2020-09-19 05:30:00 -1 -1 -1 -1 -1 -1 -1
99/97/95 -/-/90 -1","win183 NULL NULL +D win183 2020-09-19 05:30:00 2020-09-19
06:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NULL +D win183 20
20-09-19 06:30:00 2020-09-19 07:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","
win183 NULL NULL +D win183 2020-09-19 07:30:00 2020-09-19 08:30:00 -1 -1 -1 -1 -
1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NULL +D win183 2020-09-19 08:30:00 2020
-09-19 09:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NULL +D wi
n183 2020-09-19 09:30:00 2020-09-19 10:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/9
0 -1","win183 NULL NULL +D win183 2020-09-19 10:30:00 2020-09-19 11:30:00 -1 -1
-1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NULL +D win183 2020-09-19 11:30:
00 2020-09-19 12:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NUL
L +D win183 2020-09-19 12:30:00 2020-09-19 13:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/9
5 -/-/90 -1","win183 NULL NULL +D win183 2020-09-19 13:30:00 2020-09-19 14:30:00
-1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NULL +D win183 2020-09-19
14:30:00 2020-09-19 15:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 N
ULL NULL +D win183 2020-09-19 15:30:00 2020-09-19 16:30:00 -1 -1 -1 -1 -1 -1 -1
99/97/95 -/-/90 -1","win183 NULL NULL +D win183 2020-09-19 16:30:00 2020-09-19 1
7:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","win183 NULL NULL +D win183 202
0-09-19 17:30:00 2020-09-19 18:30:00 -1 -1 -1 -1 -1 -1 -1 99/97/95 -/-/90 -1","w

Figure 4.19: Sample output with the threshold data configured for the measures of a chosen test

4.9 Retrieving Infrastructure Health

Using the eG REST API, administrators can figure out the health of the Zone/Service/Segment/Component Type managed in the eG manager without logging into the eG console. The table below specifies the URL and the parameters that should be used to retrieve the health of the infrastructure.

URL: http://192.168.8.206:7077/api/eg/analytics/getInfraHealth

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	Example to retrieve the health of a zone: { "type":"Zone", "name":"east zone"

Parameters	Key values	Example
		}

Body	Default:
	{
	"type":"Zone/Service/Segment/Component Type",
	"name":"Name of Zone/Service/Segment/Component Type"
	}
		Example to retrieve the health of the components of a chosen Component Type:
		{
		"type":"Component Type",
		"name":"Microsoft Windows"
		}

Success Response

Type	Code	Content
JSON	200	{ "root": [{ "Component": "Virtual_center:vmware_vcenter:NULL ", "State": "HIGH" }, { "Component": "VmEsx_i_server:esx51-15:NULL ", "State": "HIGH" }, { "Component": "VmEsx_i_server:vmware_vsphere_esx:NULL ", "State": "INTERMEDIATE" }, {

Type	Code	Content

		"Component": "Xen_desktop_server:citrix_xenserver_vdi:NULL ",
		"State": "LOW"
		},
		{
		"Component": "Xen_virtual_server:citrix_xenserver_vdi:NULL ",
		"State": "LOW"
		},
		{
		"Component": "commzComp1_ex:commzgate183:NULL ",
		"State": "UNKNOWN"
		}
		],
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

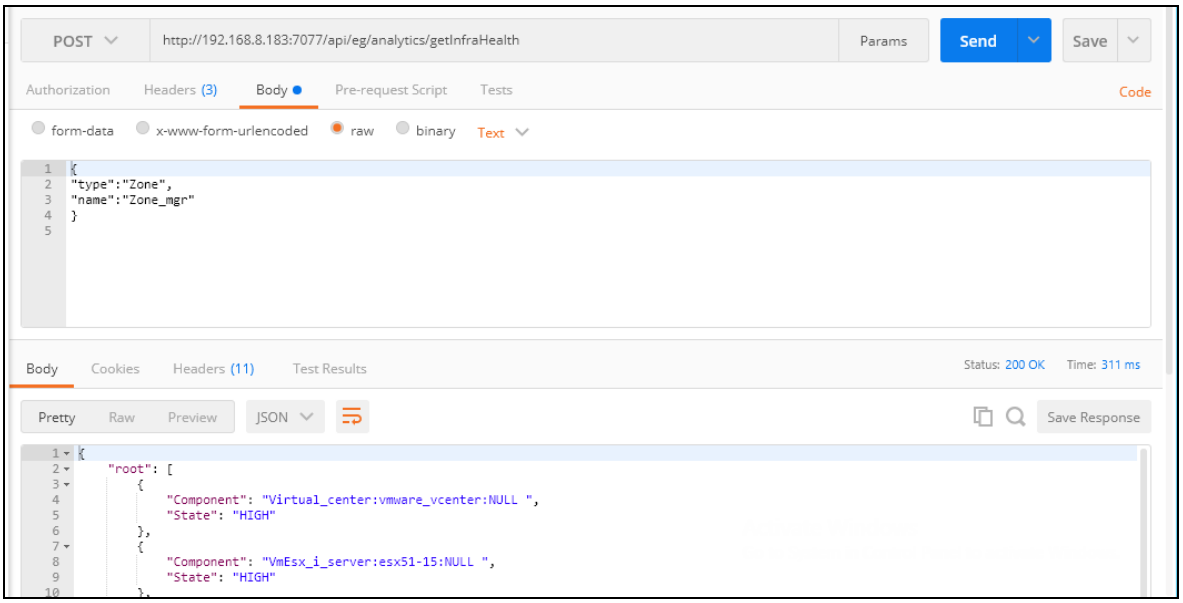

Figure 4.20: Retrieving the health of the components in a zone using Postman REST Client

4.9.1 Retrieving Infrastructure Health using cURL

To retrieve the health of the Zone/Service/Segment/Component Type managed in the eG manager using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getInfraHealth" -H "user:<eG username or domain/eG username>" -
H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -H"Content-
Type:application/json" --data-raw "{\"type\":\"Zone/Service/Segment/Component
Type\",\"name\":\"Name of Zone/Service/Segment/Component Type\"}
```

Figure 4.21 shows an example of retrieving the health of the components within a zone in the target environment using cURL.

Figure 4.21: Retrieving the health of the components in a zone using cURL

4.10 Retrieving Problem Distribution of Components

Use the URL specified below to retrieve the priority based problem distribution of the chosen components in the target environment using the eG REST API.

URL: http://192.168.8.206:7077/api/eg/analytics/getServerListProblemDistribution

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded
		{
		serverlist":"Microsoft Windows:win112:NULL,Oracle Database Server:Oradb123:1521:egora"
		}

Parameters	Key values	Example
	password	

Body	**Default:**
	{
	"serverlist":"comma-separated
	list of ComponentType:Component:Port/Null:SID",
	}	

Success Response

Type	Code	Content

JSON	200	{
		"eG_Manager:eGDP169:7077": [
		{
		"date": "16/09",
		"CRITICAL": "0",
		"MAJOR": "0",
		"MINOR": "0"
		},
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

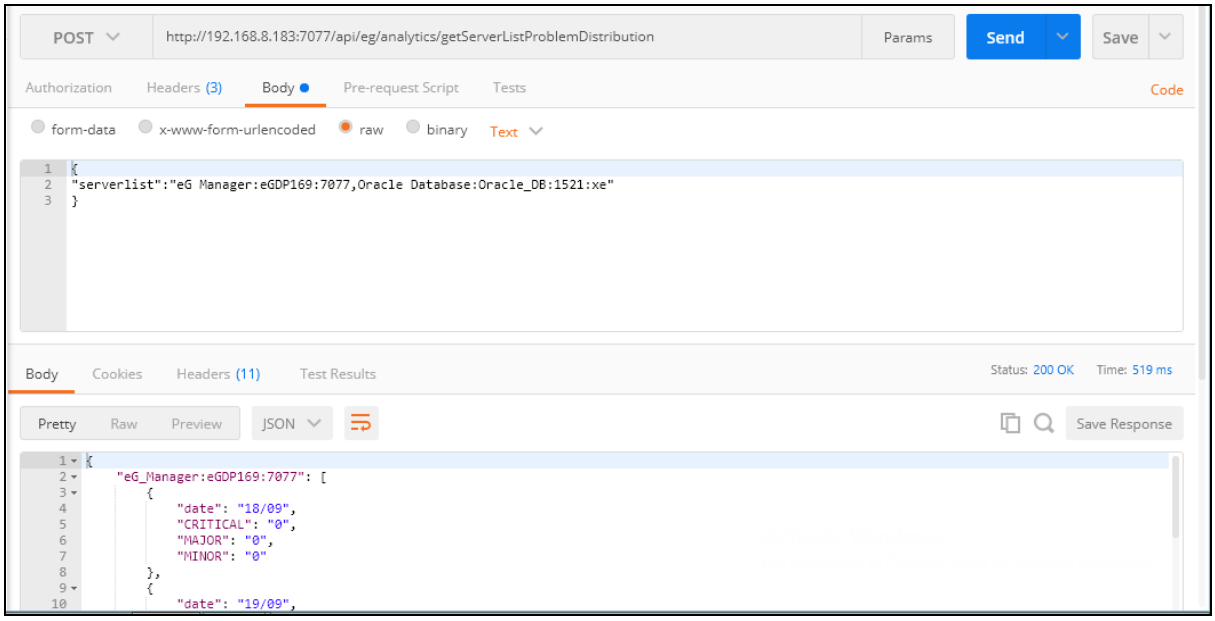

Figure 4.22: Retrieving the priority based problem distribution of a chosen component using Postman REST Client

4.10.1 Retrieving Problem Distribution of Components using cURL

To retrieve the priority based problem distribution of the chosen components using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getServerListProblemDistribution" -H "user:<eG username or
domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager
IP:Port>" -H"Content-Type:application/json" --data-raw "{\"serverlist\":\"comma-separated
list of ComponentType:Component:Port/Null:SID\"}
```

Figure 4.23 shows an example of retrieving the priority based problem distribution of the chosen component using cURL.

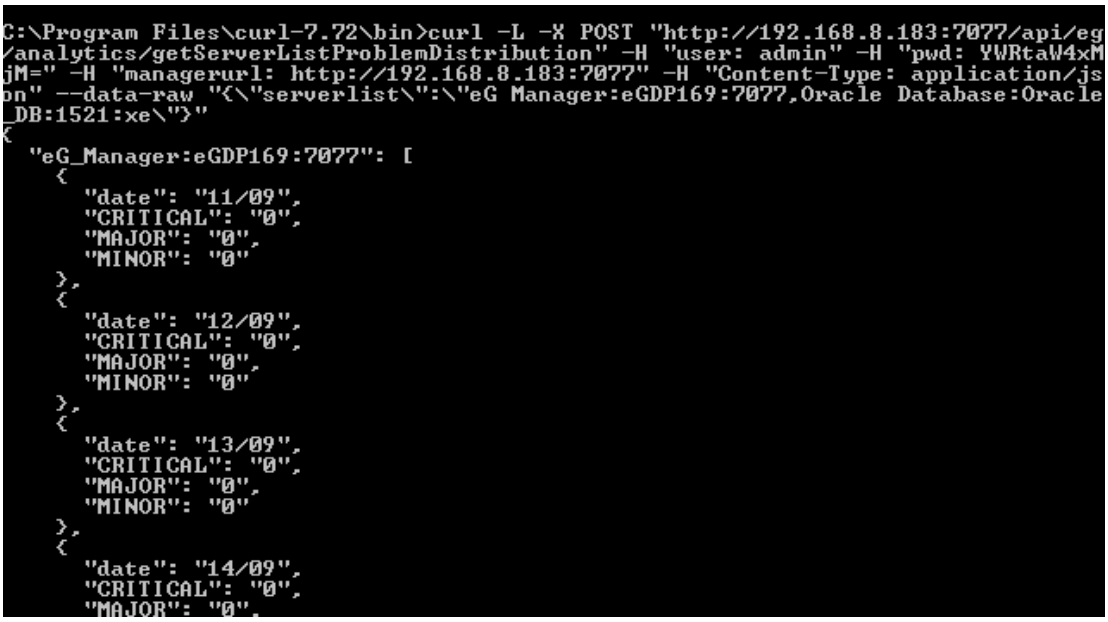

Figure 4.23: Retrieving the priority based problem distribution of a chosen component using cURL

4.11 Retrieving Problem Distribution of the Target Environment

Using the eG REST API, administrators can retrieve the alarm count based on severity for all component types, components, layers and tests specific to the target environment.

