



## **Cisco Ultra Services Platform Deployment Automation Guide, Release 6.4**

**First Published:** 2018-11-02

**Last Modified:** 2020-07-21

### **Americas Headquarters**

Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, CA 95134-1706  
USA  
<http://www.cisco.com>  
Tel: 408 526-4000  
800 553-NETS (6387)  
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at [www.cisco.com/go/offices](http://www.cisco.com/go/offices).

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/c/en/us/about/legal/trademarks.html>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2018-2020 Cisco Systems, Inc. All rights reserved.



## CONTENTS

---

### PREFACE

<b>About this Guide</b>	<b>xi</b>
Conventions Used	xi
Obtaining Documentation	xii
Contacting Customer Support	xii

---

### CHAPTER 1

<b>Ultra Services Platform (USP) Introduction</b>	<b>1</b>
USP Introduction	1
USP Architecture	2
USP VNF Architecture	3
Ultra Element Manager (UEM)	3
Life Cycle Manager	6
Service Configuration Manager	7
Service Level Agreement Manager	7
Control Function	10
Service Function	11
Network Function	12
Application Function	13
USP VNF Types	14
Ultra Gateway Platform	15
Ultra Service Framework	15
Ultra Automation Services	16
AutoIT	17
AutoDeploy	19
AutoVNF	21
Ultra Web Services	24
USP VNF Component Redundancy and Availability	25

Platform Requirements 25

UEM Redundancy 25

CF Redundancy 26

SF Redundancy 26

NF Redundancy 27

AF Redundancy 27

Ultra Service Component (USC) Redundancy 27

ICSR Support 27

---

**CHAPTER 2**

**USP Installation Prerequisites 29**

Ultra M Deployments Using UAS 29

    Virtual Machine Recommendations 29

    Software Requirements 30

    Hardware Requirements 30

        Server Functions and Quantities 30

    Password Requirements and Login Security 32

VNF Deployments Using AutoVNF 32

    Virtual Machine Recommendations 32

    Software Requirements 33

    Hardware Requirements 34

        Server Functions and Quantities 34

    Network Requirements 35

    Password Requirements and Login Security 37

---

**CHAPTER 3**

**Deploying Hyper-Converged Ultra M Models Using UAS 39**

Virtual Infrastructure Manager Installation Automation 39

    Introduction 39

    VIM Installation Automation Overview 40

    Pre-Virtual Infrastructure Manager Installation Verification 41

    Install the VIM Orchestrator 42

        Install and Configure RHEL 42

        Onboard the USP ISO 48

        Extract the UAS Bundle 50

        Deploy AutoIT 51

Deploy AutoDeploy	54
Prepare the VIM Orchestrator and VIM Configuration File	58
Activate the VIM Orchestrator and VIM Deployment	58
VNF Deployment Automation	60
VNF Deployment Automation Overview	60
Pre-VNF Installation Verification	64
Deploy the USP-based VNF	64
Configure VIM Tenants	64
Configure OpenStack Prerequisites	66
Configure the VNF Rack and the VNF Descriptors	68
Configuring Fully-Defined VM Names for ESC	70

**CHAPTER 4****Deploying VNFs Using AutoVNF 73**

Introduction	73
VNF Deployment Automation Overview	73
Pre-VNF Installation Verification	77
Deploy the USP-based VNF	77
Onboard the USP ISO	78
Extract the UAS Bundle	79
Extract the UEM VM Image	80
Extract the UGP VM Image	81
Upload the USP VM Images to Glance	82
Deploy the AutoVNF VM	82
Activate the AutoVNF Configuration Files	85
Upgrading/Redeploying the Stand-alone AutoVNF VM Instance	87

**CHAPTER 5****Upgrading the Ultra Services Platform Deployment 89**

Upgrading/Redeploying USP for Hyper-Converged Ultra M Models	89
Upgrading/Redeploying VNFs Deployed Through a Stand-alone AutoVNF Instance	92

**CHAPTER 6****Post Deployment Operations 95**

Deactivating the USP Deployment	95
Terminating the AutoDeploy VM	96
Terminating the AutoIT VM	96

Deploy and Undeploy the Card with the NCS CLI	97
Monitoring and Troubleshooting the Deployment	100
Pre-Deactivation/Post-Activation Health Check Summary	100
Checking OSP-D Server Health	100
Viewing Stack Status	100
Viewing the Bare Metal Node List	101
Viewing the OpenStack Server List	102
Viewing the OpenStack Stack Resource List	103
Verifying Node Reachability	103
Verify NTP is running	104
Checking OSP-D Server Health	105
Verifying VM and Other Service Status and Quotas	105
Checking Cinder Type	107
Checking Core Project (Tenant) and User Core	108
Checking Nova/Neutron Security Groups	110
Checking Tenant Project Default Quotas	115
Checking the Nova Hypervisor List	116
Checking the Router Main Configuration	117
Checking the External Network Using the core-project-id	118
Checking the Staging Network Configuration	120
Checking the DI-Internal and Service Network Configurations	121
Checking the Flavor List	122
Checking Host Aggregate and Availability Zone Configuration	122
Checking Controller Server Health	123
Checking the Pacemaker Cluster Stack (PCS) Status	123
Checking Ceph Storage Status	124
Checking Controller Node Services	125
Check the RabbitMQ Database Status	126
Checking OSD Compute Server Health	127
Checking Ceph Status	127
Checking OSD Compute Node Services	127
Monitoring AutoDeploy Operations	127
Viewing AutoDeploy Logs	128
AutoDeploy Transaction Logs	128

Checking AutoDeploy Processes	131
Determining the Running AutoDeploy Version	131
Monitoring AutoIT Operations	132
Viewing AutoIT Logs	132
Viewing AutoIT Operational Data	136
Checking AutoIT Processes	136
Monitoring AutoVNF Operations	137
Viewing AutoVNF Logs	137
General AutoVNF Logs	137
AutoVNF Transaction Logs	138
AutoVNF Event Logs	138
Viewing AutoVNF Operational Data	143
Example show confd-state Command Output	145
Example show confd-state ha Command Output	145
Example show log Command Output	145
Example show running-config Command Output	145
Example show uas Command Output	146
Example show vnfr Command Output	148
Example show vnf-packager Command Output	152
UAS Log Collection	153
Feature Description	154
Limitations	154
Collecting the UAS Logs	154
Via the ConfD CLI Command	154
Via the Standalone Script	155
Collecting VNF Diagnostic Information	158
Sample Logs	158
Monitoring VNFM Operations	160
Viewing ESC Status	160
Monitoring Status Through the ESC Command Line	160
Monitoring Status Through an AutoVNF API	160
Viewing ESC Health	161
Viewing ESC Logs	162
ESC Logs	162

- ESC YANG Logs 163
- Monitoring VNF Operations 164
  - Viewing UEM Service Status 164
  - Viewing UEM Logs 164
  - Viewing UEM Zookeeper Logs 165
  - Viewing VNF Information through the Control Function 166
- Monitoring and Recovering AutoVNF Through AutoIT 167
- Monitoring and Recovering VNFC Through AutoVNF 168
- Troubleshooting Deactivation Process and Issues 171
  - Deactivation Fails Due to Communication Errors with AutoVNF 171
  - Deactivation Fails Because AutoDeploy Generates an Exception 171
  - Deactivation Fails Because of AutoVNF-VNFM Communication Issues 173
  - Deactivation Fails Because of Issue at VNFM 173
  - Deactivation Fails Because AutoVNF Generates an Exception 174
- Troubleshooting UEM Issues 175
  - UEM VM Stuck in a Boot Loop 175

---

**APPENDIX A**     [boot\\_uas.py Help](#) 179

---

**APPENDIX B**     [Sample VIM Orchestrator and VIM Configuration File](#) 183

---

**APPENDIX C**     [Sample Tenant Configuration File](#) 187

---

**APPENDIX D**     [Sample VNF Rack and VNF Descriptor Configuration File](#) 199

---

**APPENDIX E**     [Sample system.cfg File](#) 211

---

**APPENDIX F**     [Sample ESC VIM Connector Configuration](#) 213

---

**APPENDIX G**     [Sample AutoVNF VNFM Configuration File](#) 215

---

**APPENDIX H**     [Sample AutoVNF VNF Configuration File](#) 217

---

**APPENDIX I**     [USP KPI Descriptions](#) 223  
                           USP KPI Descriptions 223

---

<b>APPENDIX J</b>	<b>Backing Up Deployment Information</b>	<b>225</b>
	Overview	225
	Identify Component IP Addresses	225
	Backup Configuration Files	228
	Backup UAS ConfD Databases	229
	Collect Logs	230
	Collect Charging Detail Records	230

---

<b>APPENDIX K</b>	<b>Example RedHat Network Interface and Bridge Configuration Files</b>	<b>233</b>
	/etc/sysconfig/network-scripts/ifcfg-eno2	233
	/etc/sysconfig/network-scripts/ifcfg-eno1	233
	/etc/sysconfig/network-scripts/ifcfg-br-ex	234
	/etc/sysconfig/network-scripts/ifcfg-br-ctlplane	234

---

<b>APPENDIX L</b>	<b>Patch Upgrade Procedure</b>	<b>235</b>
	VNFM Upgrade Operations	235
	Limitations	235
	Feature Description	236
	VNFM Upgrade Workflow	236
	Initiating the VNFM Upgrade	238
	Via the CLI	238
	Via the NETCONF API	239
	UAS Upgrade Operations	241
	Feature Description (AutoDeploy and AutoIT)	241
	AutoDeploy and AutoIT Upgrade Workflow	242
	Upgrading AutoDeploy or AutoIT	242
	Limitations	243
	Feature Description (AutoVNF)	243
	AutoVNF Upgrade Workflow	244
	Initiating the AutoVNF Upgrade	245
	Via the CLI	245
	Via the NETCONF API	246
	UEM Upgrade Operations	248

- Limitations 248
- Feature Description 249
- UEM Upgrade Workflow 249
- Initiating the UEM Patch Upgrade 253
  - Via the CLI 253
  - Via the NETCONF API 253



## About this Guide

This preface describes the *Cisco Ultra Services Platform Deployment Automation Guide*, how it is organized, and its document conventions.

The Ultra Services Platform (USP) is a 5G-ready virtual mobility network platform that provides a robust and highly scalable architecture that can quickly deploy mobility services across a distributed network in a virtualized environment.

The USP is a complex Virtual Network Function (VNF) conforming to the European Telecommunications Standards Institute (ETSI) Network Function Virtualization (NFV) and NFV Management and Orchestration (MANO) specifications.

- [Conventions Used, on page xi](#)
- [Obtaining Documentation, on page xii](#)
- [Contacting Customer Support, on page xii](#)

## Conventions Used

The following tables describe the conventions used throughout this documentation.

Notice Type	Description
Information Note	Provides information about important features or instructions.
Caution	Alerts you of potential damage to a program, device, or system.
Warning	Alerts you of potential personal injury or fatality. May also alert you of potential electrical hazards.

