End to End Automation with REST API in Dell EMC OpenManage Enterprise 3.0

This technical paper describes the uses of REST API use in Dell EMC OpenManage Enterprise 3.0 for Systems Management.

Abstract

This technical paper describes the uses of REST API use in Dell EMC OpenManage Enterprise 3.0 for Systems Management.

January 2019

Revisions

Date	Description
January 2019	Initial release

Acknowledgements

This paper was produced by the following members of the Dell EMC storage engineering team:

Authors: Reg Stumpe, Pushkala Iyer, Raajeev Kalyanaraman, Amit M Anand, Rakesh Ayolasomyajul, Bhavya DN

The information in this publication is provided “as is.” Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

© <Jan/04/2019> Dell Inc. or its subsidiaries. All Rights Reserved. Dell, EMC, Dell EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be trademarks of their respective owners.

Dell believes the information in this document is accurate as of its publication date. The information is subject to change without notice.

Contents

1 Executive summary

OpenManage Enterprise 3.0 is the logical follow-on to the existing OpenManage Essentials (OME) console or its predecessor OpenManage–Tech Release. OpenManage Enterprise provides REST APIs that can be used by an administrator who wants to invoke it in a programmatic manner. This technical white paper describes some of those use-cases and puts lists the corresponding APIs that must be used.

2 Introduction to REST API in OpenManage Enterprise

OpenManage Enterprise 3.0 is the logical follow-on to the existing OpenManage Essentials (OME) console or its predecessor OpenManage Tech Release. The core features of a management console, such as Discovery, Inventory, Monitoring Alerts, Firmware Updates, Reporting, and Configuration are supported with OpenManage Enterprise 3.0. Delivered as an appliance, a program requirement for OpenManage Enterprise has been to provide a comprehensive set of NorthBound (NB) REST APIs which can be used to for programmatic invocation.

After a successful login, https://IP/api will return the list of services supported by the appliance. The purpose of this document is to provide a short set of examples, for end to end automation of common tasks. This technical white paper is not a replacement for the OpenManage Enterprise 3.0 API guide which provides a comprehensive list of all the APIs that are supported.

To use the OpenManage Enterprise API, you can use a tool such as PostMan, or use Python or PowerShell scripts. This technical white paper will use both PostMan screenshots and script examples as appropriate.

2.1 Discovering devices

The DiscoveryConfigService on OpenManage Enterprise encapsulates device discovery functionality.

A Discovery task can be created via REST by a POST to the DiscoveryConfigService with the following payload. In this case the Discovery task is being created on the range: 10.35.0.0-10.35.0.255, and being asked to discover Servers by using the `root` / `calvin` credentials.

The Guided Edit offers a simplified view of the server configuration attributes. These are a few of the BIOS, Boot, Networking, and Storage configuration.

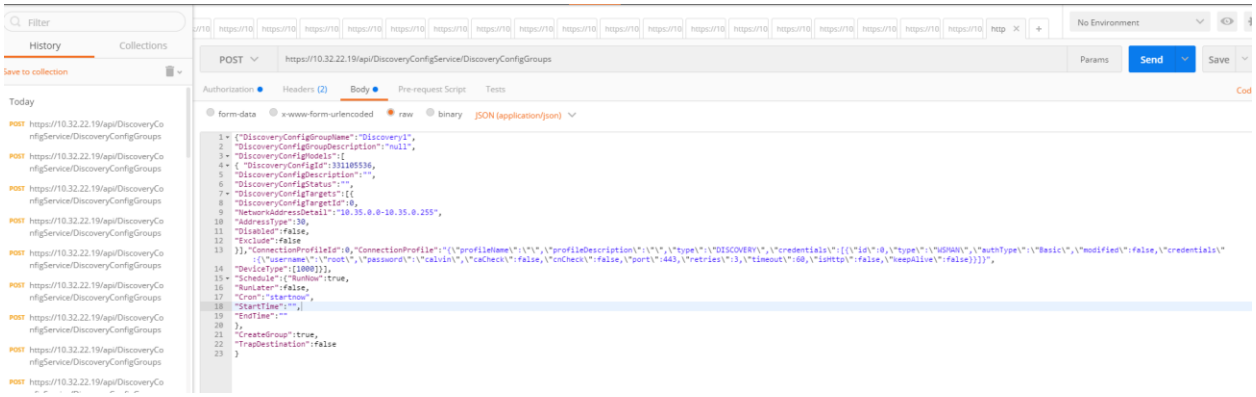


Figure 1 Request to create a discovery job

A successful discovery task creation returns the following:

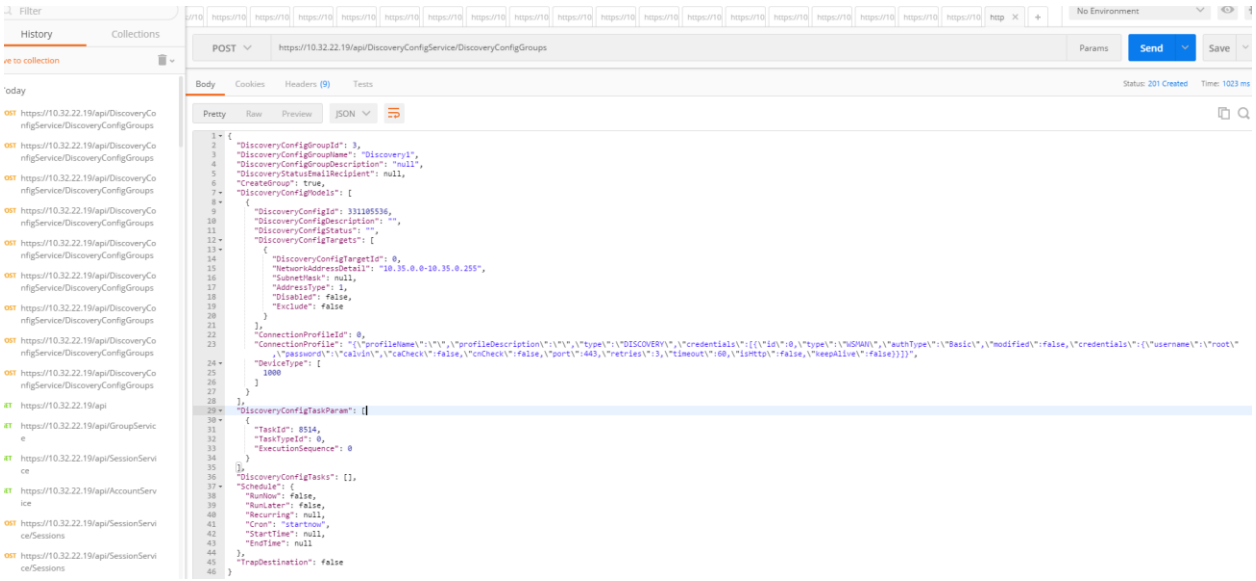


Figure 2 Response payload from DiscoveryConfigService

Enumerating the Jobs by using the `JobService` will indicate that the Discovery Job has run, because it was instructed to run immediately (Run Now). The JobService on the OpenManage Enterprise appliance provides information of what jobs (tasks) are in the appliance, their state (running, scheduled, completed, or failed), and provides methods to create and schedule jobs.