4.11.1 Retrieving Problem Distribution for all Component Types

URL: http://192.168.8.206:7077/api/eg/analytics/getProblemDistribution/servertype

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl
	user
	pwd
		{
		"timeline":"1 hour"
		}

Body	**Default:** {	

Parameters	Key values	Example
	"timeline":"Timeline for retrieving the alarms (in hours/days/weeks)" }	

Success Response

Type	Code	Content

JSON	200	{
		"Problem Distribution": [
		{
		"Server Type": "Java Application",
		"CRITICAL": "0",
		"MAJOR": "47",
		"MINOR": "7"
		},
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

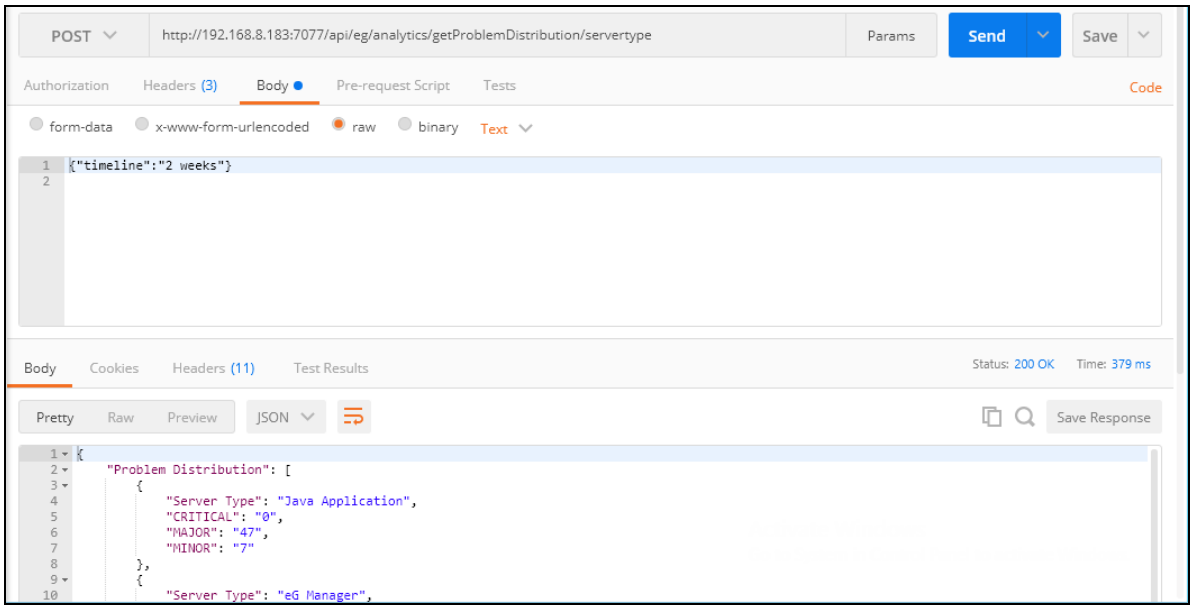

Figure 4.24: Retrieving the alarm count based on severity for all component types using Postman REST Client

4.11.2 Retrieving Problem Distribution for all Component Types using cURL

To retrieve the alarm count based on severity for all component types managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getProblemDistribution/servertype" -H "user:<eG username or
domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager
IP:Port>" -H"Content-Type:application/json" --data-raw "{\"timeline\":\"Timeline for
retrieving the alarms (in hours/days/weeks)\"}"
```

Figure 4.25 shows an example to retrieve the alarm count based on severity for all component types managed in the target environment using cURL.

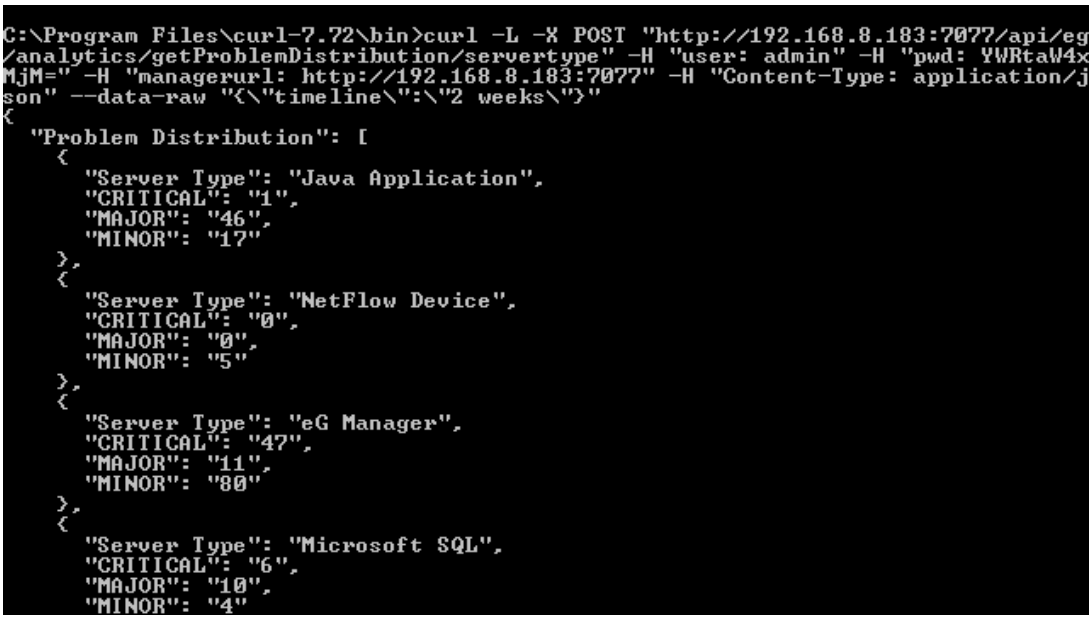

Figure 4.25: Retrieving the alarm count based on severity for all component types using cURL

4.11.3 Retrieving Problem Distribution for all Components

URL: http://192.168.8.206:7077/api/eg/analytics/getProblemDistribution/servername

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Header	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"timeline":"1 hour"
		}

Body	Default:	

Parameters	Key values	Example

	{
	"timeline":"Timeline for retrieving the alarms (in hours/days/weeks)"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Problem Distribution": [
		{
		"Server Name": "esx51-15",
		"CRITICAL": "11",
		"MAJOR": "26",
		"MINOR": "148"
		},
		{
		"Server Name": "win183",
		"CRITICAL": "5",
		"MAJOR": "6",
		"MINOR": "1"
		},
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

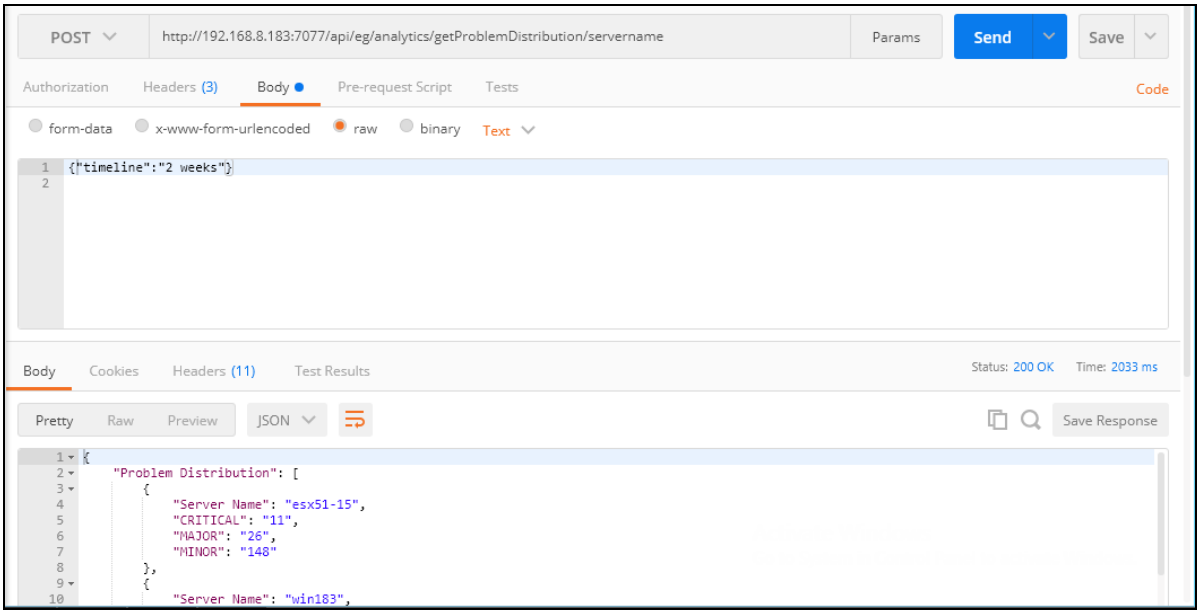

Figure 4.26: Retrieving the alarm count based on severity for all components using Postman REST Client

4.11.4 Retrieving Problem Distribution for all Components using cURL

To retrieve the alarm count based on severity for all components managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getProblemDistribution/servername" -H "user:<eG username or
domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager
IP:Port>" -H"Content-Type:application/json" --data-raw "{\"timeline\":\"Timeline for
retrieving the alarms (in hours/days/weeks)\"}"
```

Figure 4.25 shows an example to retrieve the alarm count based on severity for all components managed in the target environment using cURL.

Figure 4.27: Retrieving the alarm count based on severity for all components using cURL

4.11.5 Retrieving Problem Distribution of the Layers of a Component Type

URL: http://192.168.8.206:7077/api/eg/analytics/getProblemDistribution/layer

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"timeline":"1 hour",
		"servertype":"Microsoft Windows"
		}

Body	**Default:** {	

Parameters	Key values	Example
	"timeline":"Timeline for retrieving the alarms (in hours/days/weeks)", "servertype":"Component Type" }	

Success Response

Type	Code	Content

JSON	200	{
		"Problem Distribution": [
		{
		"Layer Name": "Application Processes",
		"CRITICAL": "4",
		"MAJOR": "3",
		"MINOR": "0"
		},
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

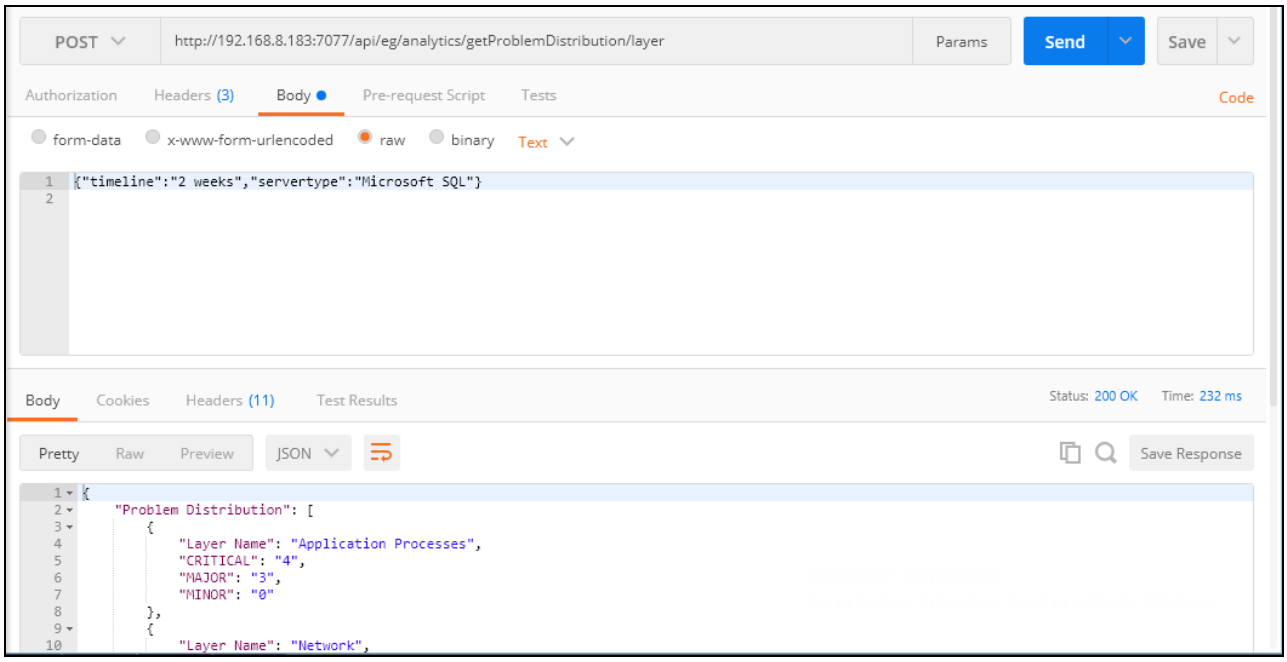

Figure 4.28: Retrieving the alarm count based on severity for all layers of a Component Type using Postman REST Client

4.11.6 Retrieving Problem Distribution of the Layers of a Component Type using cURL

To retrieve the alarm count based on severity corresponding to all layers of a chosen Component Type managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getProblemDistribution/layer" -H "user:<eG username or
domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager
IP:Port>" -H"Content-Type:application/json" --data-raw "{\"timeline\":\"Timeline for
retrieving the alarms (in hours/days/weeks)\",\"servertype\":\"Component Type\"}"}"
```

Figure 4.25 shows an example to retrieve the alarm count based on severity corresponding to all layers of a chosen Component Type managed in the target environment using cURL.

Figure 4.29: Retrieving the alarm count based on severity for all layers of a Component Type using cURL

4.11.7 Retrieving Problem Distribution of the Tests of a Component Type

URL: http://192.168.8.206:7077/api/eg/analytics/getProblemDistribution/test

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"timeline":"1 hour",
		"servertype":"Microsoft Windows"
		}

Body	**Default:** {	

Parameters	Key values	Example
	"timeline":"Timeline for retrieving the alarms (in hours/days/weeks)", "servertype":"Component Type" }	

Success Response

Type	Code	Content

JSON	200	{
		"Problem Distribution": [
		{
		"Test Name": "Windows Services",
		"CRITICAL": "5",
		"MAJOR": "0",
		"MINOR": "0"
		},
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

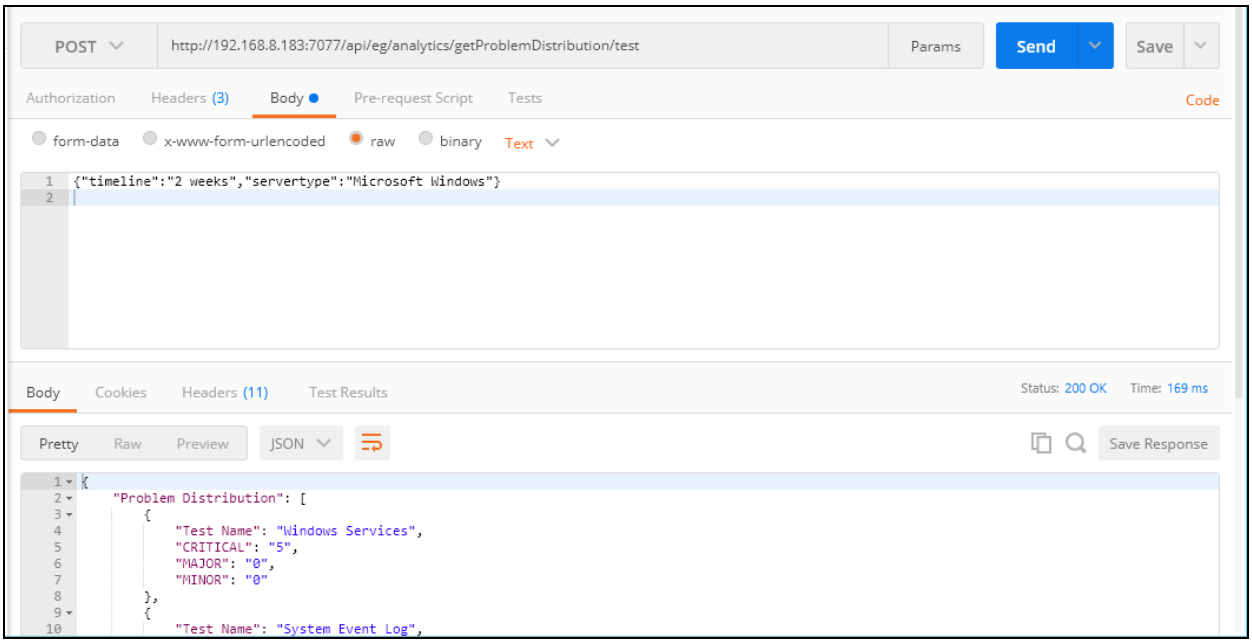

Figure 4.30: Retrieving the alarm count based on severity for all tests of a Component Type using Postman REST Client

4.11.8 Retrieving Problem Distribution of the Tests of a Component Type using cURL

To retrieve the alarm count based on severity for the tests of a chosen Component Type managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager IP:Port>/api/eg/analytics/getProblemDistribution/test" -H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -H"Content-Type:application/json" --data-raw "{\"timeline\":\"Timeline for retrieving the alarms (in hours/days/weeks)\",\"servertype\":\"Component Type\"}"}"
```

Figure 4.25 shows an example to retrieve the alarm count based on severity for all tests of a chosen Component Type managed in the target environment using cURL.