Typeface Conventions	Description
Text represented as a screen display	This typeface represents displays that appear on your terminal screen, for example:  Login :

Typeface Conventions	Description
Text represented as <b>commands</b>	This typeface represents commands that you enter, for example: <b>show ip access-list</b> This document always gives the full form of a command in lowercase letters. Commands are not case sensitive.
Text represented as a <b>command variable</b>	This typeface represents a variable that is part of a command, for example: <b>show card slot_number</b> <i>slot_number</i> is a variable representing the desired chassis slot number.
Text represented as menu or sub-menu names	This typeface represents menus and sub-menus that you access within a software application, for example: Click the <b>File</b> menu, then click <b>New</b>

## Obtaining Documentation

### Nephelo Documentation

The most current Nephelo documentation is available on the following website: [http://nephelo.cisco.com/page\\_vPC.html](http://nephelo.cisco.com/page_vPC.html)

### StarOS Documentation

The most current Cisco documentation is available on the following website: <http://www.cisco.com/cisco/web/psa/default.html>

Use the following path selections to access the StarOS documentation:

Products > Wireless > Mobile Internet > Platforms > Cisco ASR 5000 Series > Configure > Configuration Guides

## Contacting Customer Support

Use the information in this section to contact customer support.

Refer to the support area of <http://www.cisco.com> for up-to-date product documentation or to submit a service request. A valid username and password are required to access this site. Please contact your Cisco sales or service representative for additional information.



# CHAPTER 1

## Ultra Services Platform (USP) Introduction

---

- [USP Introduction, on page 1](#)
- [USP Architecture, on page 2](#)
- [USP VNF Architecture, on page 3](#)
- [Ultra Automation Services, on page 16](#)
- [Ultra Web Services, on page 24](#)
- [USP VNF Component Redundancy and Availability, on page 25](#)

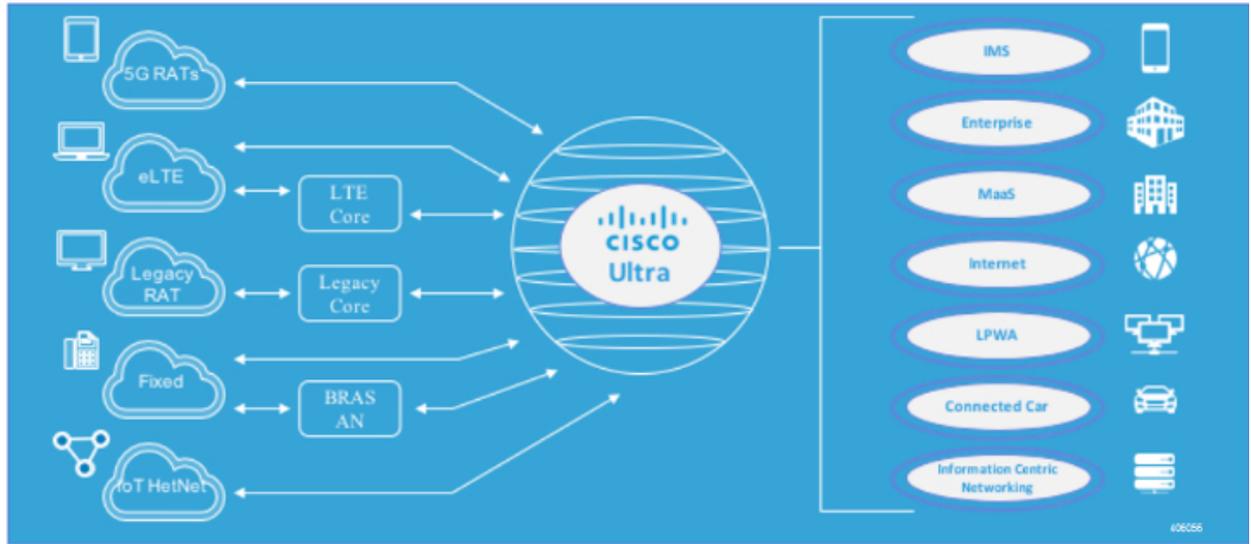
### USP Introduction

The Ultra Services Platform (USP) is a 5G-ready virtual mobility network platform that provides a robust and highly scalable architecture that can quickly deploy mobility services across a distributed network in a virtualized environment. 5G will support countless emerging use cases with a variety of applications that drive significant variability in their performance attributes. From delay-sensitive mobile video applications to infrequent connectivity for simple devices, the diversity of use cases will demand substantially increased throughput, lower latency, ultra-high reliability with substantially higher connection densities.

The USP is a complex Virtual Network Function (VNF) conforming to the European Telecommunications Standards Institute (ETSI) Network Function Virtualization (NFV) and NFV Management and Orchestration (MANO) specifications. Unlike simple VNFs constrained to a single Virtual Machine (VM), the USP is a complex VNF comprised of multiple VNF Components (VNFCs) with a variable number of VMs depending on feature optioning and desired performance specifications.

Leveraging these virtualization, automation and orchestration technologies, the USP enables a NFV architecture that allows VNFs to be “sliced” into smaller, customizable end-to-end instances capable of seamless scaling regardless of the use case. The flexibility brings network providers to true Mobility-as-a-Service (MaaS) offering.

Figure 1: USP Network Slicing



## USP Architecture

The USP solution comprises the following components:

- **Ultra Service Platform VNF:** The USP couples a Virtual Network Function Element Manager (VNF-EM) and multiple VNF components (VNFCs) into a single complex VNF. This coupling conforms to the European Telecommunications Standards Institute (ETSI) NFV Management and Orchestration (NFV MANO) standard and greatly simplifies MANO operations. A separate web-based customer portal called the Ultra Web Service (UWS) is supported in conjunction with the USP VNF and other network elements to simplify the deployment and management of the VNF.
- **Ultra Web Services (UWS):** The UWS provides an environment to graphically construct the USP VNF by allowing a user to select which VNF components are present and enter the necessary deployment parameters needed to instantiate the solution. Once this composition process is complete, the UWS passes the configuration to Ultra Automation Services which generates an ETSI NFV-compliant VNF Descriptor (VNFD). The VNFD is then on-boarded into an NFV Orchestrator (NFVO).



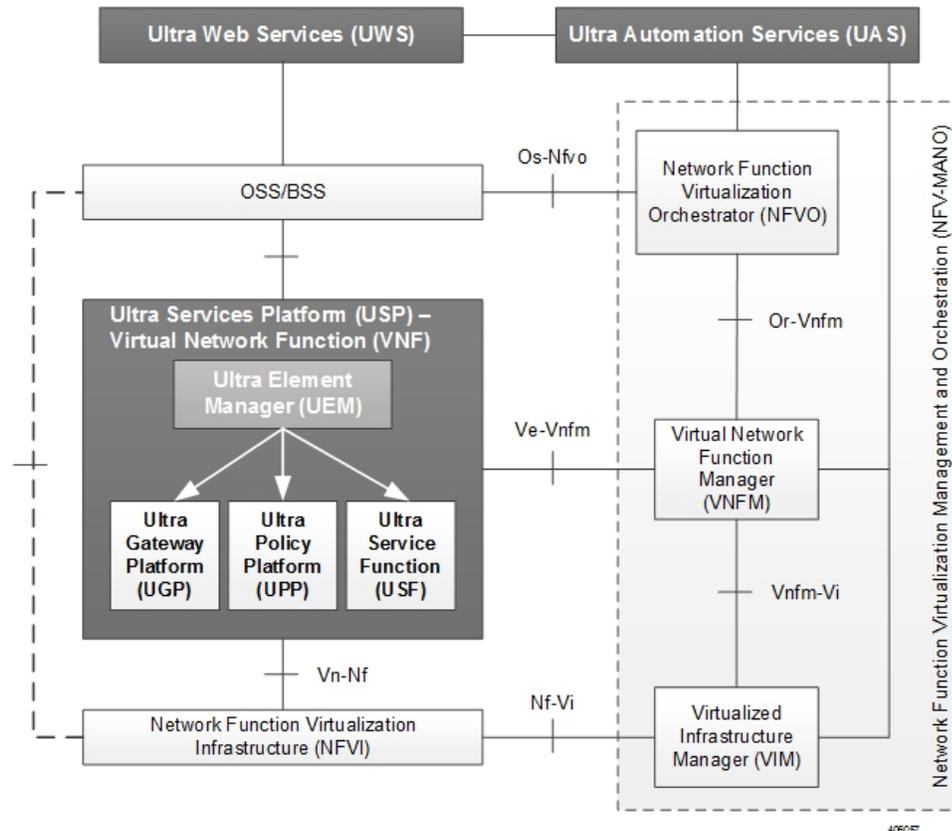

---

**Important** UWS is not supported in 6.x releases.

---

- **Ultra Automation Services (UAS):** UAS provides a suite of automation tools that simplify the on-boarding process of the USP VNF into any Cisco or third-party NFV infrastructure (NFVI).

Figure 2: USP Solution Components in the ETSI MANO Network



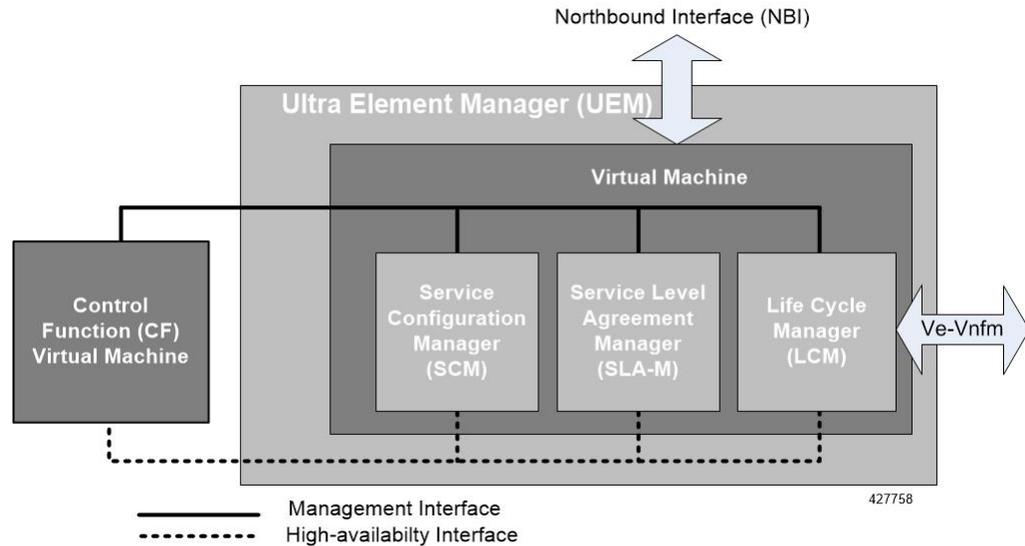
## USP VNF Architecture

This section provides information on the VNF components (VNFCs) that comprise the USP architecture.

### Ultra Element Manager (UEM)

The UEM manages all the major components of the USP architecture. Conforming to ETSI MANO, the UEM is modeled as the element management system (EMS) for the USP which is a complex VNF comprised of multiple VNFCs. The UEM and the complex VNF are represented to the Management and Orchestration (MANO) infrastructure through their own VNF descriptors (VNFDs).