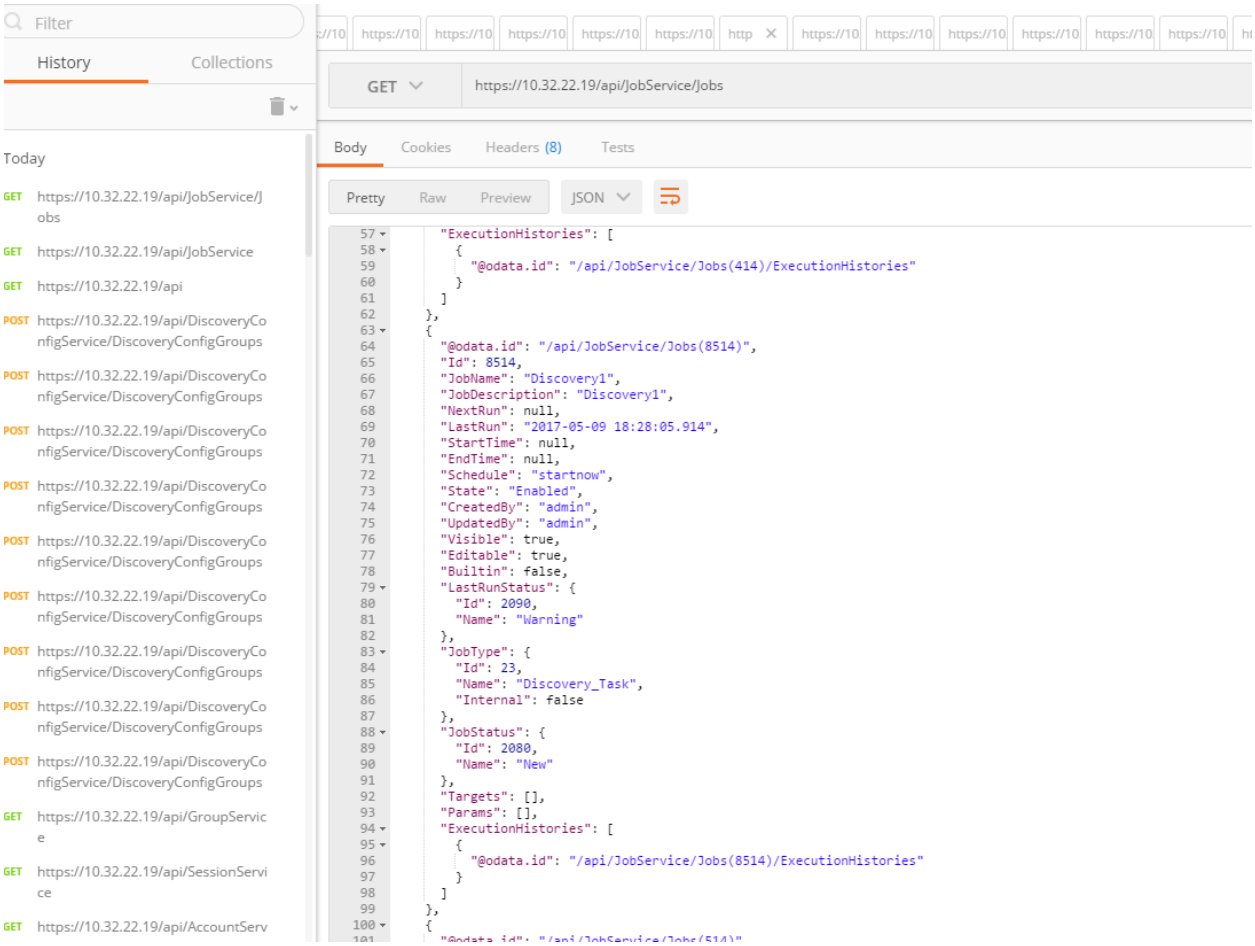


Figure 3 Job Service response listing discovery job

Note—A python script to discover devices by hostname is attached.

2.2 Listing devices, and filtering by type and status

After devices are discovered, they can be enumerated by using the `DeviceService`. The `DeviceService` on the OpenManage Enterprise appliance encapsulates all details about the devices managed by the appliance, such as the device types that are managed, devices that are managed, their status, and devices that can be globally excluded from being managed or monitored by the console.

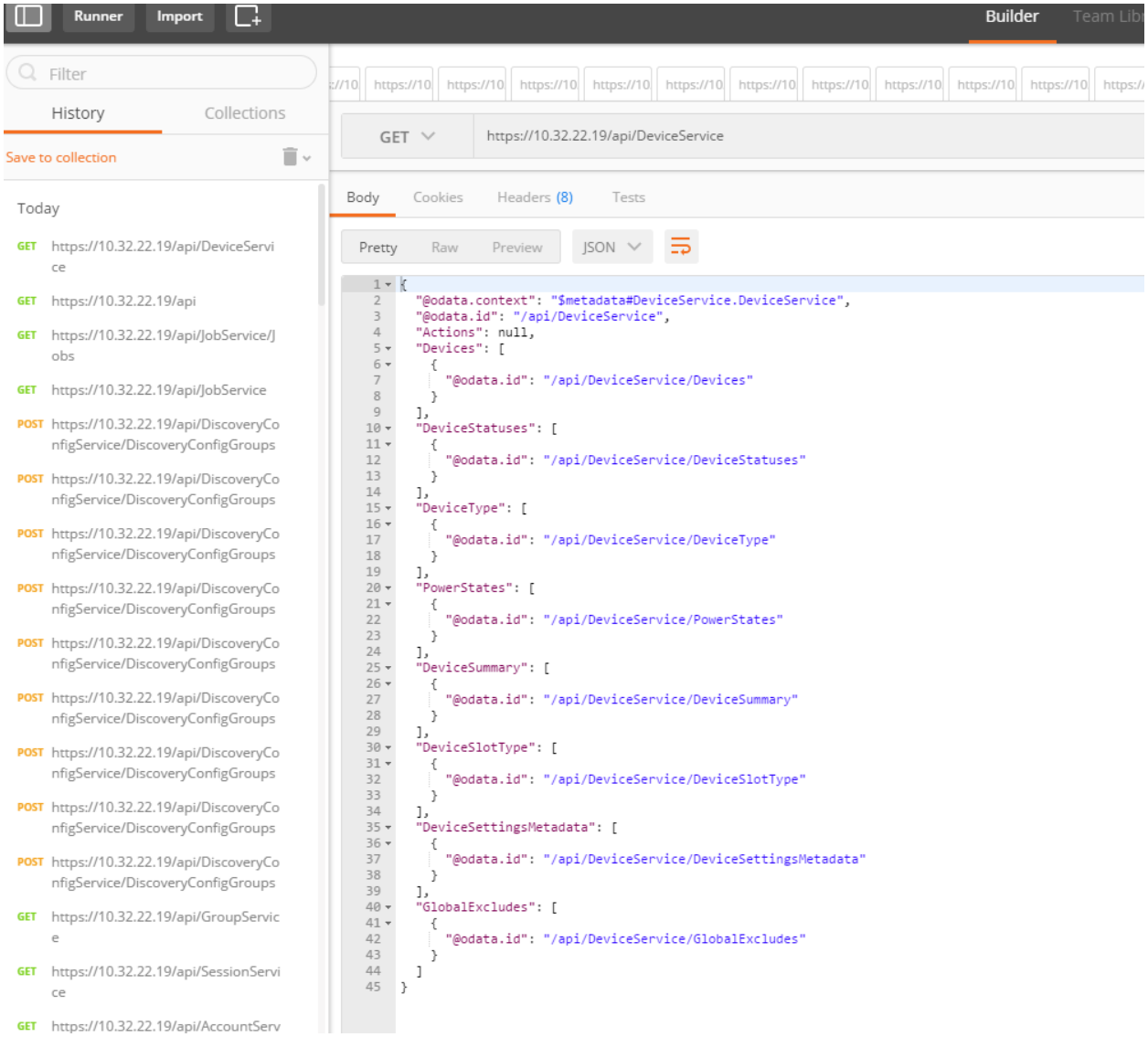


Figure 4 URIs from `DeviceService`

The following screen shot shows that there are 72 devices.