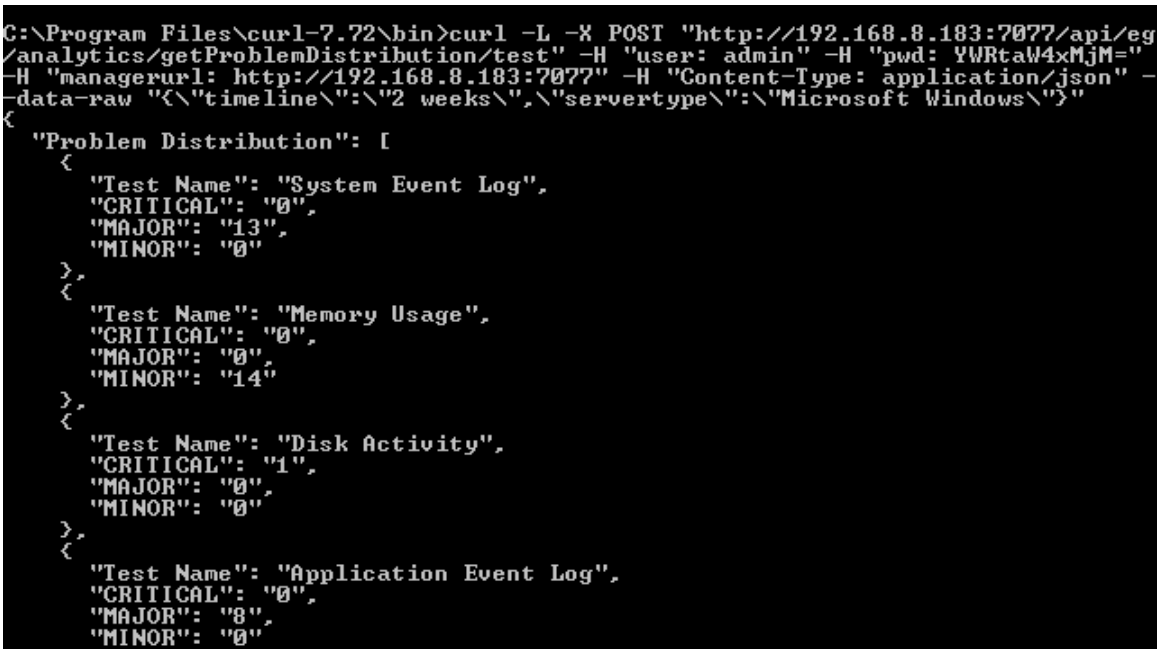

Figure 4.31: Retrieving the alarm count based on severity for all tests of a Component Type using cURL

4.12 Retrieving the Count of Events from Alarm History

Using the eG REST API, administrators can retrieve the count of events from Alarm History for all component types, components, layers and tests specific to the target environment.

4.12.1 Retrieving the Count of Events from Alarm History for all Component Types

URL: http://192.168.8.206:7077/api/eg/analytics/getEventCount/servertype

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>	{
		"timeline":"1 hour"
		}

Parameters	Key values	Example
	user: eG username or domain/eG username pwd: Base64 encoded password	

Body	**Default:**
	{
	"timeline":"Timeline for retrieving the count of events (in hours/days/weeks)"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Event Count": [
		{
		"Server Type": "Java Application",
		"Event Count": "54"
		},
		{
		"Server Type": "eG Manager",
		"Event Count": "137"
		},
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

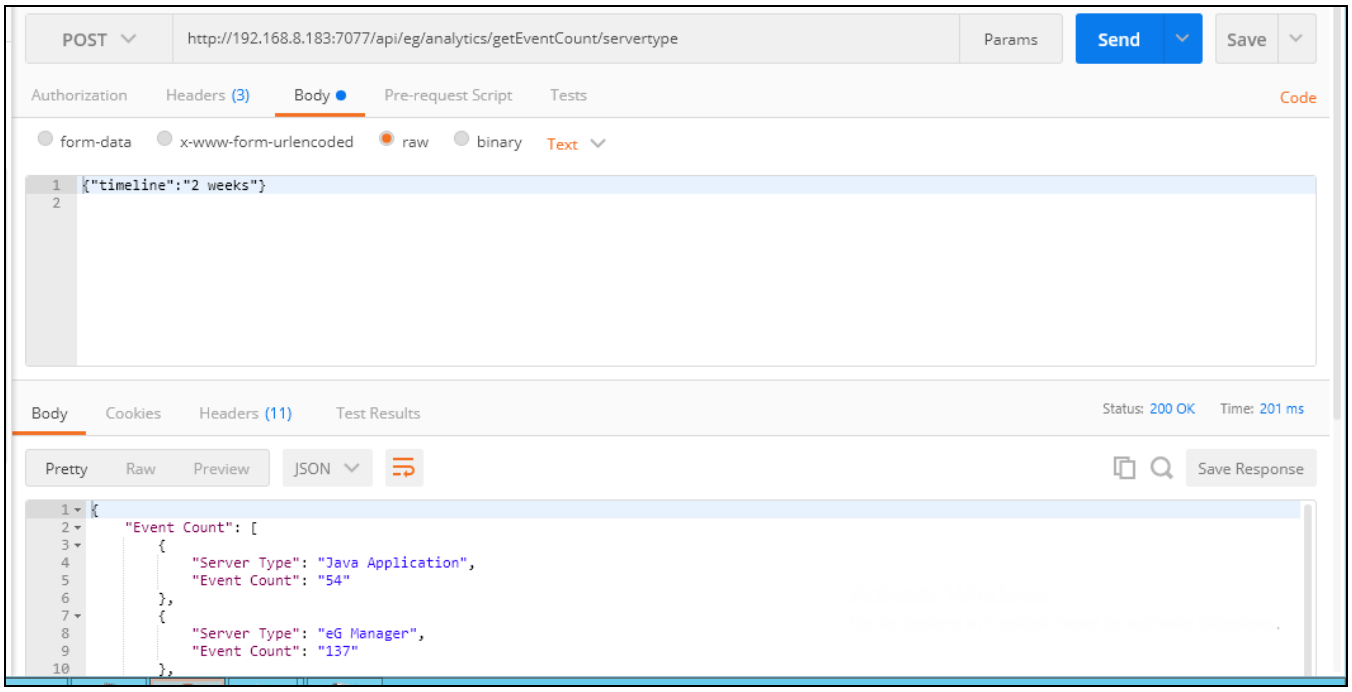

Figure 4.32: Retrieving count of events from Alarm History for all Component Types using Postman REST Client

4.12.2 Retrieving the Count of Events from Alarm History for all Component Types using cURL

To retrieve the count of events for all Component Types using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getEventCount/servertype" -H "user:<eG username or domain/eG
username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -
H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for retrieving
the count of events (in hours/days/weeks)\"}"
```

Figure 4.33 shows an example of retrieving the count of events recorded for all Components Types using cURL.

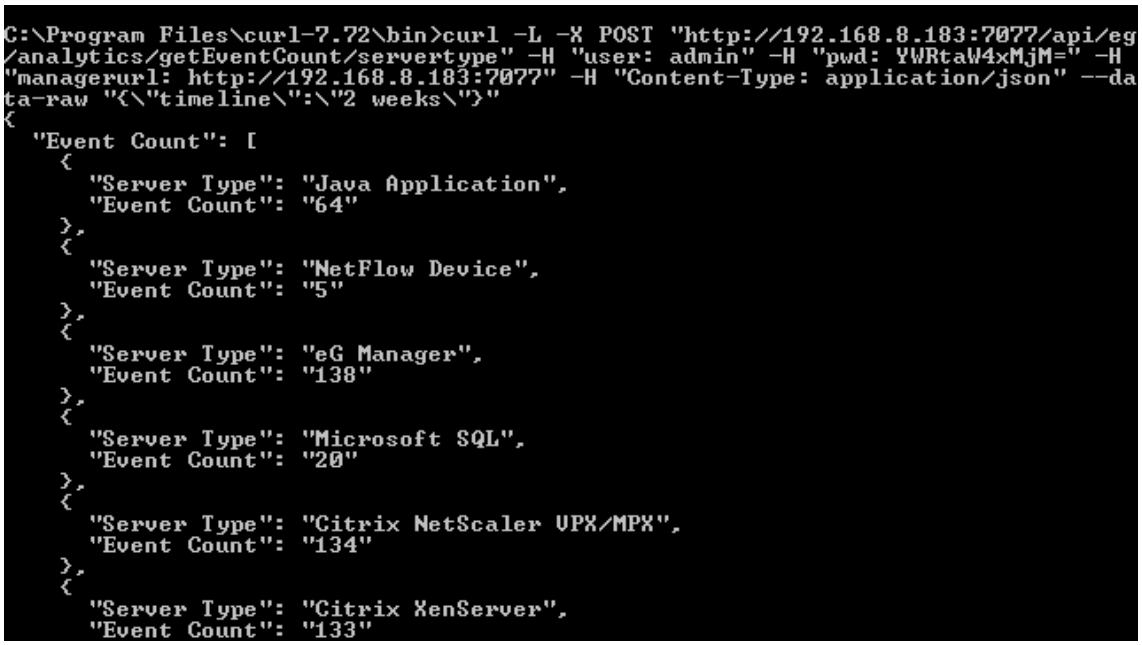

Figure 4.33: Retrieving count of events from Alarm History for all Component Types using cURL

4.12.3 Retrieving the Count of Events for all Components

URL: http://192.168.8.206:7077/api/eg/analytics/getEventCount/servername

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of	{
		"timeline":"1 hour"
		}

Parameters	Key values	Example
	the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	

Body	**Default:**
	{
	"timeline":"Timeline for retrieving the count of events(in hours/days/weeks)"
	}	

Success Response

Type	Code	Content
JSON	200	{ "Event Count": [{ "Server Name": "esx51-15", "Event Count": "185" }, { "Server Name": "win183", "Event Count": "12" }, . . .

Type	Code	Content
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

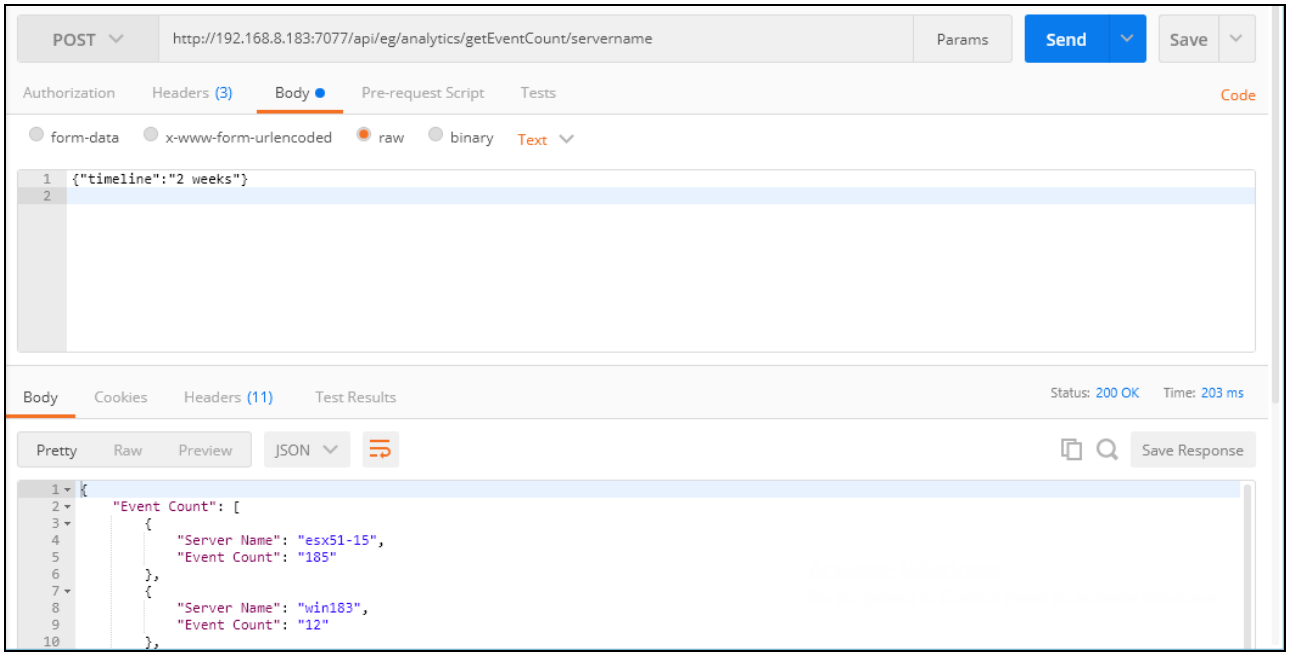

Figure 4.34: Retrieving count of events from Alarm History for all Components using Postman REST Client

4.12.4 Retrieving the Count of Events for all Components using cURL

To retrieve the count of events for all Components using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getEventCount/servername" -H "user:<eG username or domain/eG
username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -
H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for retrieving
the count of events (in hours/days/weeks)\"}"
```

Figure 4.33 shows an example of retrieving the count of events from Alarm History for all Components using cURL.