Figure 3: Ultra Element Manager Composition



Although comprised of multiple modules, the UEM provides a single northbound interface (NBI) to external elements such as the OSS/BSS and Ultra Web Service (UWS).

The UEM provides the following network management functions:

- Configuration
- Fault management
- Usage accounting
- Performance measurement
- Security management
- Operational state of VNF

The northbound interface exposes all the information collected, aggregated and exposed through an API interface.

- All the interactions with entities northbound of the UEM happen via a single programmable API interface (e.g. REST, NETCONF, SNMP, etc.) for the purpose of collecting:
  - Configuration data for platform services and for Day-N configuration of its various components
  - Operational data pertaining to the system such as topology (VDU creation and organization) and different levels of VDU and service liveness and KPIs based on the topology
  - Event streams (NETCONF notifications) that are used by the UEM to asynchronously notify northbound entities
  - Remote Procedure Calls (RPCs) used to expose some of the functionalities offered by the platform or its components such as packet tracing or mirroring
  - Asynchronous notifications: When an event that is relevant to northbound, is received from southbound, the SCM relays the event via a Netconf notification

These functions are provided via several different modules that comprise the UEM:

- **Lifecycle Manager (LCM):** The LCM exposes a single and common interface to the VNFM (Ve-Vnfm) that is used for performing life-cycle management procedures on a VNF. As a component within the UEM, it supports the various middleware application programming interfaces (APIs) required to interact with VNF and its components. Refer to [Life Cycle Manager, on page 6](#) for more information.
- **Service Configuration Manager (SCM):** Leverages a YANG-based information model for configuration to provide configuration information to the VNFC Control Function (CF) VMs and other third-party components. It performs this functionality via NETCONF interfaces using pre-configured templates/network element drivers (NEDs). Configuration information is stored in the configuration database (CDB) and passed to the CF VM over the configuration interface via ConfD. Refer to [Service Configuration Manager, on page 7](#) for more information.
- **Service Level Agreement Manager (SLA-M):** Provides timely access to information such as key performance indicators (KPIs), serviceability events, and diagnostic and troubleshooting information pertaining to components within the USP VNF instance such as:
  - The Lifecycle Manager
  - The Control Function (CF)
  - VMs that are part of the VNFCs
  - Any 3rd party applications related to USF service chains (depending on the VNFC)

The SLA-M passes the information it collects over the northbound interface of the UEM. Refer to [Service Level Agreement Manager , on page 7](#) for more information.

Based on the StarOS, the CF is a central sub-system of the VNF that interacts with other sub-systems like service functions (SFs), network functions (NFs), and Application Functions (AFs) using field-tested software tasks that provide robust operation, scalability, and availability. It is equipped with a corresponding CDB for storing configuration information provided by the SCM via ConfD and/or CLI over the management interface. Refer to [Control Function, on page 10](#) for more information.




---

**Important** In 6.3 and later releases, the UEM can be deployed as a separate VNF. In such a deployment, each UEM can deploy and communicate with multiple CFs.

---

High-availability (HA) is ensured across all of these components by the UEM-HA framework via a light-weight protocol that monitors the CF and SLA-M over the High-availability interface. All components are deployed redundantly. In the event of an issue, functions will be switched-over to the standby host. The SLA-M also uses the NETCONF interface to pull KPIs and event/log information from the CF.

[Table 1: TCP/UDP Services and Open Ports for UEM, on page 5](#) lists the TCP/UDP services and the corresponding open ports for UEM.

**Table 1: TCP/UDP Services and Open Ports for UEM**

Port	Service
67	DHCP
68	DHCP

Port	Service
22	SSH
162	SNMP
830	NETCONF
2022	NETCONF
2024	NETCONF
2888	ZOOKEEPER
2889	ZOOKEEPER
2890	ZOOKEEPER
3888	ZOOKEEPER
3889	ZOOKEEPER
3890	ZOOKEEPER
4570	NETCONF
8888	NETCONF
2181	ZOOKEEPER

## Life Cycle Manager

The Life Cycle Manager (LCM) is the UEM component that adapts an USP VNF to an external VNFM. The UEM provides a generic API to manage software, compute, and networking resources. When a VNFM brings up a new USP VNF, the VNFM starts redundant UEM VDUs. The VNFM also provides an initial set of VDUs as specified in the catalog for other USP virtual resources (for example, USP CF or USP SF). As the system initializes, the VNF components can bring VDUs online or offline using the UEM as a proxy to the external VNFM. The UEM provides a generic API to the other USP components, and a set of UEM adapters that attune the UEM to variety of external VNFMs.



### Important

The Cisco Elastic Services Controller (ESC) is the only supported VNFM in this USP release.

The LCM performs life-cycle management procedures on a VNF through a single and common interface to the VNFM. It can communicate with any off-the-shelf VNFM for resource allocation, configuration, monitoring, and lifecycle event updates. The LCM provides a common API to handle all VNFM instantiation flow requests for USP VNFs. It also communicates with a StarOS agent to provide all service and application level monitoring and lifecycle management.

The LCM provides the following functions:

- VNF registration through the onboarding of a virtualized network function descriptor (VNFD) by the VNFM

- Day-0 VNF configuration
- Handling key performance indicator (KPI) data in real-time
- Handling life-cycle events from VNFCs
- VNF termination

Communication between the Life Cycle Manager (LCM) and the VNFM is made possible through the integration of adapters that support VNFM products from multiple vendors. As an UEM component, the LCM includes middleware APIs that support the interface with SLA-M. The APIs are used to monitor KPIs pertaining to VNFC health and VM resource usage (for example, CPU, memory, etc.). APIs that support VNFC configuration establish interfaces to the CF via both the Management and High-availability buses to:

- Provision VMs based on information contained in virtualization descriptor units (VDUs) within the VNFD and associate the VMs to the internal network
- Add and initialize VMs as needed
- Request VNF infrastructure characteristics (for example, topology, deployment policies, etc.)
- Request VNF termination, migration, or destruction
- Request Day-N configuration for a specific VNFC
- Create and associate network ports to VDUs
- Provision networking configurations
- Provide life-cycle event notifications such as service status, configuration status, and HA events
- Provide an interface for determining NFVI information associated with the VDUs

## Service Configuration Manager

The Service Configuration Manager (SCM) provides configuration information to the VNFC Control Function (CF) VMs and other third-party components. It performs this functionality via NETCONF interfaces using pre-configured templates/network element drivers (NEDs). Configuration information is stored in the configuration database (CDB) and passed to the CF VM over the management bus via ConfD data models.

During the initial VNF instantiation process, the SCM component will perform the initial detailed configuration of each VNF Component (gateway, in-line service function, etc.). This process is known as a Day-1 configuration. Additionally, when a change to any of the detailed configuration parameters of any of the VNF components after the VNF has already been deployed, the SCM will modify the specific parts of a detailed service configuration for any of the VNF Components. This is known as a Day-N configuration.

## Service Level Agreement Manager

The Service Level Agreement Manager (SLA-M) provides timely access to information such as key performance indicators (KPIs), serviceability events, and diagnostic and troubleshooting information pertaining to components within the USP VNF instance including:

- The Life Cycle Manager (LCM)
- The Control Function (CF)
- VMs that are part of the VNFCs

- Any 3rd party applications related to USF service chains (depending on the VNFC)

This component is responsible for translating the requests from the Northbound layer into requests to the Southbound layer as well as for receiving and processing events and information from the Southbound layer to offer into aggregated form to the Northbound layer. It also populates a data store to maintain and expose historical data.

This component implements the following functionalities according to the way data are exposed northbound:

- **Immediate Access:** Requests coming from northbound (for example, access to the operational state of a particular VDU) are translated into a southbound request (for example, accessing the VDU operational state in a data source).
- **Historical Access:** The history of data or events in a store are maintained for later retrieval. SLA-M uses NCS's CDB for this purpose. The MA-API session is initiated with NCS and the SLA-M proactively fills the operational data corresponding to historical data whenever it is collected (via periodic polling or notifications). In this scenario, access from northbound takes place by retrieving data directly from CDB instead of invoking a callback registered previously since no callback would have been registered for such data.
- **Aggregated Access:** In this case SLA-M retrieves the “non-aggregated” data from the data sources and then applies aggregation logic using the topology information exposed in the northbound model. When the callback corresponding to the aggregated access is invoked, the SLA-M accesses the northbound operational data describing the topology via MA-API, and performs the needed aggregation of the retrieved data.

## KPIs

Each unit of the system is monitored through a set of KPIs. KPIs are quantities that evolve over time. The SLA-M provides northbound entities with mechanism for accessing a current snapshot of such quantities (instantaneous KPIs) in aggregated or non-aggregated form. In addition, it keeps a history of a user-set number of the most recent KPI samples.

Refer to [USP KPI Descriptions, on page 223](#) for a listing and description of KPIs supported in this release.

Two kinds of KPIs are collected:

- Basic (non-aggregated) KPIs
- Aggregated KPIs

### Basic (non-aggregated) KPIs:

These are performance indicators at the VDU level which are provided to the SLA-M by either the CF or the VFNM Proxy Function.

The LCM provides all basic KPIs coming from the NFVI/VIM (for example, host/guest CPU load, memory, etc.), while the CF provides all other basic KPIs such as application specific metrics and process level information.

The following non-aggregate KPIs are provided by the CF to the SLA-M:

- Performance KPIs for each constituent VDR (*/vnfrs/vnfr/deployment-flavor-record/element-group-records/element-group-record/constituent-vdrs/constituent-vdr/performance-stats*).
- The contribution of the Performance KPIs for each constituent VDR to a specific Network Path (*/vnfrs/vnfr/deployment-flavor-record/element-group-records/element-group-record/service-function-chain*).

-  
*records/service-function-chain-record/network-fwd-path-records/network-fwd-path-record/vdr-stats/vdr-stat).*

- Flow Cache KPIs for each constituent VDR (*/vnfrs/vnfr/deployment-flavor-record/element-group-records/element-group-record/constituent-vdrs/constituent-vdr/flow-cache-stats*).

The following non-aggregate KPIs are provided by the VNFM-proxy to the SLA-M:

- NFVI KPIs for each constituent VDR (*/vnfrs/vnfr/deployment-flavor-record/element-group-records/element-group-record/constituent-vdrs/constituent-vdr/nfvi-stats*). These are exposed by the LCM to the UEM and the UEM mirrors them northbound.

### Aggregated KPIs:

These are indicators derived by SLA-M from the basic KPIs and that reflect the performance of a group of VDUs.

The SLA-M builds aggregated KPIs at different levels of the grouping hierarchy by leveraging topology information. A typical example is building network throughput at the service chain level or slice level or system level. Note that while the SLA-M has the responsibility to build the aggregated KPI, it relies on other components to get the topology that drive such aggregation.