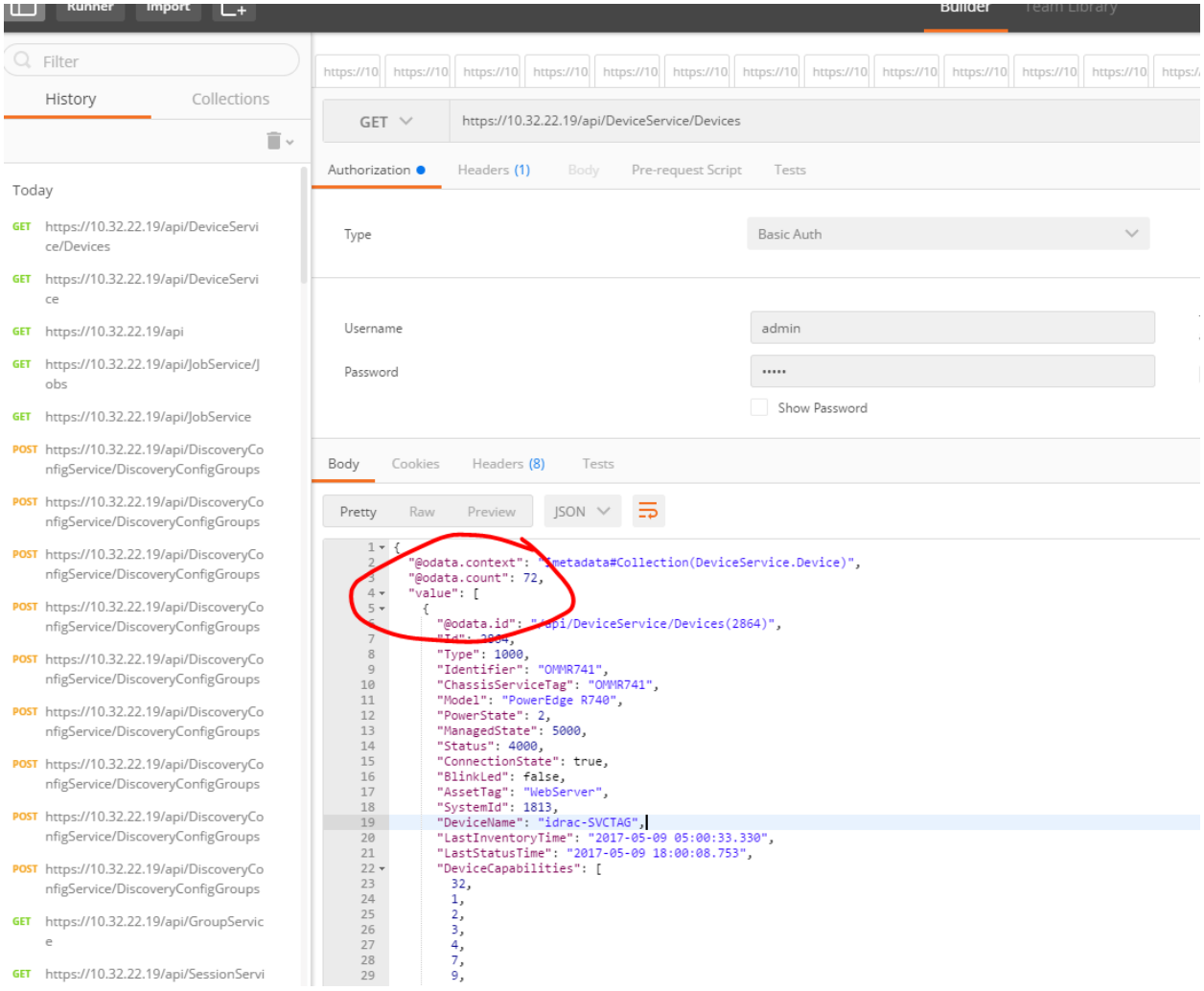


Figure 5 Screen shot showing 72 devices

Note—The following is a simple Python script to list devices.

- One use-case would be to filter the device list by a certain type of devices (say, by Type = Chassis). This can be done using the filter options. `https://IP/api/DeviceService/Devices?$filter=Type eq 2000` would return a list of Chassis devices.
- Another use-case is to figure out which of the devices are powered on. The following query can help in this case. `https://IP/api/DeviceService/Devices?$filter=PowerState eq 2` would return a list of devices that are turned on.

2.3 Retrieving device inventory

The api `https://IP/api/DeviceService/Devices(id)/InventoryDetails` returns the inventory collected for the device, specified by the device ID (id). This Python script retrieves the inventory details of a specific device.

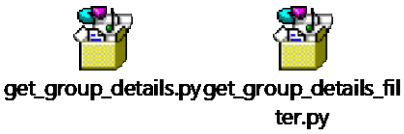
2.4 Get list of groups

OpenManage Enterprise has support for both static and dynamic groups that the user can create. In addition, the appliance has a few pre-canned groups for supported device types / well known classifications. Groups can be listed using the https://IP/api/GroupService/Groups api. This Python script lists the groups in an OpenManage Enterprise appliance.

2.5 Get details of groups and filter groups

What devices are contained in a group can be got using the https://IP/api/GroupService/Groups(id)/Devices API, where the ID specifies the group ID. Users can also quickly filter groups based on name or description.