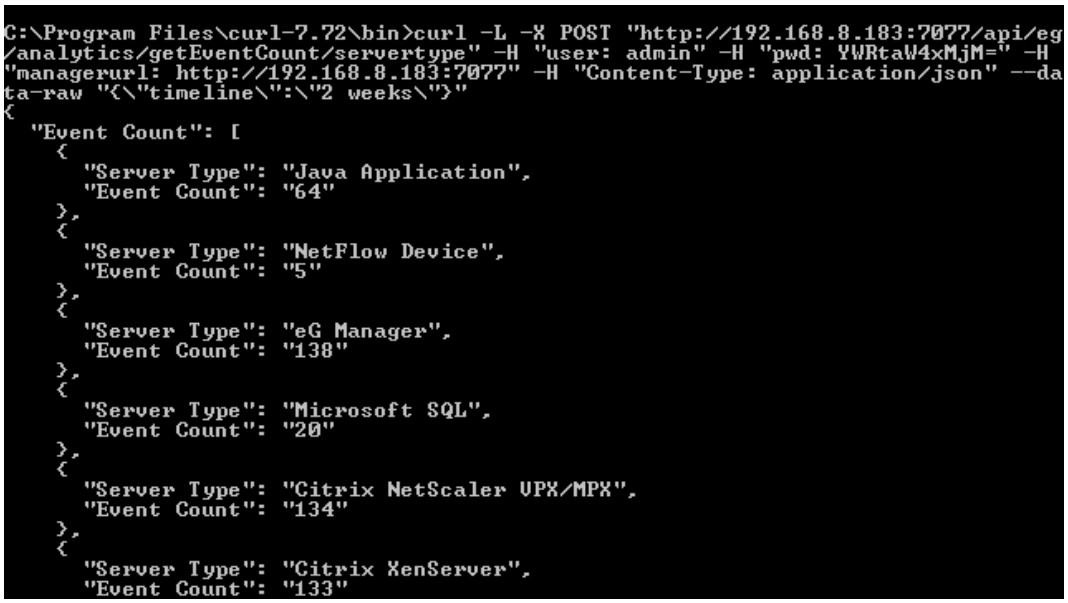

Figure 4.35: Retrieving count of events from Alarm History for all Components using cURL

4.12.5 Retrieving the Count of Events from Alarm History specific to Layers of a Component Type

URL: http://192.168.8.206:7077/api/eg/analytics/getEventCount/layer

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>	{ "timeline":"1 hour", "servertype":"Microsoft Windows"

Parameters	Key values	Example
	user: eG username or domain/eG username pwd: Base64 encoded password	}

Body	**Default:**
	{
	"timeline":"Timeline for retrieving the count of events (in hours/days/weeks)",
	"servertype":"Component Type"
	}	

Success Response

Type	Code	Content
JSON	200	{ "Event Count": [{ "Layer Name": "Operating System", "Event Count": "3" }, { "Layer Name": "Oracle Service", "Event Count": "2" }, . .

Type	Code	Content
		. }

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

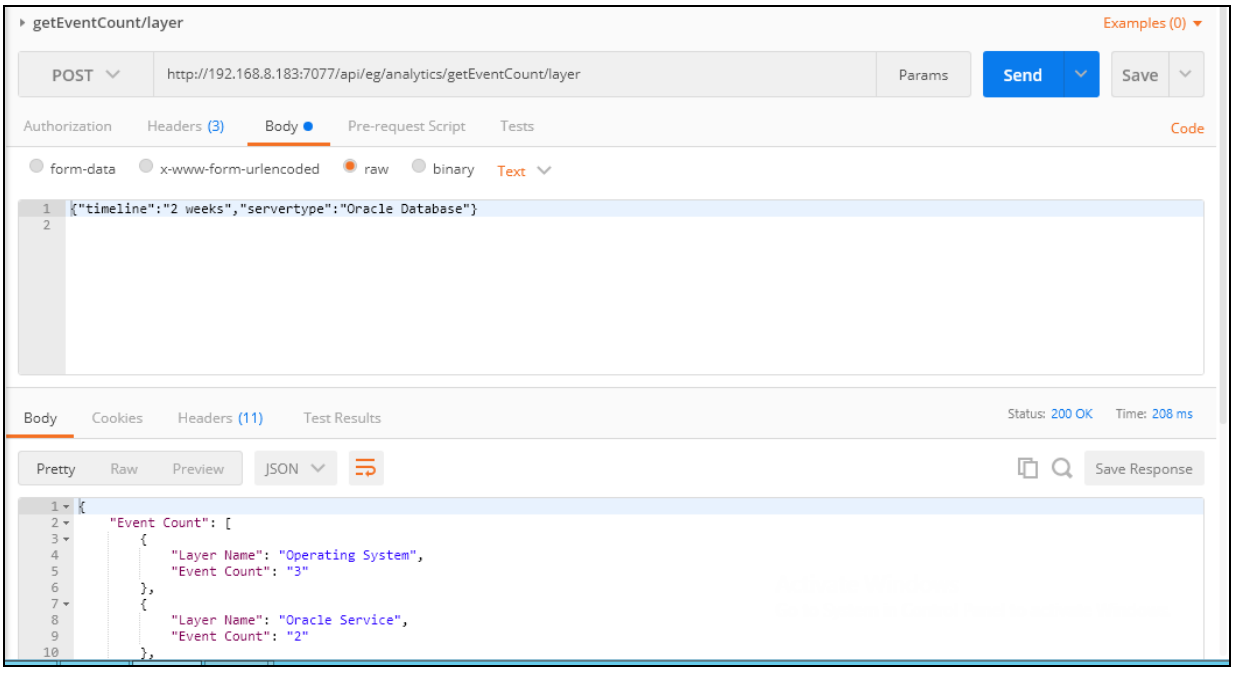

Figure 4.36: Retrieving count of events from Alarm History for the layers of a component type using Postman REST Client

4.12.6 Retrieving the Count of Events from Alarm History specific to Layers of a Component Type using cURL

To retrieve the count of events that are specific to layers of a component type using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getEventCount/layer" -H "user:<eG username or domain/eG
username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -
H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for retrieving
the count of events (in hours/days/weeks)\",\"servertype\":\"Component Type\"}"
```

Figure 4.33 shows an example of retrieving the count of events from Alarm History for the layers of a chosen component type using cURL.

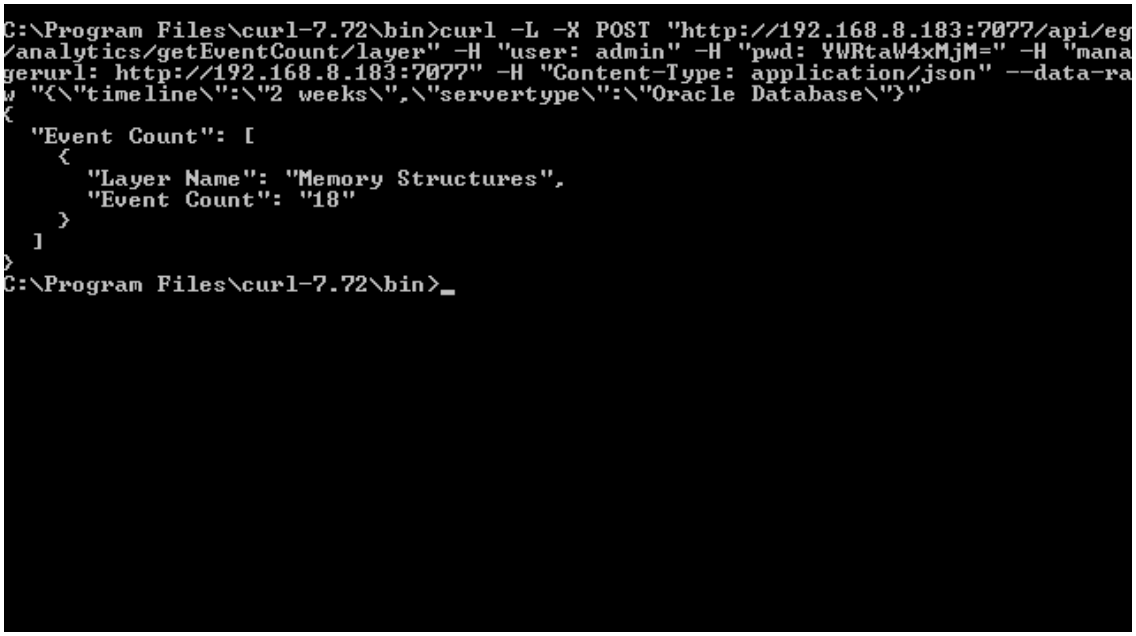

Figure 4.37: Retrieving count of events from Alarm History for the layers of a component type using cURL

4.12.7 Retrieving the Count of Events from Alarm History specific to Tests of a Component Type

URL: http://192.168.8.206:7077/api/eg/analytics/getEventCount/test

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"timeline":"1 hour",
		"servertype":"Microsoft Windows"
		}

Body	**Default:**
	{
	"timeline":"Timeline for retrieving the count of events (in hours/days/weeks)",
	"servertype":"Component Type"
	}	

Success Response

Type	Code	Content
JSON	200	{ "Event Count": [{ "Test Name": "Network", "Event Count": "1" }, { "Test Name": "Oracle SGA",

Type	Code	Content
		"Event Count": "35" }, . . . }

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

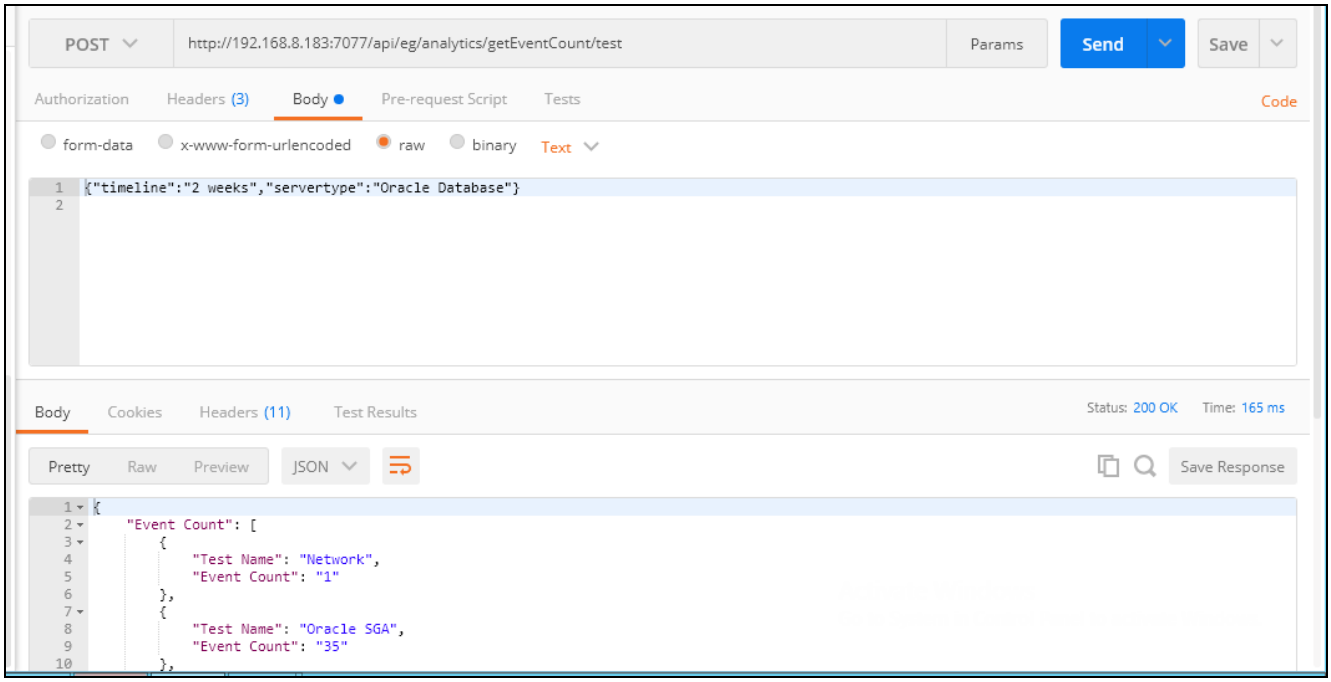

Figure 4.38: Retrieving count of events from Alarm History for the tests of a component type using Postman REST Client

4.12.8 Retrieving the Count of Events from Alarm History specific to Tests of a Component Type using cURL

To retrieve the count of events from Alarm History that are specific to tests of a component type using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getEventCount/test" -H "user:<eG username or domain/eG
username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -
H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for retrieving
the count of events (in hours/days/weeks)\",\"servertype\":\"Component Type\"}"
```

Figure 4.33 shows an example of retrieving the count of events from Alarm History for the tests of a chosen component type using cURL.

Figure 4.39: Retrieving count of events from Alarm History for the tests of a component type using cURL

4.13 Retrieving Problem Duration

Using the eG REST API, administrators can retrieve the duration for which an alarm was open for all component types or components or layers specific to the component type or tests specific to a component type.