Starting from the non-aggregate KPIs described above, the SLA-M builds the following aggregated KPIs:

- Performance KPIs aggregated at:
  - Network Path (*/vnfrs/vnfr/deployment-flavor-record/element-group-records/element-group-record/service-function-chain-records/service-function-chain-record/network-fwd-path-records/network-fwd-path-record/performance-stats*)
  - Service Function Chain (*/vnfrs/vnfr/deployment-flavor-record/element-group-records/element-group-record/service-function-chain-records/service-function-chain-record/performance-stats*)
  - Element Group (*/vnfrs/vnfr/deployment-flavor-record/element-group-records/element-group-record/performance-stats*)
  - Vnf (*/vnfrs/vnfr/performance-stats*)
  - Vnf for specific Service Function Chain (i.e. Performance-stats for a given service-function-chain across all the element-groups) (*/vnfrs/vnfr/service-function-chain-records/service-function-chain-record/performance-stats*)
- Flow Cache KPIs aggregated at:
  - VNF (*/vnfrs/vnfr/flow-cache-stats*)
- NFVI KPIs aggregated at:
  - Element group (*/vnfrs/vnfr/deployment-flavor-record/element-group-records/element-group-record/nfvi-stats*)
  - VNF (*/vnfrs/vnfr/nfvi-stats*)

## Control Function

The Control Function (CF) is a StarOS based central sub-system of the VNF. It interacts with other sub-systems such as service functions (SFs), network functions (NFs), and Application Functions (AFs), and uses field-tested software tasks that provide robust operation, scalability, and availability. The VNFD and VNFR are equipped with a corresponding configuration database (CDB) for storing configuration information provided by the SCM via ConfD and/or CLI NEDs over the management interface.

The CF also communicates over the High-availability (HA) interface for communicating with the LCM and to provide KPIs and event logs to the SLA-M.

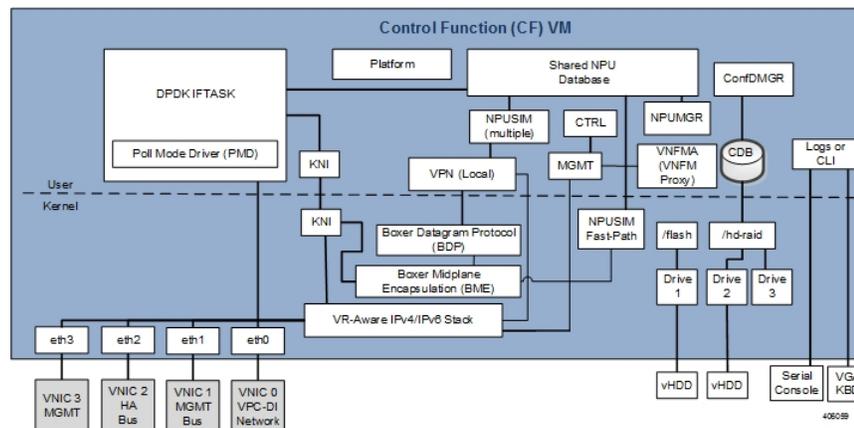
Two CF VMs act as an active:standby (1:1) redundant pair. Within the StarOS, each CF VM is viewed as a virtual card and is responsible for the following functions:

- Hosting Controller tasks
- Hosting the Local context VPNMGR
- Hosting Local context (MGMT) and DI-Network vNICs
- Managing System boot image and configuration storage on vHDD
- Facilitating record storage on vHDD
- Providing Out-of-Band (OOB) management (vSerial and vKVM) for CLI and logging
- Working with the LCM to:
  - Bring VDUs online or offline during system initialization, request more VDUs for scale-out, return VDUs for scale-in lifecycle operations using VPD
  - Facilitate VDU internal management and configuration using predefined artifacts
- Providing KPI, event, and log information to the SLA-M as requested/needed



**Note** Refer to the [Life Cycle Manager, on page 6](#) section for more information.

**Figure 4: CF VM**



**Important**

The Intel Data Plane Development Kit (DPDK) Internal Forwarder task (IFTASK) is used to enhance USP system performance. It is required for system operation. Upon CF instantiation, DPDK allocates a certain proportion of the CPU cores to IFTASK depending on the total number of CPU cores.

## Service Function

Service Function (SF) VMs provide service context (user I/O ports) and handle protocol signaling and session processing tasks. A UGP instance can have a maximum of 14 SF VMs, of which a maximum of 12 SF VMs can be active. See the *Cisco UGP System Administration Guide*.

Each SF VM dynamically takes on one of three roles as directed by the CF:

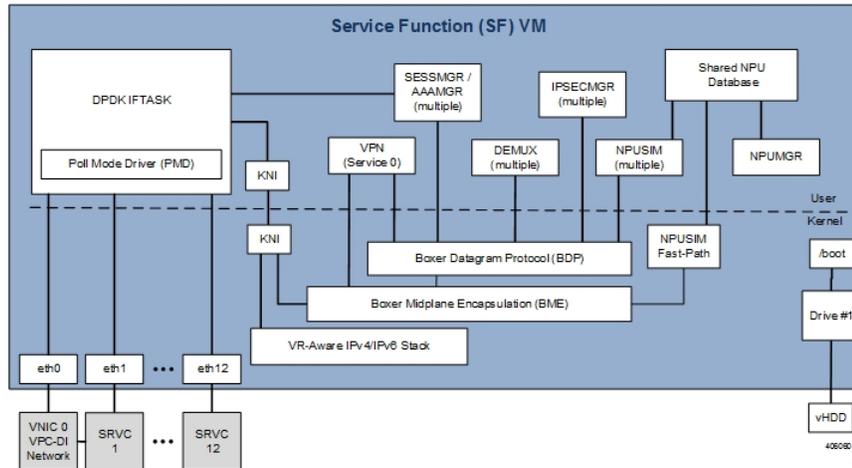
- Demux VM (flow assignments)
- Session VM (traffic handling)
- Standby VM (n+1 redundancy)

An SF provides the following functions:

Function Location	Runs on
NPUSIM fastpath/slow path (NPU emulation and routing to CPU)	Demux VM, Session VM, Standby VM
IFTASK based on the Intel® Data Plane Development Kit (DPDK)	Demux VM, Session VM, Standby VM
Non-local context (SRVC) vNIC ports	Demux VM, Session VM, Standby VM
VPNMGR and Demux for service contexts (first VM)	Demux VM
SESSMGR and AAAMGR for session processing (additional VMs)	Session VM
Egress forwarding decisions	
Crypto processing	

The minimum configuration for an Ultra Gateway Platform instance requires four SFs: two active, one demux, and one standby.

Figure 5: SF VM



**Note** The Intel Data Plane Development Kit (DPDK) Internal Forwarder task (IFTASK) is used to enhance USP system performance. It is required for system operation. Upon CF instantiation, DPDK allocates a certain proportion of the CPU cores to IFTASK depending on the total number of CPU cores.

When deployed in support of the Ultra Services Framework (USF), the SF facilitates the StarOS software tasks pertaining to the IP Services Gateway (IPSG) traffic detection function (TDF). The IPSG receives subscriber policy information from the Policy and Charging Rules Function (PCRF) over the Gx/Gx+ interface. It uses this policy information to steer subscriber session traffic received over the Gi/SGi interface through the SFC as required.

## Network Function

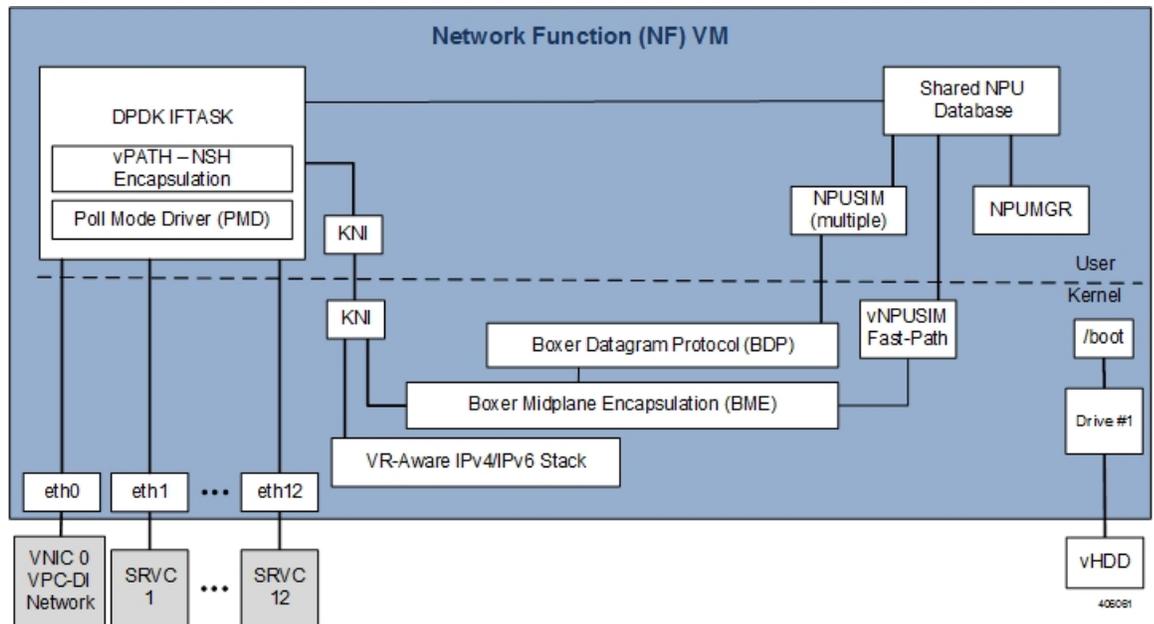
The Network Function (NF) is a virtual machine that is dedicated as a networking adapter between a DI system and external routers. The NF can be used to aggregate the VNF external connection points to a consolidated set of external interfaces. NF virtual machines are typically used for larger DI systems to limit the number of external interfaces to those present on a smaller set of virtual machines. The NF facilitates the building of large scale, high performance systems by providing the virtual equivalent of specialized Network Processing Unit (NPU) hardware.

The NF provides the following functions:

- Serves as a dedicated system for performing high speed traffic classification and flow/counter aggregation based on policies (n-tuple; each NF has access to complete set of policies)
- Limits the number of external interfaces required by aggregating external connection points to a consolidated set of high speed interfaces
- Operates as networking adapter between USP VNFs and external routers
- Subscriber awareness and stickiness as part of flow classification.
- Traffic classification and load balancing

The NF deploys a FAST-PATH architecture leveraging the NPU Manager and NPU SIM software tasks to ensure performance and scalability.

Figure 6: NF VM



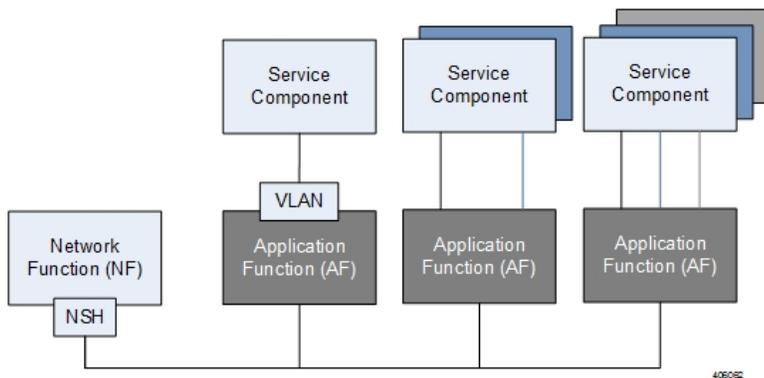
The mobility/DPDK internal forwarder (IF) is the core functional block for the USP architecture. It runs NPUSIM with DPDK into NF. The main functions of the mobility forwarder are:

- Performing the flow classification for each incoming packet, based on pre-configured rules.
- Deriving the service chain that needs to be associated with a flow
- Maintaining the subscriber stickiness - Meaning all the flows of a subscriber should land on the same service path (service path maps to AF).
- Performing the NSH encapsulation/ decapsulation. It uses NSH for communicating the service chain information across the nodes.