The following Python scripts list the devices in a group, and also show how to filter a group by name or description.



2.6 Get alerts by device or group

The AlertService in OpenManage Enterprise enables you to see all the alerts received by the appliance and to create or configure alert policies. https://IP/api/AlertService is the URI for the AlertService. Oftentimes, it is useful to filter alerts by those received for a specific device or for a specific group of devices.

The following Python scripts illustrate filtering alerts by device and by group.

2.7 Get list of reports

The most frequently used reports are built-in and available in OpenManage Enterprise. OpenManage Enterprise also enables you to build custom reports, if the built-in reports do not meet your requirements. Both pre-canned and custom reports can be listed by using the ReportService API https://IP/api/ReportService/ReportDefs.

The following Python script lists the reports available in an appliance.


get_report_list.py

2.8 Run a report

A report can be run by using the RunReport api on ReportService. Internally, running a report translates into a job or a task, which the user should poll for completion by checking the job status. When the job completes, several result rows are persisted and are retrievable. The report definition also identifies the columns for the report. The following Python script illustrates running an existing report, and printing out its result rows.


run_existing_report.py

2.9 Update installed firmware for a device

There are two workflows to update the firmware for a device:

- You can create a catalog either from the Online catalog or an offline catalog stored on an NFS or CIFS share. And then associate one or more devices with the catalog thus creating a firmware baseline. The compliance status of a baseline indicates how the devices in that baseline differ in firmware levels from the reference catalog.
- The user may also update the firmware for an individual device by using the UpdateService UploadFile URI.

A PowerShell script to update the installed firmware for a device is provided here.


Update-InstalledFirmware.ps1

2.10 Create Identity Pool

Identity Pools provide sets of values that can be used to assign a unique virtual identity to a device for network operations. Virtual identities are used for stateless computing—for example, for transferring (migrating) virtual identity from one device to another.

An Identity Pool optionally contains several protocol-specific sub-pools—for Ethernet communications, iSCSI operations, FCoE operations, and Fibre Channel (FC) operations. For information about Identity Pools, see the *Advanced Server Configuration in Dell EMC OpenManage Enterprise 3.0* available on the support site.

The following URI and payload are used to create an Identity Pool. Sub-pools that are not required can be omitted or set to null:

HTTP Method: `POST`
URI: `https://<ip-addr>/api/IdentityPoolService/IdentityPools`

Example payload:

The following request is to create an Identity Pool that has a sub-pool for each protocol (Ethernet, iSCSI, FCoE, and FC). The iSCSI section also specifies values for iSCSI Initiator configuration.

```
{
  "Name":"Sample Identity Pool",
  "Description":"This is a sample Identity Pool",
  "EthernetSettings":{
    "Mac": {
      "IdentityCount":55,
      "StartingMacAddress": "UFBQUFAA"
    }
  },
  "IscsiSettings":{
    "Mac": {
      "IdentityCount":65,
      "StartingMacAddress": "YGBgYGAA"
    },
    "InitiatorConfig":{
        "IqnPrefix":"iqn.myprefix."
    },
    "InitiatorIpPoolSettings":{
        "IpRange":"10.35.0.1-10.35.0.255",
        "SubnetMask":"255.255.255.0",
        "Gateway":"10.35.0.1",
        "PrimaryDnsServer":"10.35.0.15",
        "SecondaryDnsServer":"10.35.0.16"
    }
  },
  "FcoeSettings":{
    "Mac": {
      "IdentityCount":75,
      "StartingMacAddress": "cHBwcHAA"
    }
  },
  "FcSettings":{
    "Wwnn": {
      "IdentityCount":85,
      "StartingAddress": "IACAgICAgAA="
    },
    "Wwpn": {
      "IdentityCount":85,
      "StartingAddress": "IAGAgICAgAA="
    }
  }
}
```

The example above includes an iSCSI sub-pool; the “IpRange” string is expressed as a range of IP addresses, with a separate “SubnetMask” value specified. Alternatively, the “IpRange” could have been expressed in CIDR format, and the “SubnetMask” value omitted.

Note that the starting MAC address (“StartingAddress”) for each protocol sub-pool is expressed as a Base64 value. Several encoder/decoder applications are available online. The value “`UFBQUFAA`”, specified above as the starting MAC address for Ethernet communications decodes to 50:50:50:50:50:00 (0x505050505000).

The reply from this URI may indicate a success or error result.

- Is the request is processed successfully, the reply status is “201 Created” and the payload indicates success and provides the ID of the new Identity Pool, as in the following:

```
{
    "Id": 10,
    "IsSuccessful": true,
    "Issues": []
}
```

- If there are one or more issues with the request (such as a duplicate Identity Pool name or overlap between address ranges), an error reply is returned, with a status of “400 Bad Request”. For example, if an Identity Pool already exists with the specified name, a reply payload such as the following is returned:

```
{
    "error": {
        "code": "Base.1.0.GeneralError",
        "message": "A general error has occurred. See ExtendedInfo for more
information.",
        "@Message.ExtendedInfo": [
            {
                "MessageId": "CTEM9028",
                "RelatedProperties": [],
                "Message": "Unable to create or update the Identity Pool because an
Identity Pool with the same name already exists.",
                "MessageArgs": [],
                "Severity": "Warning",
                "Resolution": "Enter a unique Identity Pool name and retry the
operation. For valid names, see the MSM User's Guide available on the support
site."
            }
        ]
    }
}
```

Note that the `ExtendedInfo` element is an array. If there are multiple issues with the contents of the request, a separate entry will be included in the reply for each issue.