4.13.1 Retrieving Problem Duration for Component Types

URL: http://192.168.8.206:7077/api/eg/analytics/getProblemDuration/servertype

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl
	user
	pwd
		{
		"timeline":"1 hour"
		}

Body	**Default:**
	{
	"timeline":"Timeline for retrieving the alarm duration (in hours/days/weeks)"
	}	

Success Response

Type	Code	Content
JSON	200	{ "Problem Duration": [{ "ServerType": "Java Application", "MIN_DURATION": "4m 7s", "MAX_DURATION": "8h 4m", "AVG_DURATION": "18m 23s"

Type	Code	Content
		}, . . . }

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

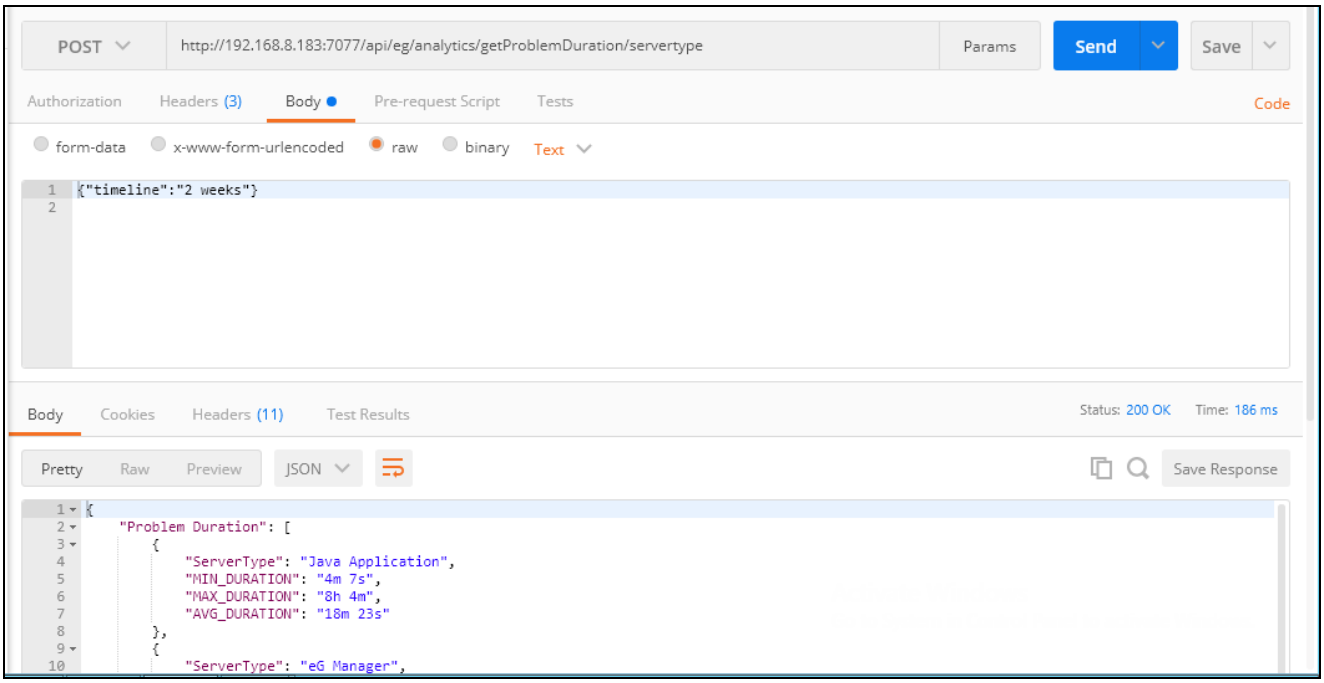

Figure 4.40: Retrieving the duration for which an alarm was open for all Component Types using Postman REST Client

4.13.2 Retrieving Problem Duration for all Component Types using cURL

To retrieve the duration for which an alarm was open for the component types managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getProblemDuration/servertype" -H "user:<eG username or
domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager
IP:Port>" -H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for
retrieving the alarm duration(in hours/days/weeks)\"}"
```

Figure 4.41 shows an example of retrieving the duration for which an alarm was open for the component types managed in the target environment using cURL.

Figure 4.41: Retrieving the duration for which an alarm was open for all Component Types using cURL

4.13.3 Retrieving Problem Duration for all Components

URL: http://192.168.8.206:7077/api/eg/analytics/getProblemDuration/servername

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl
	user
	pwd
		{
		"timeline":"1 hour"
		}

Body	**Default:**
	{
	"timeline":"Timeline for retrieving the alarms (in hours/days/weeks)"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Problem Duration": [
		{
		"Server Name": "esx51-15",
		"MIN_DURATION": "0s",
		"MAX_DURATION": "4D 1h",
		"AVG_DURATION": "19h 21m"
		},
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

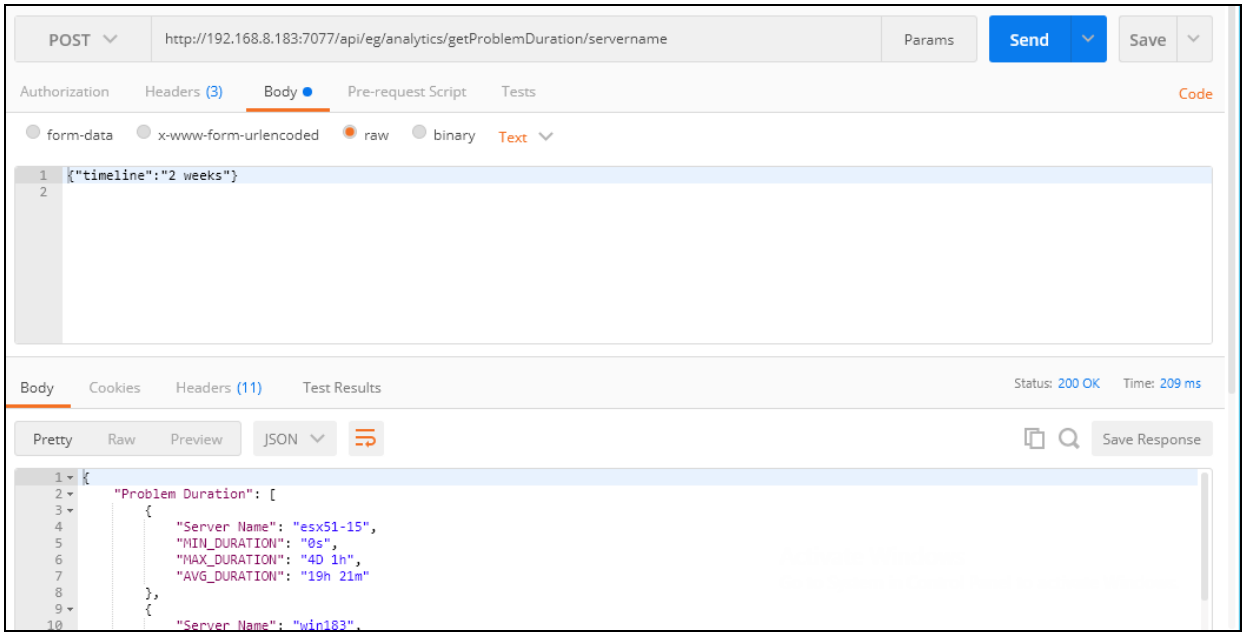

Figure 4.42: Retrieving the duration for which an alarm was open for all Components using Postman REST Client

4.13.4 Retrieving Problem Duration for all Components using cURL

To retrieve the duration for which an alarm was open for all components managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getProblemDuration/servername" -H "user:<eG username or
domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager
IP:Port>" -H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for
retrieving the alarm duration(in hours/days/weeks)\"}"
```

Figure 4.41 shows an example of retrieving the duration for which an alarm was open for all components managed in the target environment using cURL.

Figure 4.43: Retrieving the duration for which an alarm was open for all Components using cURL

4.13.5 Retrieving Problem Duration for all Layers of a Component Type

URL: http://192.168.8.206:7077/api/eg/analytics/getProblemDuration/layer

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl user pwd	{ "timeline":"1 hour", "servertype":Microsoft Windows"
Body	**Default:** { "timeline":"Timeline for retrieving the alarms (in hours/days/weeks)", "servertype":"Component	}

Parameters	Key values	Example
	Type" }	

Success Response

Type	Code	Content

JSON	200	{
		"Problem Duration": [
		{
		"Layer Name": "Operating System",
		"MIN_DURATION": "24m 36s",
		"MAX_DURATION": "24m 36s",
		"AVG_DURATION": "24m 36s"
		},
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

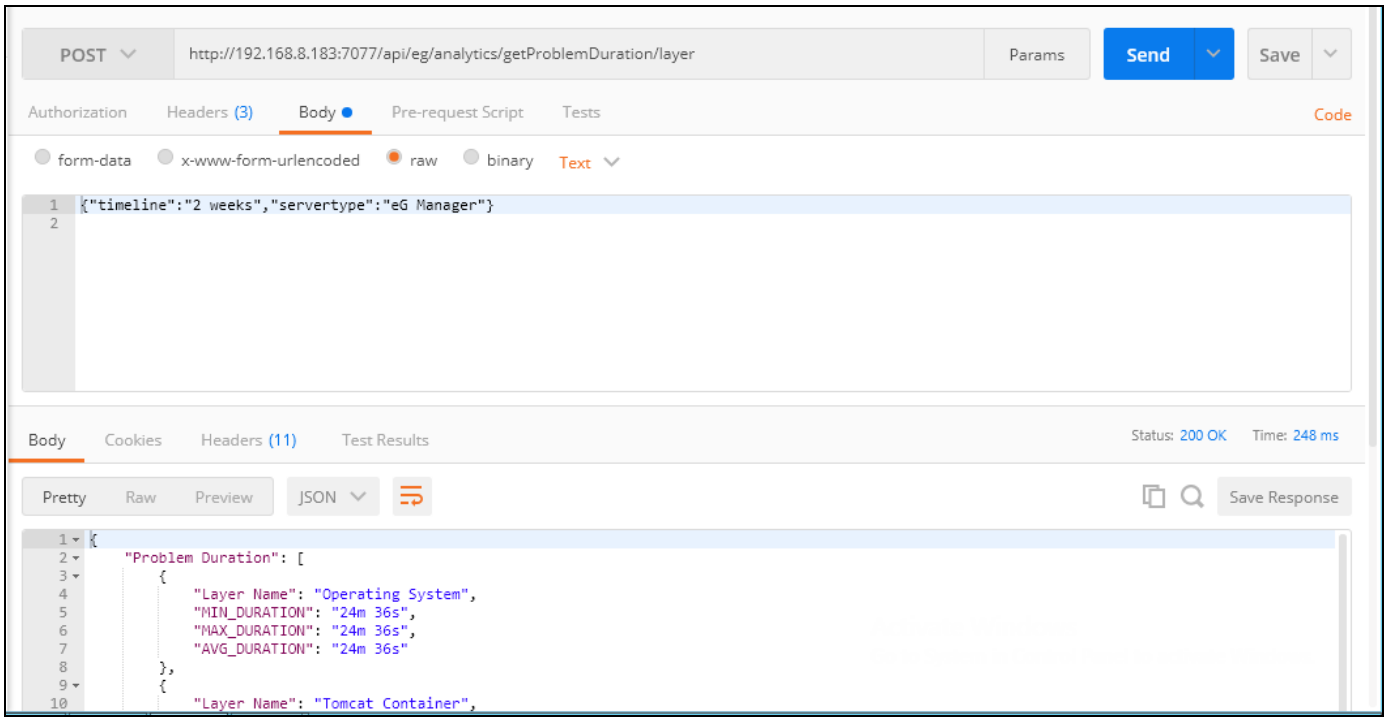

Figure 4.44: Retrieving the duration for which an alarm was open for all layers of a Component types using Postman REST Client

4.13.6 Retrieving Problem Duration for all Layers of a Component Type using cURL

To retrieve the duration for which an alarm was open for all layers of a component type managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager IP:Port>/api/eg/analytics/getProblemDuration/layer" -H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for retrieving the alarm duration(in hours/days/weeks)\",\"servertype\":\"Component Type\"}"
```

Figure 4.41 shows an example of retrieving the duration for which an alarm was open for all layers corresponding to a component type managed in the target environment using cURL.

Figure 4.45: Retrieving the duration for which an alarm was open for all layers of a Component types using cURL

4.13.7 Retrieving Problem Duration for all Tests of a Component Type

URL: http://192.168.8.206:7077/api/eg/analytics/getProblemDuration/test

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"timeline":"1 hour",
		"servertype":Microsoft Windows"
		}

Body	**Default:**	

Parameters	Key values	Example

	{
	"timeline":"Timeline for retrieving the alarms (in hours/days/weeks)",
	"servertype":"Component Type"
	}	

Success Response

Type	Code	Content

JSON	200	{
		"Problem Duration": [
		{
		"Test Name": "Uptime",
		"MIN_DURATION": "4m 11s",
		"MAX_DURATION": "5m 6s",
		"AVG_DURATION": "4m 38s"
		},
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401	{"code": 401,"error": "Unauthorized user"}

Type	Code	Content
	UNAUTHORIZED	
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

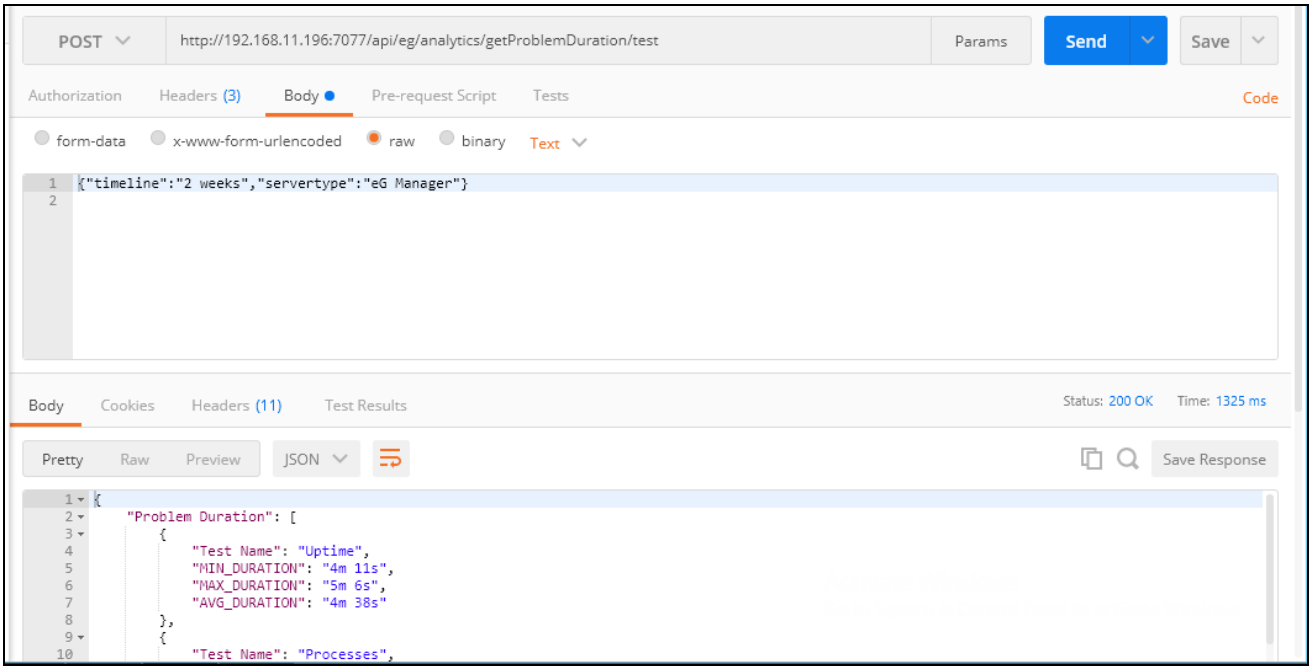

Figure 4.46: Retrieving the duration for which an alarm was open for all Tests of a Component Type using Postman REST Client

4.13.8 Retrieving Problem Duration for all Tests of a Component Type using cURL

To retrieve the duration for which an alarm was open for all tests of a component type managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getProblemDuration/test" -H "user:<eG username or domain/eG
username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" -
H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for retrieving
the alarm duration(in hours/days/weeks)\",\"servertype\":\"Component Type\"}"
```

Figure 4.41 shows an example of retrieving the duration for which an alarm was open for all tests of a component type managed in the target environment using cURL.