## Application Function

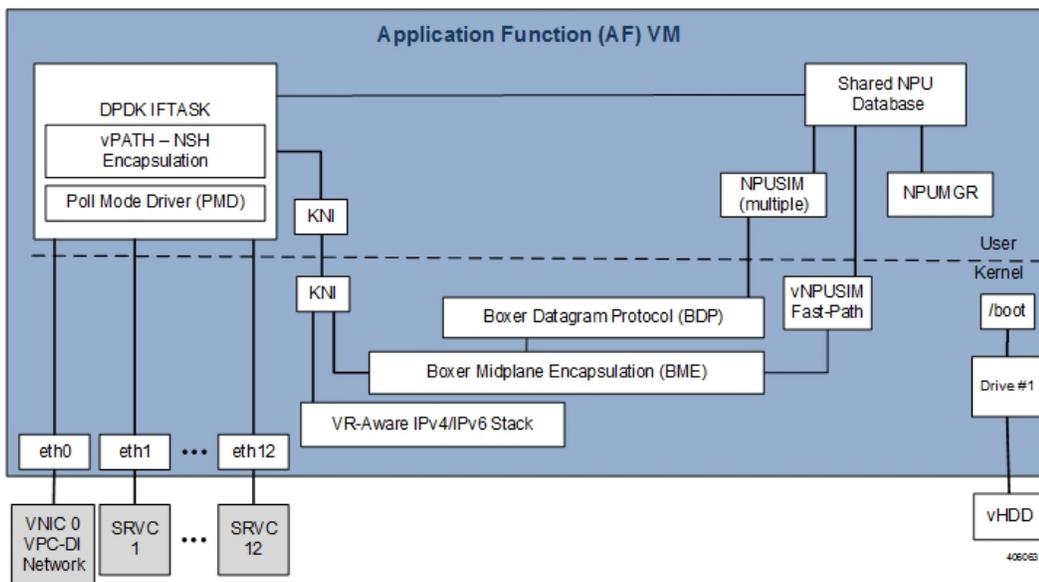
The Application Function (AF) is a virtual machine that is dedicated for Ultra Service Framework within a Gi-LAN Service Function Chain. The CF manages the system initialization, resource management, and high availability of the AF virtual machines. Packets that will be routed through a service function are encapsulated by the NF using NSH chain and routed to the AF. The AF learns of the specific service chain from the NSH header and routes the un-encapsulated packets through the Ultra Service Components (USCs) that comprise the chain. Once the packets are serviced, they are re-encapsulated and routed back to the NF.

Figure 7: AF Network



The AF VM maps the service chain identifier to a local tag representing the link/path between the NF and service component. The service path consists of a single service function, chain of different service functions, or service path spawned over multiple hosts. Like the NF, the AF deploys a FAST-PATH architecture leveraging the network processing unit (NPU) Manager and NPU SIM software tasks to ensure performance and scalability.

Figure 8: AF VM



## USP VNF Types

The USP supports different types of VNFs that provide a variety of mobility services. Each VNF consists of components (VNFCs) which run on different virtual machines (VMs). The following VNF types are supported in this release:

- Ultra Gateway Platform (UGP):** The UGP currently provides virtualized instances of the various 3G and 4G mobile packet core (MPC) gateways that enable mobile operators to offer enhanced mobile data services to their subscribers. The UGP addresses the scaling and redundancy limitations of VPC-SI (Single Instance) by extending the StarOS boundaries beyond a single VM. UGP allows multiple VMs

to act as a single StarOS instance with shared interfaces, shared service addresses, load balancing, redundancy, and a single point of management.

- **Ultra Policy Platform (UPP):** Delivers next generation policy and subscriber management functionality by leveraging the Cisco Policy Suite (CPS). CPS is carrier-grade policy, charging, and subscriber data management solution. It helps service providers rapidly create and bring services to market, deliver a positive user experience, and optimize network resources.



---

**Note** The UPP is not supported in this release.

---

- **Ultra Service Framework (USF):** The USF enables enhanced processing through traffic steering capabilities for subscriber inline services. USF Gi-LAN Service Function Chains (SFC) classify and steer traffic enabling mobile operators to quickly deploy new services and applications to their subscribers.

## Ultra Gateway Platform

The UGP currently provides virtualized instances of the various 3G and 4G mobile packet core (MPC) gateways that enable mobile operators to offer enhanced mobile data services to their subscribers. The UGP addresses the scaling and redundancy limitations of VPC-SI (Single Instance) by extending the StarOS boundaries beyond a single VM. UGP allows multiple VMs to act as a single StarOS instance with shared interfaces, shared service addresses, load balancing, redundancy, and a single point of management.

The UGP includes the following features:

- Software defined, fully featured packet core functionality
- Multi-generational
- Separated management, control and user-planes
- Remotely deployable user plane for ultimate elasticity and scalability

## Ultra Service Framework

The Ultra Service Framework (USF) is a Cisco 4G/5G pluggable framework that enables enhanced session processing through traffic steering capabilities for packets received over the Gi/SGi interface. It provides a pluggable framework for in-line, subscriber-aware, enhanced services.

It is integrated as separately upgradeable software packages. These applications are generically referred to as enablers or services. However, in the context of USF, they are known as Ultra Service Components (USCs). Mobile operators not only deploy USCs to improve and add value to subscriber experience, but also to optimize and increase performance and efficiency within their network infrastructure.

The USF provides native life-cycle management and configuration automated by the converged platform framework. Leveraging 3GPP Flexible Mobile Service Steering (FMSS) and IETF(S) Gi-LAN Service Function Chaining (SFC) concepts, the USF classifies and steers session traffic (per-session or per-flow) to applications based on defined policies.

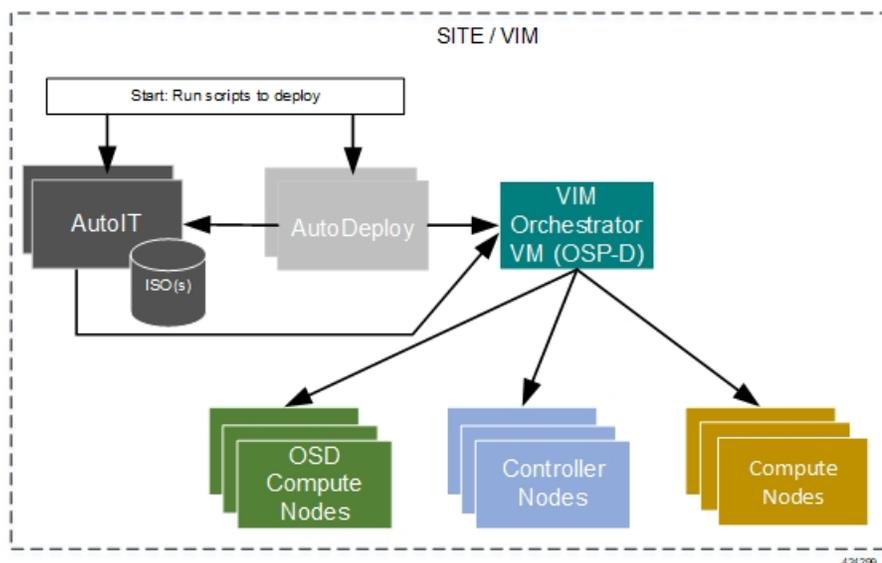
# Ultra Automation Services

Ultra Automation Services (UAS) is an automation framework consisting of a set of software roles used to automate the VIM and USP-based VNF deployment as well as related components such as the VNFM. Beyond deployment automation, UAS manages software bundle components within an inventory manager. In addition, it can also be used to automate the deployment of third party components such as NFVI/VIM, test tools, and USFs that are not part of the distributed USP software bundle. The UAS consists of:

- [AutoIT, on page 17](#)
- [AutoDeploy, on page 19](#)
- [AutoVNF, on page 21](#)

[Figure 9: VIM Installation Automation Workflow, on page 16](#) displays a high-level view of the VIM installation automation process workflow using UAS.

**Figure 9: VIM Installation Automation Workflow**



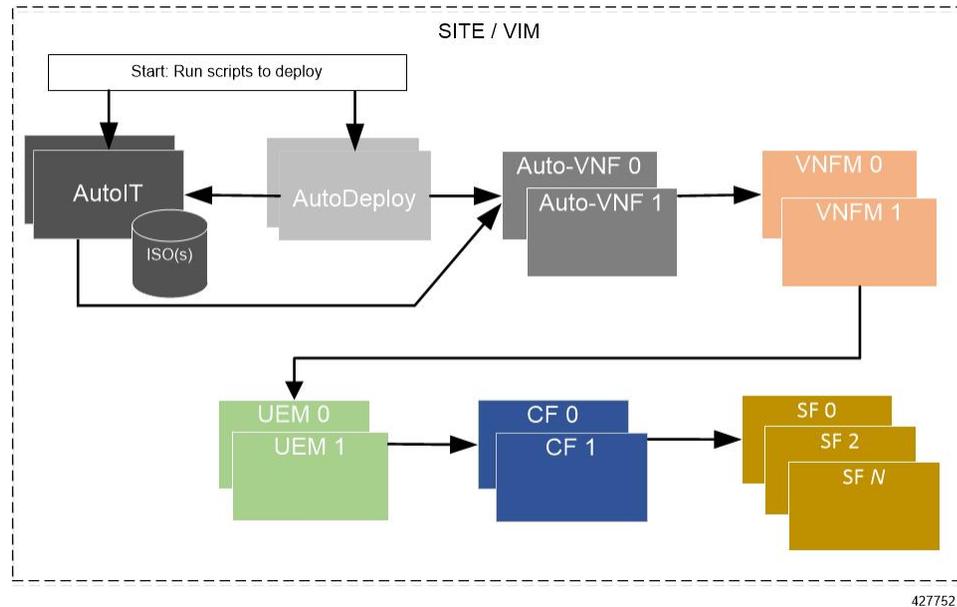
[Figure 10: High-level Single VNF Deployment Automation Workflow, on page 17](#) displays a high-level view of the deployment automation workflow for a single VNF. In a multi-VNF environment, AutoDeploy can deploy up to four VNFs concurrently. Additional details pertaining to the deployment automation process are provided in the deployment automation documentation.



## Important

In this release, multi-VNF deployments are supported only in the context of the Ultra M solution. Refer to the *Ultra M Solutions Guide* for details.

Figure 10: High-level Single VNF Deployment Automation Workflow



## AutoIT

AutoIT is the UAS software role used to automate the process of:

- Deploying the VIM Orchestrator (synonymous with the OpenStack Undercloud).
- Installing the virtual infrastructure manager (VIM, synonymous with the OpenStack Overcloud) which manages the network function virtualization infrastructure (NFVI).
- Onboarding/upgrading the USP ISO software package onto the Ultra M Manager Node.