2.11 Create Template from Reference Server

A system configuration template may be created several ways:

- By getting the current system configuration from a specific device (referred to as a “Reference Device”)
- By importing a file containing a system configuration (an SCP file)

- By cloning a built-in template provided by OpenManage Enterprise.

This section discusses the first option—Creating a template from a reference device.

A device must be discovered by OpenManage Enterprise before it can be used as a reference device. The following URI and payload are used to create a system configuration template:

HTTP Method: `POST`
URI: `https://<ip-addr>/api/TemplateService/Templates`

Example payload:

The following example creates a new template named “Test Template”, using the device with ID 25010 as the reference device.

```
{
  "Name":"Test Template",
  "Description":"This is a test Template",
  "TypeId":2,
  "ViewTypeId":2,
  "SourceDeviceId":25010,
  "Options":null,
  "Fqdds":"All"
}
```

- The “TypeId” field is not used for template creation and can be ignored.
- The “ViewTypeId” should be 2 to specify a system configuration template for deployment usage.
- The “Fqdds” field provides functionality to copy only certain areas of system configuration from the specified reference server. One or more of the following values may be specified in a comma-separated string:

  `iDRAC, System, BIOS, NIC, LifeCycleController, RAID,` and `EventFilters`

Note—OpenManage Enterprise 3.0 does not include “FC” as an option for this list. In order to get Fibre Channel configuration from a reference server, the “All” string must be specified.

The string “All” specifies to get the entire system configuration from the reference device.

OpenManage Enterprise supports the creation of templates for both servers and chassis. The device type of the specified Source Device ID determines the template type that gets created.

If the Create Template request is successful, the returned status is “201 Created” and the reply contains the ID of the new template, such as the following, indicating that a template was created with an ID of 11:

```
11
```

2.12 Assign Identity Pool and Network Settings to Template

Associating an Identity Pool with a Template tells OpenManage Enterprise to (a) create a virtual identity for each device to which the template is deployed, and (b) get identity values from that Identity Pool when it needs to create a virtual identity for a device (for example, when deploying system configuration to a device).

The payload used to assign an Identity Pool to a template is also used to specify network-specific settings to use when deploying the template to devices. This network information consists of VLAN Network settings and bandwidth settings for partitioned NIC ports.

The following URI and payload are used to create a system configuration template:

HTTP Method: `POST`
URI: `https://<ip-addr>/api/TemplateService/Actions/TemplateService.UpdateNetworkConfig`

Example payload:

The following example assigns the Identity Pool with ID 1 to the template with ID 11:

```
{
  "TemplateId" : 11,
  "IdentityPoolId" : 1
}
```

The preceding payload merely specifies the ID of the Identity Pool for the Template to use. The following example also shows minimum and maximum bandwidth being set for a NIC port specified in the Template:

```
{
  "TemplateId" : 11,
  "IdentityPoolId" : 1,
  "Attributes": [{
        "DeviceId" : null,
      "Attributes": [{
              "Id" : 13143,
              "Value" : 25,
              "IsIgnored" : false
      },{
              "Id" : 13144,
              "Value" : 75,
              "IsIgnored" : false
      }]
  }],
  "VlanAttributes": [{
            "ComponentId" : 19,
            "Untagged" : null,
            "Tagged" : null
  }]
}
```

To remove an existing association between a template and an Identity Pool, specify the applicable template ID, with `IdentityPoolId` specified as zero.

The reply to this request specifies the number of bandwidth attributes that were updated (whether or not any were included in the request). If the request was successful, the result status is “200 OK”. The first version above would return a status of “200 OK” and a reply value of 0; the second version would also have returned a status of “200 OK”, but a reply value 2 (since two attributes were updated).

2.13 Deploy Template

Deploy Template refers to taking the system configuration defined in a template and applying that configuration to one or more target devices. As discussed earlier, when a template is created, it may contain configuration settings for specific functional areas (such as iDRAC, BIOS, and RAID) or a full system configuration. In most cases, deployment involves full system configurations. Each template may be used as often as desired for configuration deployment to target devices.

A device must be discovered by OpenManage Enterprise before it can be specified as a deployment target.

If a Template *does not* have an Identity Pool associated with it, only non-identity attributes will get sent to the specified target devices. Without a virtual identity, stateless functions supported by OpenManage Enterprise (such as identity migration) cannot be executed.

If the template does have an Identity Pool associated with it, then unused identities get reserved from the Identity Pool (as needed) and included in the attributes that get deployed to a target server. A separate set of unique identities is reserved for each specified target device (if the template includes identity attributes).