Figure 4.47: Retrieving the duration for which an alarm was open for all Tests of a Component Type using cURL

4.14 Retrieving Percentage of Proactive Alarms in the Target Environment

Use the URL specified below to retrieve the percentage of proactive alarms for all component types, components, layers and tests specific to the target environment. The percentage is calculated by considering the count of Major and Minor alarms against the Total alarms raised in the target environment.

4.14.1 Retrieving Percentage of Proactive Alarms across Component Types

URL: http://192.168.8.206:7077/api/eg/analytics/getProactiveProblemPercent/servertype

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e.,	{ "timeline":"1 hour"

Parameters	Key values	Example
	http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	}

Body	**Default:**
	{
	"timeline":"Timeline for retrieving proactive alarm percent (in hours/days/weeks)"
	}	

Success Response

Type	Code	Content
JSON	200	{ "Proactive Problem Percent": [{ "Server Type": "Java Application", "ProactivePercent": "100" }, { "Server Type": "NetFlow Device", "ProactivePercent": "0" }, . .

Type	Code	Content
		. }

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

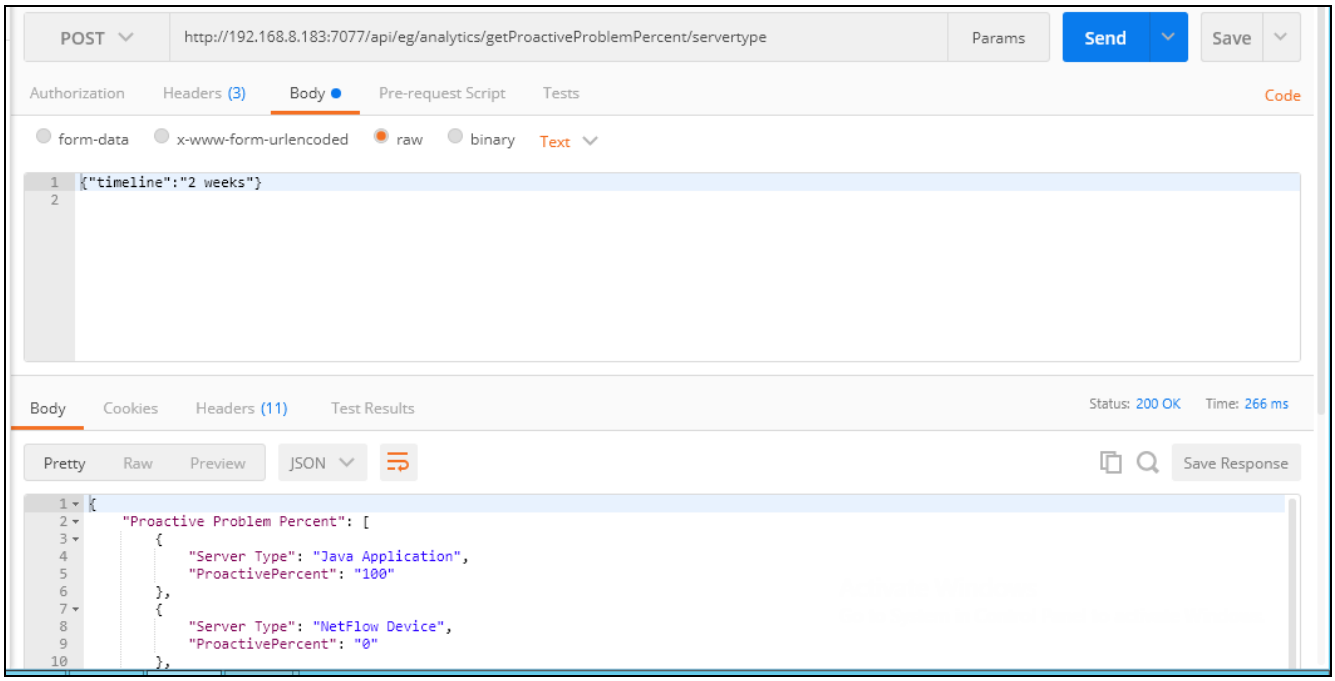

Figure 4.48: Retrieving the percentage of proactive alarms for all Component Types using Postman REST Client

4.14.2 Retrieving Percentage of Proactive Alarms across Component Types using cURL

To retrieve the percentage of proactive alarms across all component types managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getProactiveProblemPercent/servertype" -H "user:<eG username or
domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager
IP:Port>" -H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for
retrieving proactive alarm percent(in hours/days/weeks)\"}"
```

Figure 4.49 shows an example of retrieving the percentage of proactive alarms for all component types managed in the target environment using cURL.

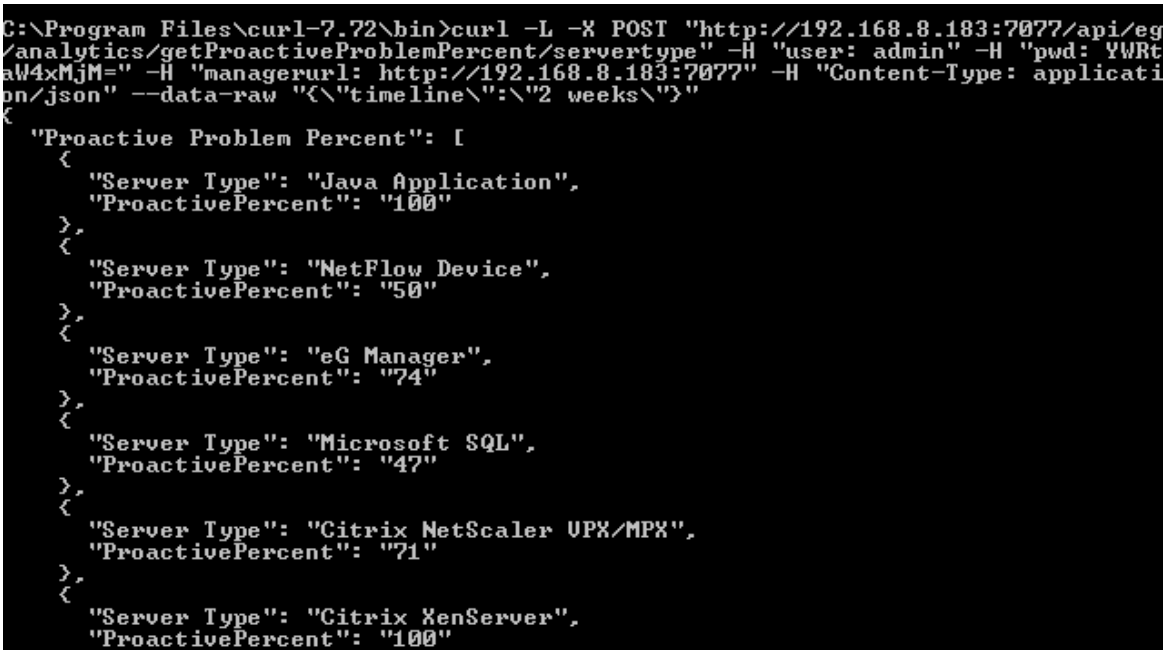

Figure 4.49: Retrieving the percentage of proactive alarms for all Component Types using cURL

4.14.3 Retrieving Percentage of Proactive Alarms across all Components

URL: http://192.168.8.206:7077/api/eg/analytics/getProactiveProblemPercent/servername

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"timeline":"1 hour"
		}

Body	**Default:**
	{
	"timeline":"Timeline for retrieving proactive alarm percent (in hours/days/weeks)"
	}	

Success Response

Type	Code	Content
JSON	200	{ "Proactive Problem Percent": [{ "Server Name": "esx51-15", "ProactivePercent": "80" }, { "Server Name": "win183",

Type	Code	Content
		"ProactivePercent": "93" }, . . . }

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

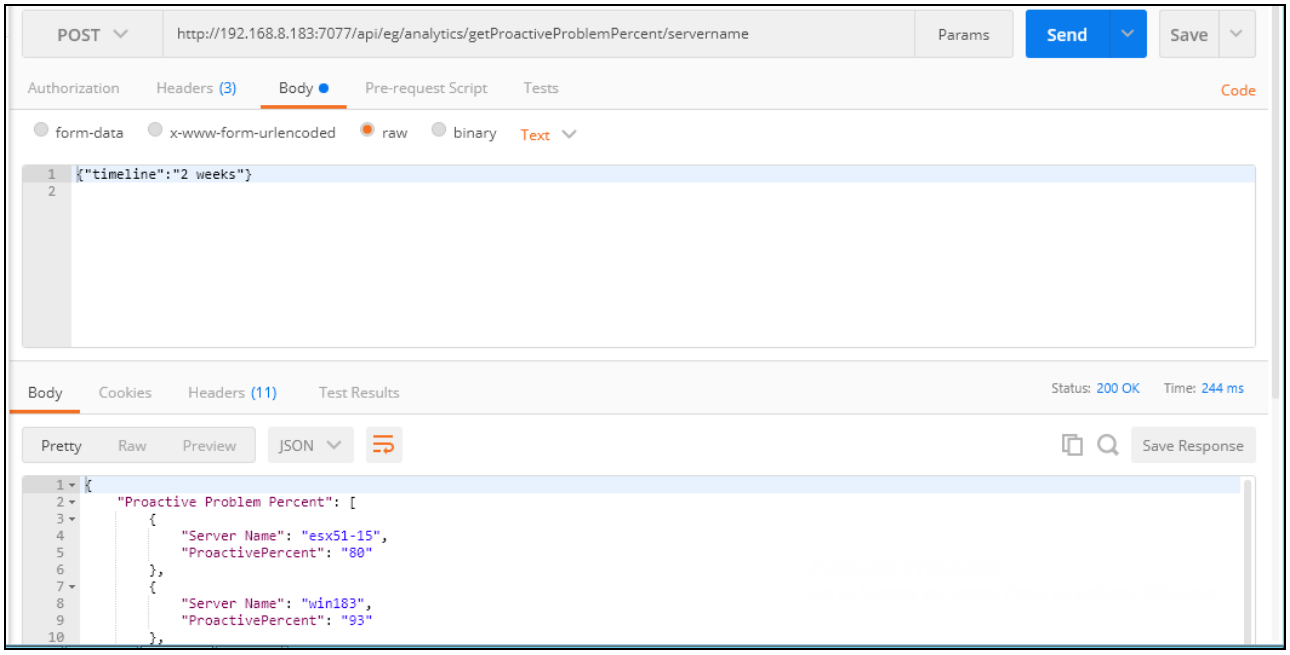

Figure 4.50: Retrieving the percentage of proactive alarms for all Components using Postman REST Client

4.14.4 Retrieving Percentage of Proactive Alarms across Components using cURL

To retrieve the percentage of proactive alarms across all components managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getProactiveProblemPercent/servername" -H "user:<eG username or
domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager
IP:Port>" -H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for
retrieving proactive alarm percent(in hours/days/weeks)\"}"
```

Figure 4.49 shows an example of retrieving the percentage of proactive alarms for all components managed in the target environment using cURL.

Figure 4.51: Retrieving the percentage of proactive alarms for all Components using cURL

4.14.5 Retrieving Percentage of Proactive Alarms specific to Layers of a Component Type

URL: http://192.168.8.206:7077/api/eg/analytics/getProactiveProblemPercent/layer

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example

Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port>
	user: eG username or domain/eG username
	pwd: Base64 encoded password
		{
		"timeline":"1 hour",
		"servertype":"Microsoft Windows"
		}

Body	**Default:**
	{
	"timeline":"Timeline for retrieving proactive alarm percent (in hours/days/weeks)",
	"servertype":"Component Type"
	}	

Success Response

Type	Code	Content
JSON	200	{ "Proactive Problem Percent": [{ "Layer Name": "Operating System", "ProactivePercent": "95" }, { "Layer Name": "eG Application",

Type	Code	Content
		"ProactivePercent": "38" }, . . . }

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

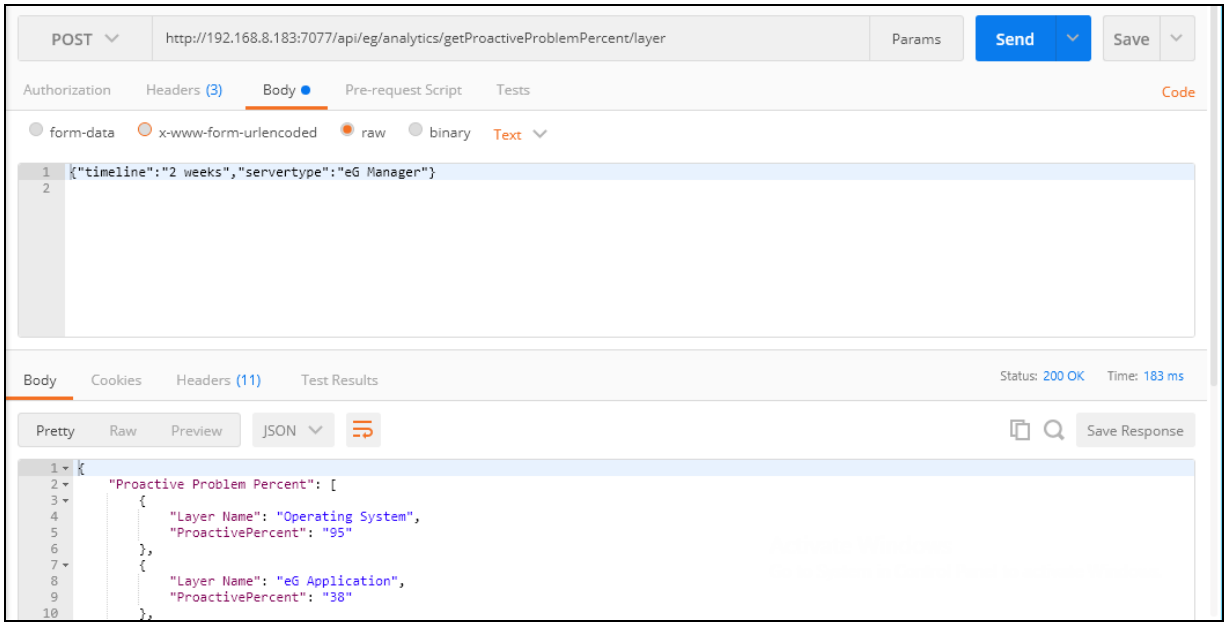

Figure 4.52: Retrieving the percentage of proactive alarms for the layers of a component type using Postman REST Client