AutoIT performs the deployments based on manifests it receives from AutoDeploy. Additionally, also hosts a webserver to facilitate VM deployment and delivery of software packages using REST and ConfD APIs for provisioning Overcloud nodes.

AutoIT can be deployed in the following scenarios:

- As a single VM on the Ultra M Manager Node (the same physical server as AutoDeploy and OSP-D VM) during a bare metal installation.
- In high-availability (HA) mode which provides 1:1 redundancy. When deployed in HA mode, two AutoIT VMs are deployed: one active, one standby.
- As a single VM within an existing OpenStack deployment.
- In HA mode within an existing OpenStack deployment.

When supporting VIM installation automation processes, AutoIT:

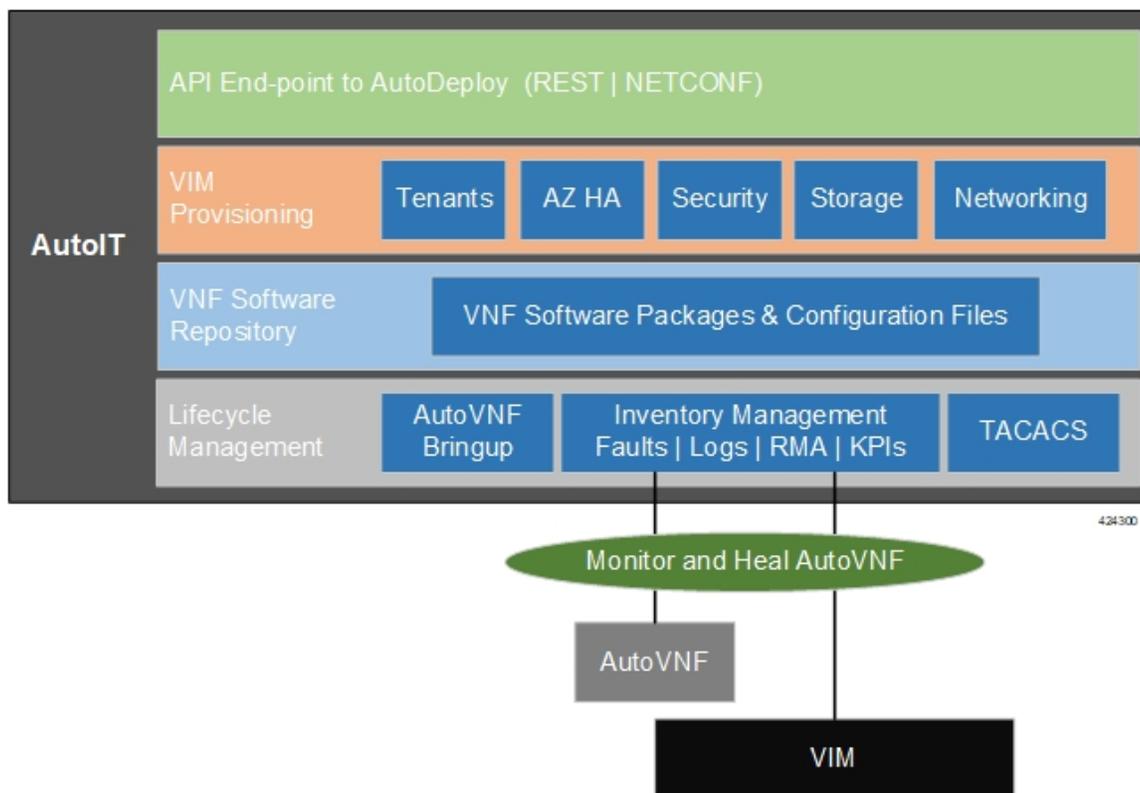
- Sets up AutoIT nodes
- API endpoint based on ConfD to Auto-Deploy and NSO

- Deploys the VIM Orchestrator
- Works through the VIM Orchestrator to deploy the VIM
- Brings up OSP-D as a VM

When supporting VNF deployment automation processes, AutoIT:

- Onboarding Ultra Automation Services (UAS) VMs.
- VIM provisioning to onboard VNFs.
- Manages different version of software packages by hosting into YUM repo.
- APIs to onboard VNF packages.
- Brings up AutoVNF VMs and monitors for failures.
- Stores release public key information in the ISO database for RPM signature verification by YUM through the installation process.

Figure 11: AutoIT Functions



**Important**

In this release, AutoIT is only supported for use with Ultra M solutions based on the Hyper-Converged architecture.

In addition to supporting deployment workflows, AutoIT provides a centralized monitor and management function within the Ultra M solution. This function provides a central aggregation point for events (faults and alarms) and a proxy point for syslog generated by the different components within the solution.

[Table 2: TCP/UDP Services and Open Ports for AutoIT, on page 19](#) lists the TCP/UDP services and the corresponding open ports for AutoIT.

**Table 2: TCP/UDP Services and Open Ports for AutoIT**

Port	Service
22	SSH
8888	CONFD
4569	CONFD
514	SYSLOG
2022	CONFD
5000	HTTP
8008	CONFD
2024	CONFD
161	SNMP

## AutoDeploy

AutoDeploy is the UAS software role that provides single- and multi-Site AutoVNF orchestration. In this context, a “Site” is a single VIM instance. As such, a single AutoDeploy instance is capable of deploying the AutoVNF UAS software roles within multiple deployment scenarios:

- Single VIM/Single VNF
- Single VIM/Multi-VNF



### Important

In this release, multi-VNF deployments are supported only in the context of the Ultra M solution. Refer to the *Ultra M Solutions Guide* for details.

In a multi-VNF environment, AutoDeploy can deploy up to four VNFs concurrently. Additional details pertaining to the deployment automation process are provided in the deployment automation documentation.

AutoDeploy can be deployed in the following scenarios:

- As part of VIM installation automation process:
  - On bare-metal with high availability (HA) support. HA support provides 1:1 VM redundancy. When deployed in HA mode, two AutoDeploy VMs are deployed on the same physical server: one active, one standby.

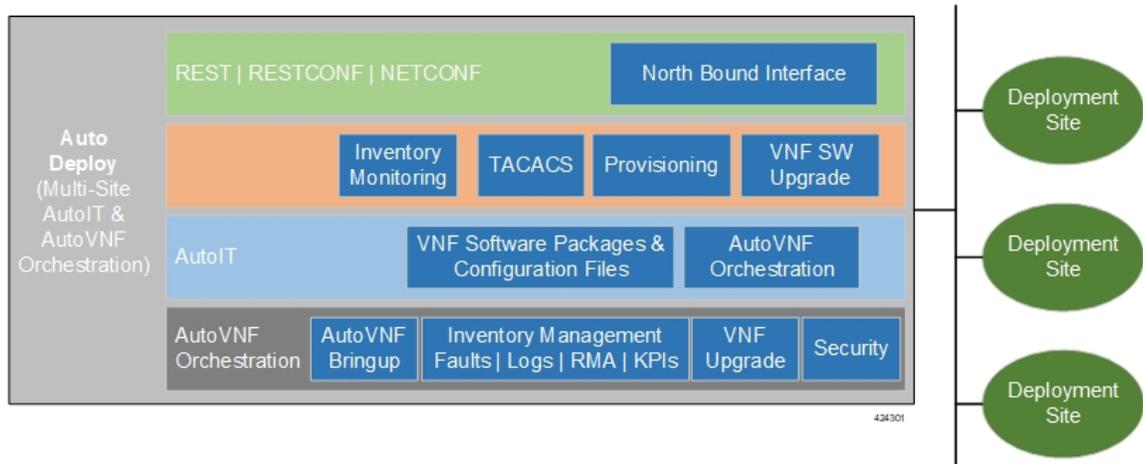
- On bare-metal without HA support. In this scenario, a single AutoDeploy VM is deployed.
- As part of an existing deployment:
  - In HA mode within an existing OpenStack deployment. When deployed in HA mode, two AutoDeploy VMs are deployed on the same physical server: one active, one standby.
  - As a single VM within an existing OpenStack deployment.

In this release, one AutoDeploy VM is deployed per VIM. The AutoDeploy VM must have network access to the VIM in order to provide orchestration.

Once instantiated, AutoDeploy provides the following functionality:

- AutoVNFs bootstrapping and provisioning for deployments (Day-0/Day-1/Day-N).
- AutoVNF Deployments Life-Cycle including start, stop and Inventory management (consolidated).
- Performs release image signing validation by verifying the certificate and public key provided in the release ISO.

**Figure 12: AutoDeploy Functions**



AutoDeploy operations are performed using any of the following methods:

- ConfD CLI and API based transactions
- WebUI based transactions

Table 3: TCP/UDP Services and Open Ports for AutoDeploy, on page 20 lists the TCP/UDP services and the corresponding open ports for AutoDeploy.

**Table 3: TCP/UDP Services and Open Ports for AutoDeploy**

Port	Service
5000	HTTP
22	SSH

Port	Service
2022	CONF
2024	CONF
8443	HTTPS
8888	CONF
514	SYSLOG

## AutoVNF

AutoVNF is the software role within UAS that provides deployment orchestration for USP-based VNFs. It does this by emulating an NFVO and VNFM for deployments.

When used in Ultra M solution deployments, AutoVNF is instantiated by the AutoDeploy software role based on configuration data you provide. It is deployed with a 1:1 HA redundancy model. Processes across the VMs are monitored and restarted if necessary. ConfD synchronizes the CDB between the active and standby VMs. Each of the VMs are deployed on separate Compute nodes within your VIM.

For VNF deployments brought up using only AutoVNF (e.g. Stand-alone AutoVNF-based deployments), only a single VM is deployed.

Once operational, AutoVNF provides the following functionality:

- Deploys the Elastic Services Controller (ESC), which serves as the VNFM, per configurable YANG-based definitions.




---

**Note** The Cisco Elastic Services Controller (ESC) is the only supported VNFM in this USP release.

---

- Onboards all required UEM VMs via the VNFM.
- Leverages configurable YANG-based definitions to generate the VNF descriptor (VNFD) required to onboard the VNF using UEM workflows.
- Determines all required resources for the VNF including images, flavors, networks, subnets and invokes NETCONF-based APIs to provision all of these resources into OpenStack through the VNFM.
- Ensures all references, network, images, and flavors exist on the VIM, if supplied.
- Monitors for NETCONF-based notifications, submits the transaction, and waits until the given transaction succeeds.
- Monitors inventory in terms of operational state and KPIs and auto-heals the VNFM and UEM.
- Orchestrates USP-based VNF upgrades regardless of whether or not Inter-Chassis Session Recovery (ICSR) is enabled on the VNF.
- Implements a ConfD-based architecture to provide life cycle management (LCM) through VNF-EM, VNFM, and VIM plugins as shown in [Figure 14: AutoVNF ConfD-based Architecture for Deployment Automation, on page 23](#).