In OpenManage Enterprise, the Deploy Template wizard presents a UI for setting up a Deploy Template request. Setting up a Deploy Template request is a complex process. The Deploy Template wizard uses several different APIs to support all that needs to be done. It uses the following APIs:

- `Assign Identities`

  This API is only used when the selected template has an Identity Pool associated with it. It reserves (rather than assigns) identities from the template's associated Identity Pool. The types and numbers of identities that need to be reserved depends on the template's BIOS, NIC, and FC configuration. For example, Ethernet MAC addresses are usually required for each NIC port or partition, and certain iSCSI identities (such as MAC address, and Initiator name and IP address) are needed when iSCSI is set up for access or booting.

  Calling the API from within the Deploy Template wizard provides the customer with early feedback regarding identity availability and reservations. Reserved identity values are also displayed in the UI.

- `Get Assigned Identities and Boot Options`

  This API gets all the identity attributes reserved for the current deployment request (e.g., via the Assign Identities API discussed above). And it also returns Boot Options, as discussed here.

  For most non-identity attributes, the same value that is, the value in the template) is sent to every target device. However, when a template configuration specifies that a port is to be used for booting (for example, iSCSI boot is enabled), certain deployment attributes may need to be set differently for different target devices. This set of attributes is referred to as Boot Options.

  The Deploy Template wizard invokes this API to get identity reservations and a list of any boot option attributes applicable to the Template specified in the wizard. Sometimes, there aren't any boot options; it just depends on the Template configuration. When boot options exist, required values may be set separately in the UI for each target device.

If this functionality was not provided in OpenManage Enterprise, you would have to access each target device after deployment completed and customize its boot attributes as required at that point. Being able to do this customization in the Deploy Template wizard is a lot more convenient.

Values set in the wizard for boot option attributes are included in the Deploy Template API (discussed below).

- `Deploy Template`

This API specifies the ID of the template to be deployed, various related instructions (such as whether or not Boot-from-ISO is wanted, and associated values if it is wanted), the ID of each target device, and when to do the deployment. The request may specify that the deployment execute immediately, or at a specified time in the future. The request also includes boot option values for each target device.

Each of the preceding APIs is discussed here:

Assign Identities

The Assign Identities API uses the following URI:

HTTP Method: `POST`
URI: `https://<ip-addr>/api/TemplateService/Actions/TemplateService.AssignIdentities`

The following example payload requests that unused values be reserved from the Identity Pool associated with Template 12, based on the applicable identity attributes contained in that Template. A set of those Identity Pool values is requested for a single target device (with ID 25028):

```
{
  "TemplateId" : 12,
  "BaseEntityIds" : [25028]
}
```

As indicated by the payload structure above, multiple target device IDs could have been specified (in a comma-separated list). OpenManage Enterprise will reserve a unique set of identity values for each specified target device.

The reply indicates whether or not the request succeeded. If it did not succeed, the reply will contain an error entry for each problem that was found.

Get Assigned Identities and Boot Options

The Get Assigned Identities API uses the following URI:

HTTP Method: `POST`
URI: `https://<ip-addr>/api/TemplateService/Actions/TemplateService.GetAssignedIdentities`

The following example payload requests information about identity values reserved for deployment of template 19 for device 6664:

```
{
  "TemplateId" : 19,
  "BaseEntityId" : 6664
}
```

Note that this payload requests assigned identity info for a single target device. When there are multiple target devices, it must be invoked separately for each one.

The reply to this API is a hierarchical structure corresponding to the cards, ports, and partitions which have either reserved identities or boot option attributes.

The two text boxes below show a portion of a reply, for a port which has both reserved identities and boot option attributes. The first text box shows a portion of the reply which specifies the boot option attributes. The second text box shows a portion of the reply for the reserved identities on a port.

```
{
    "Id": 5,
    "Name": "NetModel",
    "Description": "Hierarchy containing NIC and BIOS Network Settings and Boot Options",
    "AttributeGroupNames": [],
    "AttributeGroups": [
        {
            "GroupNameId": 1001,
            "DisplayName": "NICModel",
            "SubAttributeGroups": [
                {
                    "GroupNameId": 3001,
                    "DisplayName": "Fibre Channel in Mezzanine 2C",
                    "SubAttributeGroups": [
                        {
                            "GroupNameId": 1,
                            "DisplayName": "Port",
                            "SubAttributeGroups": [
                              {
                                "GroupNameId": 0,
                                "DisplayName": "BootOptions",
                                "SubAttributeGroups": [],
                                "Attributes": [
                                  {
                                    "AttributeId": 0,
                                    "CustomId": 0,
                                    "AttributeEditInfoId": 0,
                                    "DisplayName": "First Target",
                                    "Description": "Settings for First Target",
                                    "Value": "First Target",
                                    "IsReadOnly": true,
                                    "IsIgnored": false
                                  },
                                  {
                                    "AttributeId": 1338,
                                    "CustomId": 0,
                                    "AttributeEditInfoId": 1968,
                                    "DisplayName": "First FC Target LUN",
                                    "Description": "First FC Target LUN",
                                    "Value": "0",
                                    "IsReadOnly": false,
                                    "IsIgnored": false
                                  },
                                  {
                                    "AttributeId": 1336,
                                    "CustomId": 0,
                                    "AttributeEditInfoId": 1967,
                                    "DisplayName": "First FC Target World Wide Port Name",
                                    "Description": "First FC Target World Wide Port Name",
                                    "Value": "00:00:00:00:00:00:00:00",
                                    "IsReadOnly": false,
                                    "IsIgnored": false
                                  }
                                ]
                              },
```

The preceding shows the Boot Option identities for Port 1 of Fibre Channel in Mezzanine 2C. The first “attribute” (with AttributeId 0) provides a title (“First Taget”) for the attributes in that set of boot options.