4.14.6 Retrieving Percentage of Proactive Alarms specific to Layers of a Component Type using cURL

To retrieve the percentage of proactive alarms specific to layers of a component type managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/analytics/getProactiveProblemPercent/layer" -H "user:<eG username or
domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager
IP:Port>" -H "Content-Type: application/json" --data-raw "{\"timeline\":\"Timeline for
retrieving proactive alarm percent(in hours/days/weeks)\",\"servertype\":\"Component
Type\"}"
```

Figure 4.49 shows an example of retrieving the percentage of proactive alarms for the layers specific to a component type managed in the target environment using cURL.

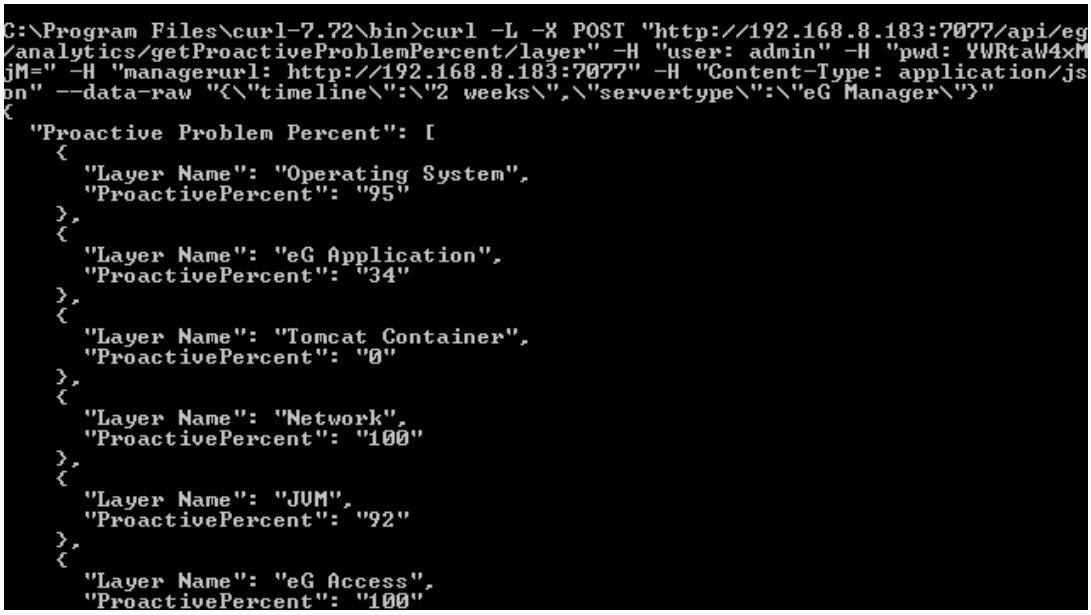

Figure 4.53: Retrieving the percentage of proactive alarms for the layers of a component type using cURL

Chapter 5: Extracting Miscellaneous Data from eG Manager Using eG REST API

In some environments, administrators may need certain additional information with respect to the infrastructure configured in the eG manager. For example, administrators may need to figure out the tests and measures supported by eG Enterprise. Such data can be retrieved with ease using the REST API commands.

The sections below will discuss in detail on the miscellaneous data that can be extracted from the eG Manager.

5.1 Retrieving Details of Components Managed in the target environment

The eG REST API can be used to retrieve all the components managed in the eG Manager along with their respective Component Types. For this, specify the URL in the following format:

URL: http://192.168.8.206:7077/api/eg/miscservice/getComponentMapping

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	Not Applicable

Success Response

Type	Code	Content

JSON	200	[
		{
		"ComponentType": "Cisco_router",
		"servers": [
		"cisco_router_d"
		]
		},
		{
		"ComponentType": "Citrix_NetScaler",
		"servers": [
		"Netscaler176",
		"Netscaler"
		]
		},
		.
		.
		.
		}
		]

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

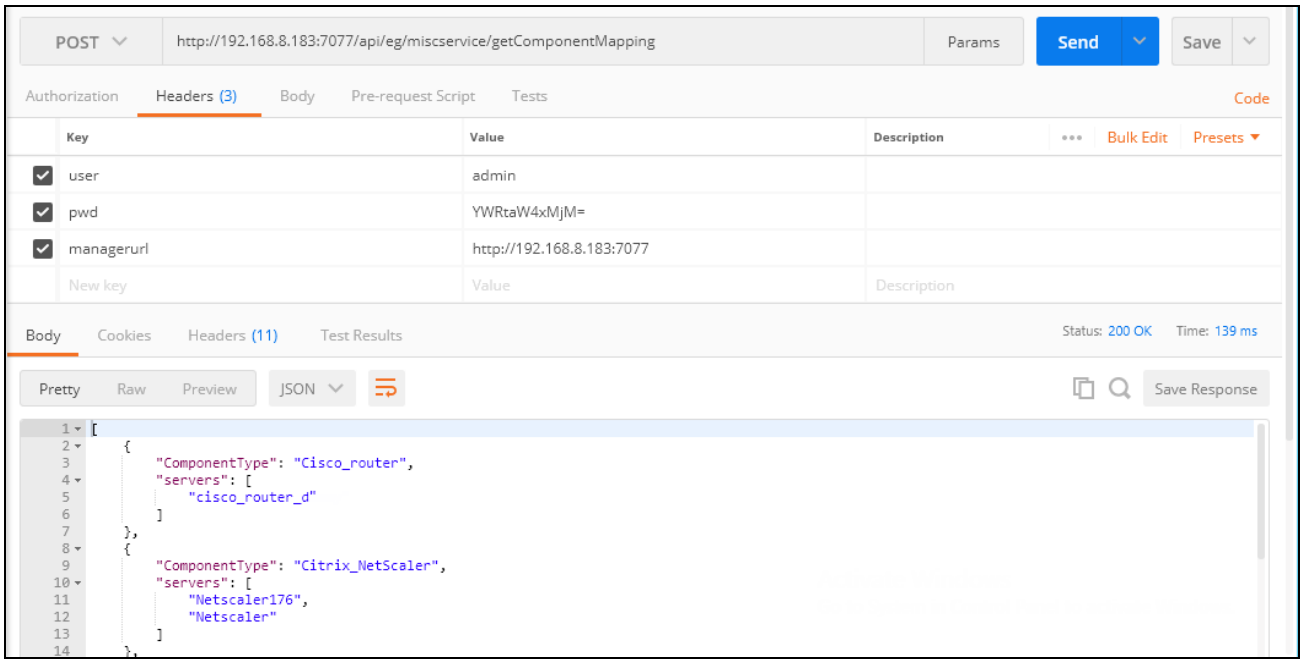

Figure 5.1: Retrieving the components corresponding to all Component Types using Postman REST Client

5.1.1 Retrieving Details of Components Managed in the target environment using cURL

To retrieve the components corresponding to all component types managed in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager IP:Port>/api/eg/miscservice/getComponentMapping" -H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" --data-raw ""
```

Figure 5.2 shows an example for retrieving the components corresponding to all component types managed in the target environment using cURL.

Figure 5.2: Retrieving the components corresponding to all Component Types using cURL

5.2 Retrieving Zone Details from eG Manager

To retrieve the details of the zones and the elements associated with the zone (services, segments, servers etc), administrators can use the eG REST API. The URL can be specified in the following format:

URL: http://192.168.8.206:7077/api/eg/miscservice/getZoneMapping

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded	Not Applicable

Parameters	Key values	Example
	password	

Success Response

Type	Code	Content

JSON	200	[
		{
		"zone": "Zone-VDI",
		"Group": [],
		"Service": [],
		"Segment": [],
		"Server": [
		"eGDP169:7077",
		"java183:13600",
		"win183",
		"mssql100:1433"
		]
		},
		.
		.
		.
		}
		]

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

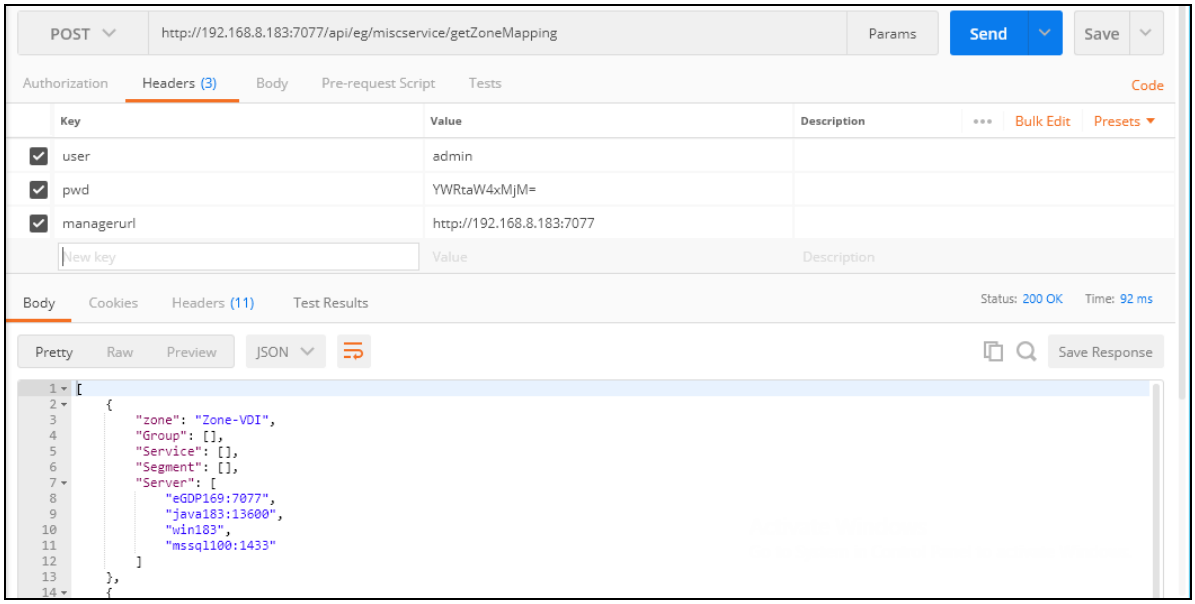

Figure 5.3: etrieving the details of the zones created in the target environment using Postman REST Client