- Supports standard, ConfD-based REST/RESTCONF/NETCONF north-bound interfaces (NBIs).
- Provides VNF security, credentials, and SSH keys through the use of secure-tokens.
- Hosts an HTTP server to serve GET URLs supplied into the VNFD that include such things as configuration files, VDU images, etc.
- Supplies the VNFD to the UEM upon instantiation as Day-0 configuration using an appropriate VNFM-supported mechanism (e.g. in the case of ESC as the VNFM, the VNFD is passed as a Day-0 configuration using the ESC's deployment APIs).
- Onboards all Day-0 configuration files onto the UEM to be passed on to VDUs.
- Allocates the management IP for the CF and UEM VMs along with Virtual IP (VIP) addresses.

**Figure 13: AutoVNF Functions**

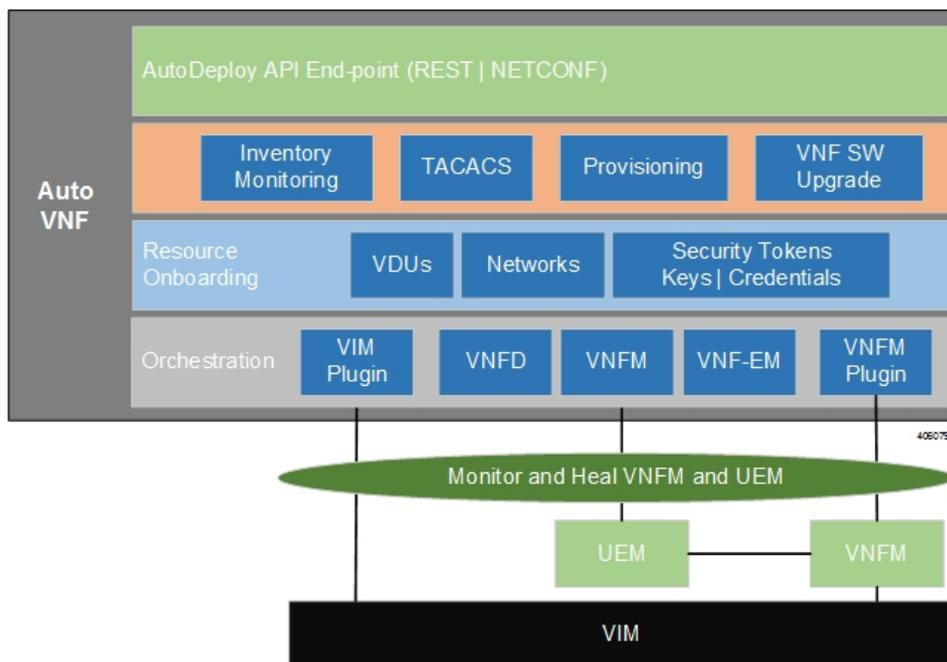
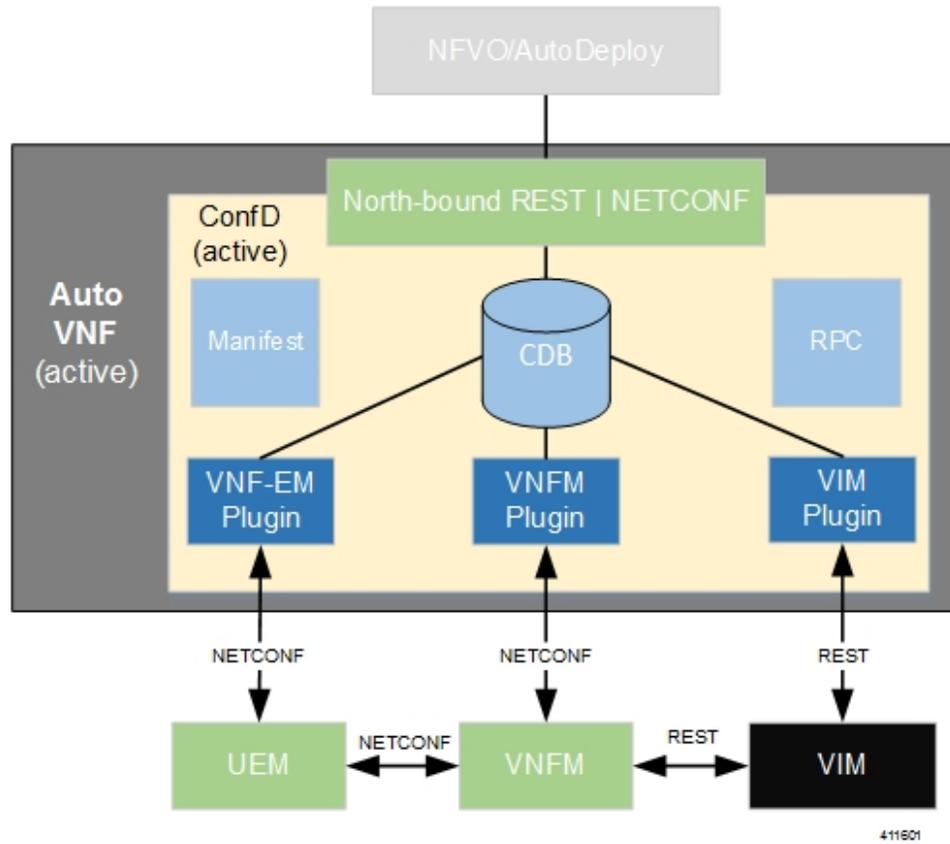


Figure 14: AutoVNF ConfD-based Architecture for Deployment Automation



AutoVNF operations can be performed using any of the following methods:

- ConfD CLI based transactions
- WebUI based transactions
- Netconf based transactions

Table 4: TCP/UDP Services and Open Ports for AutoVNF, on page 23 lists the TCP/UDP services and the corresponding open ports for AutoVNF.

Table 4: TCP/UDP Services and Open Ports for AutoVNF

Port	Service
5000	HTTP
22	SSH
2022	CONF D
2024	CONF D
4569	CONF D

Port	Service
8443	HTTPS
8888	CONFD
8008	CONFD
514	SYSLOG
2181	ZOOKEEPER. AutoVNF connects to Zookeeper running on UEM for status

## Ultra Web Services

The Ultra Web Service (UWS) provides a web-based graphical user interface (GUI) and a set of functional modules that enable users to manage and interact with the USP VNF. It provides a single framework and a top-level dashboard for users to interact with the USP VNF. It includes the following features:

- Service Assurance
- Validation
- VNF-EM Virtualization
- VNF Components
- NFVI/VIM

Leveraging backend-APIs, the VNF visualization module of UWS is used to create, deploy and monitor a variety of USP VNFs based on specific use cases and applications. The VNFs can include definitions for the gateway type, policy options, service function chaining features, and more. After the VNFs are created, users can deploy each VNF to a target infrastructure choice. The USP tracks deploy operations. Users can display the tracked information on the dashboard, and can monitor the status of a selected deployment operation. The dashboard also displays aggregate KPIs from all deployed VNFs, allowing users to easily view and monitor aggregate metrics for a given environment.

UWS software is part of the UAS software package and is installed automatically with the AutoDeploy software role.

The following browser versions are supported for use with the UWS GUI:

- Firefox: 55.0.3 (64-bit)
- Safari: Version 10.1.1 (10603.2.5)
- Chrome: 58.0.3029.110 (64-bit)
- Edge: 38.14393.1066.0




---

**Important** UWS is not supported in 6.x releases.

---

# USP VNF Component Redundancy and Availability

## Platform Requirements

The USP VNF relies on the underlying hardware and hypervisor for overall system redundancy and availability.

The hardware and hypervisor should provide:

- Redundant hardware components where practical (such as power supplies and storage drives)
- Redundant network paths (dual fabric/NICs, with automatic failover)
- Redundant network uplinks (switches, routers, etc.)

High availability can be achieved only if the underlying infrastructure (hosts, hypervisor, and network) can provide availability and reliability that exceeds expected values. The USP VNF is only as reliable as the environment on which it runs.

Inter-Chassis Session Recovery (ICSR) is also recommended to improve availability and recovery time in the case of a non-redundant hardware failure (such as CPU, memory, motherboard, hypervisor software). ICSR provides redundancy at the session level for gateways only. See [ICSR Support, on page 27](#) for more information.

## UEM Redundancy

A minimum of three UEM VMs is required to support redundancy in USP releases prior to 6.3. In 6.3 and later releases, changes are made to the UEM redundancy model in order to optimize the VM requirements. In this release, a minimum of two UEM VMs is sufficient to support redundancy. The UEM supports active-standby 1:1 instances for redundancy reasons.

When three UEM VMs are used, they are deployed as part of an HA cluster which are 1:n redundant for overall management and inter-VNFM communications. The three VMs are deployed as follows: 1 leader or master (active), 1 follower or slave (standby), and 1 follower (standby).

When two VMs are used, the master UEM has two zookeeper instances running, both instances have their own IP, ID, PID, log/data directory and configuration files. The slave UEM also has one zookeeper running, thus meeting three zookeeper instance requirements.

The UEM services will no longer run on the slave UEM to simplify troubleshooting, maintenance, and synchronization related issues.

The number of instances for UEM can be defined as 2 or 3 through the VNFC configuration within NSD. You can configure the instances based on the resource availability and deployment requirements.

To configure the number of instances for UEM, use the following parameter for VNFC EM:

```
number-of-instances <instance_num>
```

Note that the **number-of-instances** parameter is mandatory. This parameter allows the user to configure either 2 or 3 UEM instances.

In releases prior to 6.3, the default value was 3 and this parameter was not user configurable. In release 6.3 and beyond, the default value is 2.

Example Configuration for AutoDeploy:

```

nsd nsd-autovnf
  vnfd vpc
    vnfc em
      number-of-instances 2
...

```

For more information, refer to the *Cisco Ultra Services Platform NETCONF API Guide*.

## CF Redundancy

By default, the UEM deploys two CF VMs which are 1:1 redundant for control of the USP VNF and the local context/management port. This is the recommended configuration.

The management port vNIC on both CFs are 1:1 redundant for each other and must be placed in the same VLAN in the infrastructure. Only one management port is active at a time.




---

**Note** The two CF VMs must not run on the same physical host (server or blade) to achieve redundancy in case of the failure of the host or hypervisor.

---

## SF Redundancy

SFs are deployed using 1:N redundancy. It is recommended that you have at least 2 active and 1 standby SF, however, the number of SF instances will change according to your deployment requirements.

Each SF VM provides network connectivity for service ports. Each SF provides one or more ports and associated interfaces, but the SFs do not provide 1:1 port redundancy as they are not paired together. Redundancy of SF ports should be achieved using ECMP or another supported L3 protocol.

The total throughput required of the USP VNF Instance should not exceed N-2 SFs with session recovery enabled so that any single SF can fail while the others take over its load. Use of loopback interfaces for service IP addresses is highly recommended.

Cisco recommends that you use Bidirectional Forwarding Detection (BFD) and Link Aggregation Group (LAG) for detection of path failures between an SF and the peer router so ECMP paths are excluded in the event of a failure.

1:1 session redundancy within a VNF and Inter-Chassis Session Recovery (ICSR) between VNFs is supported. Note that the session state is check-pointed at various call points within a call flow. Although session state is check-pointed in the UGP, the IP flow state and connection tracking tables are not mirrored. Therefore, any state associated with an IP flow will be lost.