The text box below continues where the preceding text box left off, showing reserved identities for the same card and port (since the partition number is zero):


```
                                {
                                    "GroupNameId": 0,
                                    "DisplayName": "Partition",
                                    "SubAttributeGroups": [],
                                    "Attributes": [
                                        {
                                            "AttributeId": 0,
                                            "CustomId": 0,
                                            "AttributeEditInfoId": 0,
                                            "DisplayName": "Protocol",
                                            "Description": "",
                                            "Value": "FC",
                                            "IsReadOnly": true,
                                            "IsIgnored": false
                                        },
                                        {
                                            "AttributeId": 0,
                                            "CustomId": 0,
                                            "AttributeEditInfoId": 0,
                                            "DisplayName": "WWNN",
                                            "Description": "WWNN",
                                            "Value": "20:00:16:16:16:16:16:03",
                                            "IsReadOnly": true,
                                            "IsIgnored": false
                                        },
                                        {
                                            "AttributeId": 0,
                                            "CustomId": 0,
                                            "AttributeEditInfoId": 0,
                                            "DisplayName": "WWPN",
                                            "Description": "WWPN",
                                            "Value": "20:01:16:16:16:16:16:03",
                                            "IsReadOnly": true,
                                            "IsIgnored": false
                                        }
                                    ]
                                }
                            ],
                            "Attributes": []
                        },
                        ...
                    ],
                    "Attributes": []
                },
                ...
            ],
            "Attributes": []
        }
    ],
    "TemplateEditInfo": null
}
```


Figure 6 Response payload continued

The first “attribute” (with AttributeId 0) provides a title (“FC”) for the set of attributes.

The full reply listing includes reserved identities and boot options for every card, port, and partition for the specified device, based on the specified template.

Deploy Template

The Deploy Template API uses the following URI:

HTTP Method: `POST`
URI: `https://<ip-addr>/api/TemplateService/Actions/TemplateService.Deploy`

The following example payload is for a Deploy Template request:

```
{
  "Id" : 13,
  "Schedule" : {
        "RunNow" : true,
        "RunLater" : false,
        "Cron" : null,
        "StartTime" : null,
        "EndTime" : null
  },
  "Attributes" : [ {
        "DeviceId" : 25049,
        "Attributes" : [ {
                "Id" : 14315,
                "Value" : "0.0.0.0",
                "IsIgnored" : true
        }, {
                "Id" : 14317,
                "Value" : "100.100.230.1",
                "IsIgnored" : true
        }, {
                "Id" : 14316,
                "Value" : "255.255.254.0",
                "IsIgnored" : true
        }, {
                "Id" : 14322,
                "Value" : "::",
                "IsIgnored" : true
        } ]
  } ],
  "Options" : {
        "ShutdownType" : 1,
        "TimeToWaitBeforeShutdown" : 300,
        "EndHostPowerState" : 1
  },
  "NetworkBootIsoModel" : {
        "BootToNetwork" : false,
        "ShareType" : "CIFS",
        "ShareDetail" : {
                "IpAddress" : null,
                "ShareName" : null,
                "WorkGroup" : null,
                "User" : null,
                "Password" : null
        },
        "IsoPath" : null
  },
  "TargetIds" : [25049],
  "Hibernate" : null
}
```

The request does not include any reserved identities, as they are already marked as reserved in the database. It does include the following attributes:

- iDRAC Management attributes

  The **iDRAC Management IP** page of the Deploy Template wizard enables you to specify a required IP addressing method. One of the following may be selected: “Don’t change IP settings”, “Set as DHCP”, or “Set static IP for each device”. One or more of the following attributes could be set for iDRAC management, depending on the option chosen:

  ```
  IPv4.1#Enable
  IPv4.1#DHCPEnable
  IPv6.1#Enable
  IPv6.1#AutoConfig
  IPv4Static.1#Address
  IPv4Static.1#Netmask
  IPv4Static.1#Gateway
  IPv6Static.1#Gateway
  ```

  In the example above, the “Don’t change IP settings” option was selected. Therefore, no attribute changes were applicable. The payload includes the iDRAC Management attributes, but sets `IsIgnored` to true for each one, indicating that the attribute should not be deployed to target devices.

- Boot Option attributes and values

In the example above, the template did not have any applicable boot option attributes. Therefore, the attributes shown are all related to iDRAC Management. Because to space limitation, only 3 of those attributes were included.

If the request is successful, the reply status is “200 OK” and the reply value is the ID of the task created to deploy the template to the target devices. The task will execute in the background. The preceding request returned the following reply:

```
25053
```

The task ID can be used to get the task’s status.

A Technical support and resources

Dell.com/support is focused on meeting customer needs with proven services and support.

The *Dell EMC OpenManage Enterprise* and *OpenManage Enterprise-Modular Edition RESTful API Guide* available on the support site is a full reference of all APIs supported by the OpenManage Enterprise appliance.