5.2.1 Retrieving Zone Details from eG Manager using cURL

To retrieve the details of the zones and the elements associated with the zone (services, segments, servers etc) in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/miscservice/getZoneMapping" -H "user:<eG username or domain/eG username>"
-H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" --data-raw
""
```

Figure 5.4 shows an example for retrieving the details of all the zones created in the target environment using cURL.

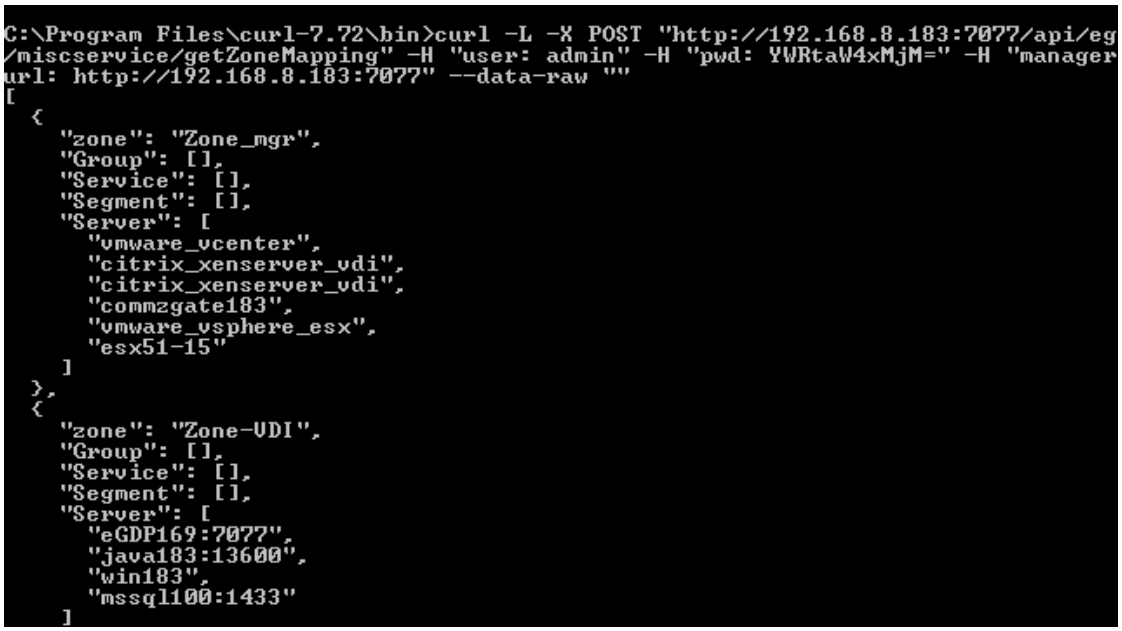

Figure 5.4: Retrieving the details of the zones created in the target environment using cURL

5.3 Retrieving the Tests Supported by eh Enterprise Using eG REST API

To retrieve the tests that are available in eG Enterprise for execution, by default, administrators can use the eG REST API. The URL can be specified in the following format:

URL: http://192.168.8.206:7077/api/eg/miscservice/getTestMapping

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username	Not Applicable

Parameters	Key values	Example
	pwd: Base64 encoded password	

Success Response

Type	Code	Content

JSON	200	{
		"IBSubnetMgrTest": "IB Subnet Manager Statistics",
		"IBSmaPortTest": "IB SMA Port",
		"IBPmaExtPortTest": "IB PMA Extended Port",
		"IBPmaPortTest": "IB PMA Port",
		"IBFabricElemTest": "Fabric Elements",
		"Db2DPFSQLNetTest": "Db2 DPF SQL Network",
		"FileUpdateTest": "File Updates",
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

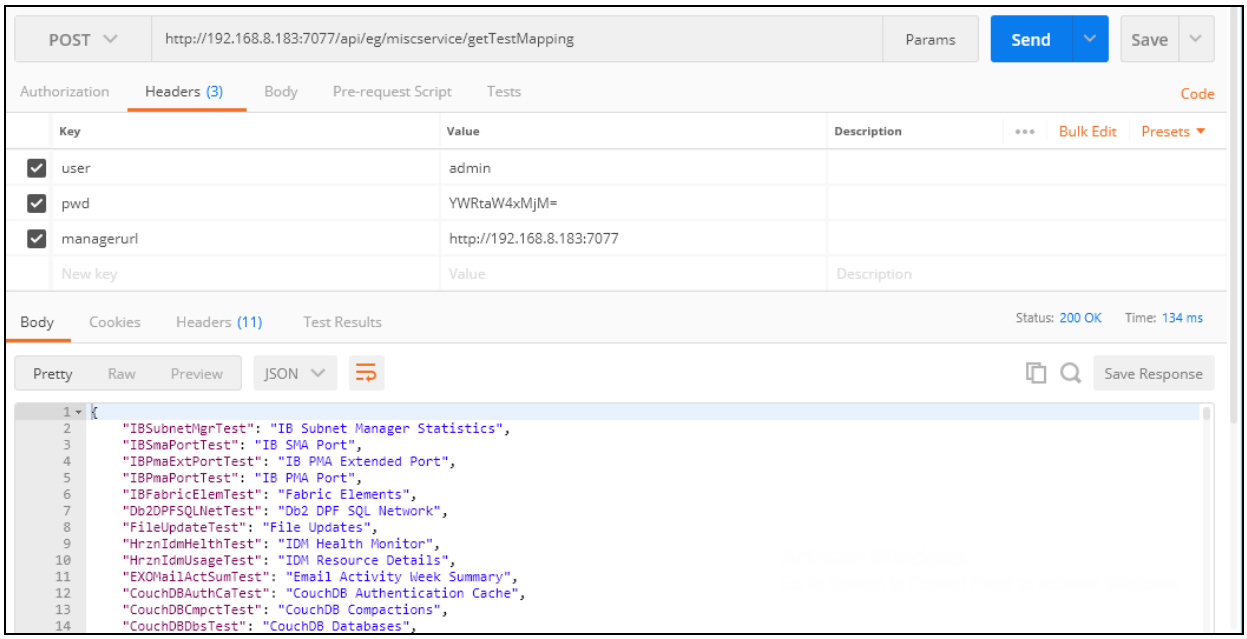

Figure 5.5: Retrieving the tests supported by eG Enterprise using Postman REST Client

5.3.1 Retrieving the Tests Supported by eG Enterprise using cURL

To retrieve the details of the tests available in eG Enterprise for execution in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager
IP:Port>/api/eg/miscservice/getTestMapping" -H "user:<eG username or domain/eG username>"
-H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" --data-raw
""
```

Figure 5.6 shows an example cURL command for retrieving the details of all the tests available for execution in the target environment.

Figure 5.6: An example cURL command to retrieve the tests supported by eG Enterprise

Figure 3 shows a sample output that retrieves the tests supported by eG Enterprise using cURL.

Figure 5.7: Sample output with the list of tests supported by eG Enterprise

5.4 Retrieving the Measurements Reported by eG Enterprise

To retrieve the measures that are reported by eG Enterprise, by default, administrators can use the eG REST API. The URL can be specified in the following format:

URL: http://192.168.8.206:7077/api/eg/miscservice/getMeasureMapping

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded	Not Applicable

Parameters	Key values	Example
	password	

Success Response

Type	Code	Content

JSON	200	{
		"IBSubnetMgrTest:smCntSMPsOutstanding": "Outstanding packets",
		"IBSubnetMgrTest:smCntSMPsOnWire": "Onwire packets",
		"IBSubnetMgrTest:smCntSMPsReceived": "Packets received",
		"IBSubnetMgrTest:smCntSMPsSent": "Packets transmitted",
		"IBSubnetMgrTest:smCntSMPsUnidirect": "Responseless packets transmitted",
		"IBSubnetMgrTest:smCntSMPsUnknownReceived": "Unknown packets received",
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

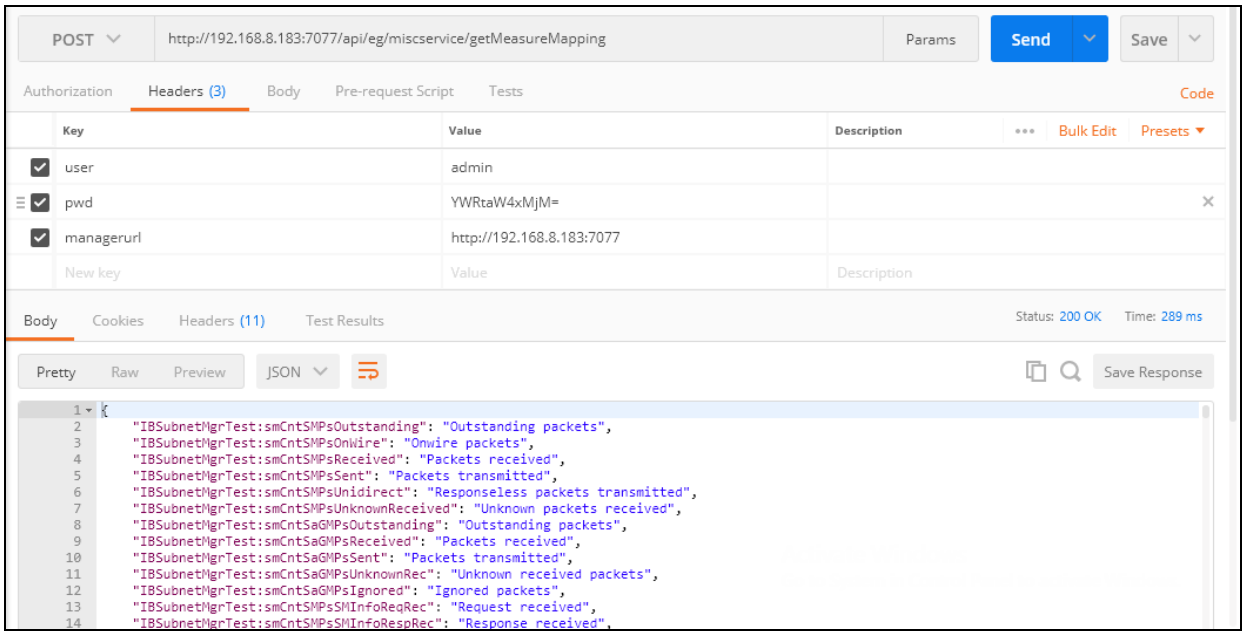

Figure 5.8: Retrieving the list of measurements using Postman REST Client

5.4.1 Retrieving the Measurements Reported by eG Enterprise using cURL

To retrieve the measures reported by eG Enterprise by monitoring the components in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager IP:Port>/api/eg/miscservice/getMeasureMapping" -H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" --data-raw ""
```

Figure 5.9 shows an example cURL command for retrieving the measurements reported by eG Enterprise by monitoring the components in the target environment.

Figure 5.9: An example cURL command to retrieve the measurements

Figure 3 shows a sample output that retrieves the measurements reported by eG Enterprise using cURL.

Figure 5.10: Sample output with the list of measurements supported by eG Enterprise

5.5 Retrieving Applications Monitored by eG Enterprise Using eG REST API

In order to retrieve the applications that are monitored by eG Enterprise by default, administrators can use the eG REST API.

URL: http://192.168.8.206:7077/api/eg/miscservice/getApplicationMapping

Method: POST

Content-Type: application/json

Inputs to be Specified

Parameters	Key values	Example
Headers	managerurl: Base URL of the eG Manager i.e., http://<IP address of the eG console:Port> user: eG username or domain/eG username pwd: Base64 encoded password	Not Applicable

Success Response

Type	Code	Content

JSON	200	{
		"Infiniband_Switch": "InfiniBand Switch",
		"MSExchangeOnline_domain": "Exchange Online Domain",
		"MSExchangeOnline_service": "Exchange Online Tenant",
		"CouchDB_Server": "Apache CouchDB",
		"VMWareHorizon_Workspace_one": "Vmware Horizon Workspace One",
		"Alibaba_Cloud": "Alibaba Cloud",
		"OracleExadataStorage": "Oracle Exadata Storage Server",
		.
		.
		.
		}

Failure Response

Type	Code	Content
JSON	401 UNAUTHORIZED	{"code": 401,"error": "Unauthorized user"}
JSON	500 Server Error	{"code": 500,"error": " Server Error "}

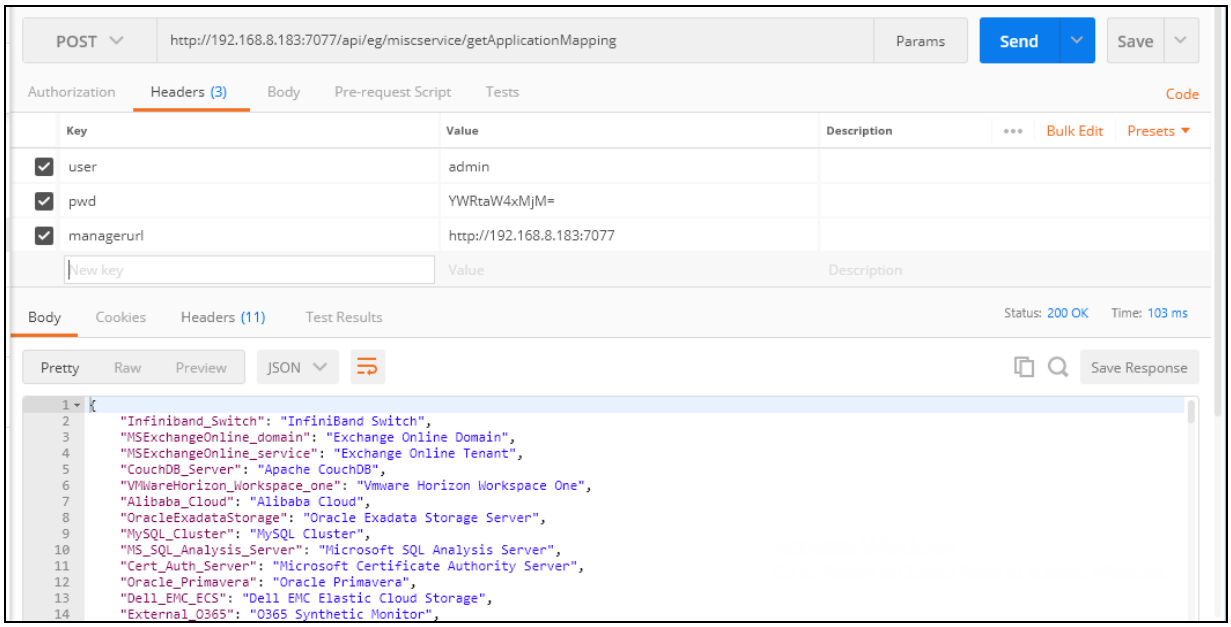

Figure 5.11: Retrieving the list of applications monitored using Postman REST Client

5.5.1 Retrieving the Applications Monitored by eG Enterprise using cURL

To retrieve the measures reported by eG Enterprise by monitoring the components in the target environment using cURL, the command should be specified in the following format:

```
curl -location -request POST "http://<eG Manager IP:Port>/api/eg/miscservice/getApplicationMapping" -H "user:<eG username or domain/eG username>" -H "pwd:Base64 encoded password" -H "managerurl:http://<eG Manager IP:Port>" --data-raw ""
```

Figure 5.12 shows an example cURL command for retrieving the applications monitored by eG Enterprise.

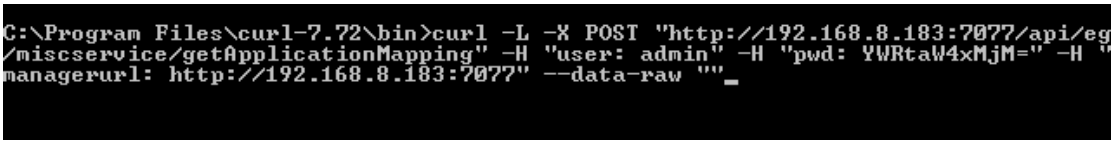

Figure 5.12: An example cURL command to retrieve the applications monitored by eG Enterprise

Figure 3 shows a sample output that retrieves the measurements reported by eG Enterprise using cURL.

Figure 5.13: Sample output with the list of applications monitored by eG Enterprise

About eG Innovations

eG Innovations provides intelligent performance management solutions that automate and dramatically accelerate the discovery, diagnosis, and resolution of IT performance issues in on-premises, cloud and hybrid environments. Where traditional monitoring tools often fail to provide insight into the performance drivers of business services and user experience, eG Innovations provides total performance visibility across every layer and every tier of the IT infrastructure that supports the business service chain. From desktops to applications, from servers to network and storage, from virtualization to cloud, eG Innovations helps companies proactively discover, instantly diagnose, and rapidly resolve even the most challenging performance and user experience issues.

eG Innovations is dedicated to helping businesses across the globe transform IT service delivery into a competitive advantage and a center for productivity, growth and profit. Many of the world's largest businesses use eG Enterprise to enhance IT service performance, increase operational efficiency, ensure IT effectiveness and deliver on the ROI promise of transformational IT investments across physical, virtual and cloud environments.

To learn more visit www.eginnovations.com.

Contact Us

For support queries, email support@eginnovations.com.

To contact eG Innovations sales team, email sales@eginnovations.com.