When session recovery is enabled, one VM becomes the VPN/Demux and the remainder are session processing VMs. A standby SF can provide redundancy for any other SF.




---

**Note** Each SF VM must run on a different physical host to achieve redundancy in case of the failure of the host or hypervisor.

---

## NF Redundancy

NFs are deployed using 1:N redundancy. You may adjust the number of NF instances according to your deployment requirements.



---

**Note** Each NF VM must run on a different physical host to achieve redundancy in case of the failure of the host or hypervisor.

---

## AF Redundancy

AFs are deployed using 1:N redundancy. You may adjust the number of AF instances according to your deployment requirements.



---

**Note** Each AF VM must run on a different physical host to achieve redundancy in case of the failure of the host or hypervisor.

---

## Ultra Service Component (USC) Redundancy

The Ultra Services Components (USCs) used in the USF are deployed along with the AF into a MANO construct called an Element Group (EG). An EG is set of VDUs arranged for a unit of redundancy. As such, redundancy is available at the EGs-level and not for the individual USCs. An N:1 redundancy model is supported for Element groups.

## ICSR Support

USP VNFs support Inter-Chassis Session Recovery (ICSR) between two VNF instances for services that support Layer 3 ICSR in the StarOS software release. When more than one service type is in use, only those services that support ICSR will be able to use ICSR.

ICSR supports redundancy for Site/row/rack/host outages, and major software faults. To do so, the two USP VNF instances should be run on non-overlapping hosts and network interconnects. ICSR is supported only between like-configured UGP instances.



---

**Note** ICSR between an USP VNF instance and another type of platform (such as an ASR 5500) is not supported.

---

For additional information, refer to the *Inter-Chassis Session Recovery* chapter in the *System Administration Guide* for your platform.





## CHAPTER 2

# USP Installation Prerequisites

This chapter contains general installation prerequisites including hardware and software requirements. Though these requirements will support various deployment scenarios, the requirements for your specific use case and deployment scenario may differ.

- [Ultra M Deployments Using UAS, on page 29](#)
- [VNF Deployments Using AutoVNF, on page 32](#)

## Ultra M Deployments Using UAS

### Virtual Machine Recommendations

[Table 5: Minimum VM Sizing Recommendations, on page 29](#) lists the minimum recommended VM sizing configurations per VNF component. Your specific requirements for CF and SF VM sizing may vary based on your deployment scenario.

**Table 5: Minimum VM Sizing Recommendations**

Functions	Minimum Required	vCPU	RAM (GB)	Root Disk (GB)
OSP-D*	1	16	32	200
AutoIT	1**	2	8	80
AutoDeploy	1**	2	8	80
AutoVNF	2	2	4	40
ESC (VNFM)	2	2	8	40
UEM***	2	2	4	40
CF	2	8	16	6
SF	3	12	16	6

Functions	Minimum Required	vCPU	RAM (GB)	Root Disk (GB)
<p>* OSP-D is deployed as a VM within the Ultra M solution. Though the recommended root disk size is 200GB, additional space can be allocated if available.</p> <p>** AutoIT and AutoDeploy each minimally require 1 VM when deployed in non-HA mode. When deployed with HA, each requires 2 VMs.</p> <p>*** UEM VM requirement is dependent on VNFD configuration. It can be either 2 or 3 instances.</p>				



**Important** Ultra M solutions have specific requirements. Refer to the *Ultra M Solutions Guide* for more information.

## Software Requirements

[Table 6: Software Requirements, on page 30](#) identifies the software that must be installed on the prerequisite hardware before installing the USP.

**Table 6: Software Requirements**

Purpose	Software
RedHat Enterprise Linux	Release 6.1 and earlier: RedHat 7.3 Release 6.2.x: RedHat 7.4 Release 6.2.bx: RedHat 7.5 Release 6.3 : RedHat 7.5
Virtual Infrastructure Manager (VIM)	<b>Hyper-Converged Ultra M Single and Multi-VNF Models:</b> RedHat OpenStack Platform 10 (OSP 10 - Newton)
VIM Orchestrator	<b>Hyper-converged Ultra M Single and Multi-VNF Models:</b> RedHat OpenStack Platform 10 (OSP 10 - Newton)
UAS Component Operating System	Release 6.3 and earlier: Ubuntu 14.04 Release 6.4 and later: Ubuntu 16.04

In addition to the preceding software, it is assumed that you have downloaded the latest USP software ISO.

## Hardware Requirements

### Server Functions and Quantities

The servers host the VMs required by the USP-based VNF. Though server functions and quantity differ depending on your deployment scenario, the following server functions are required for use with UAS in this release:

- **Ultra M Manager Node:** Required only for Ultra M deployments based on the Hyper-Converged architecture, this server hosts the following:
  - AutoIT VM(s)
  - AutoDeploy VM(s)
  - OSP-D VM



**Important** When AutoIT and/or AutoDeploy are deployed in HA mode, both the active and redundant VMs are deployed on the same physical server.

- **OpenStack Controller Nodes:** These servers host the high availability (HA) cluster that serves as the VIM. In addition, they facilitate the Ceph storage monitor function required by the OSD Compute Nodes.
- **OSD Compute Nodes:** Required only for deployments based on the Hyper-Converged architecture, these servers containing a Ceph Object Storage Daemon (OSD) providing storage capacity for the VNF. In addition to hosting the following:

[Table 7: Server Quantities by Function, on page 31](#) provides information on server quantity requirements per function. Your specific server/node requirements may vary based on your deployment scenario.

**Table 7: Server Quantities by Function**

Server Quantity (min)	Ultra M Manager Node	Controller Nodes	OSD Compute Nodes	Compute Nodes (min)	Additional Specifications
15	1	3	3	8	Based on node type as described in <a href="#">Table 8: Minimum Server Specifications by Node Type, on page 31</a> .

**Table 8: Minimum Server Specifications by Node Type**

Node Type	CPU	RAM	Storage
Ultra M Manager Node	2x 2.60 GHz	4x 32GB DDR4-2400-MHz RDIMM/PC4	2x 1.2 TB 12G SAS HDD
Controller	2x 2.60 GHz	4x 32GB DDR4-2400-MHz RDIMM/PC4	2x 1.2 TB 12G SAS HDD
Compute	2x 2.60 GHz	8x 32GB DDR4-2400-MHz RDIMM/PC4	2x 1.2 TB 12G SAS HDD
OSD Compute	2x 2.60 GHz	8x 32GB DDR4-2400-MHz RDIMM/PC4	4x 1.2 TB 12G SAS HDD 2x 300G 12G SAS HDD HDD 1x 480G 6G SAS SATA SSD

## Password Requirements and Login Security

All passwords configured for and/or through UAS components (AutoIT, AutoDeploy, and/or AutoVNF) and UEM must meet the following criteria:

- They must be a minimum of 8 alpha and/or numeric characters.
- They must contain at least one uppercase letter.
- They must contain at least one lowercase letter.
- They must contain at least one number.
- They must contain at least one special character (e.g. @, #, \$, etc.) with an exception of using exclamation (!) character.

The specified password criteria is applicable to all deployment scenarios — UAS-based Ultra M deployment, Standalone Auto-VNF-based deployment, and UEM-based VNF deployment.

For UAS and UEM components, the following login security restrictions are supported:

- You will be locked out of the system for 10 minutes upon the third incorrect attempt to login to a UAS and UEM VM.
- Should you need/want to change your password, the new password must be different than any of the last five previously configured passwords.

## VNF Deployments Using AutoVNF

### Virtual Machine Recommendations

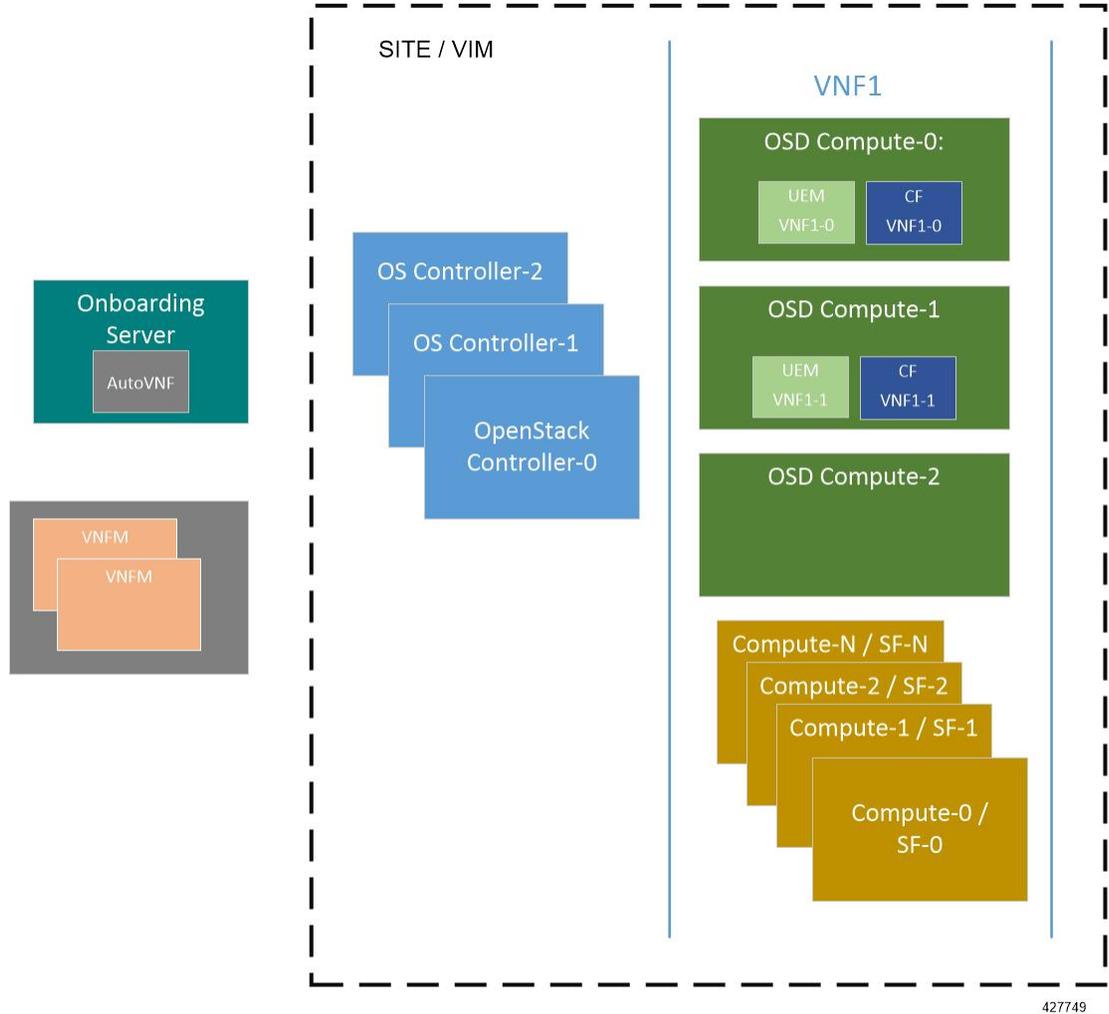
[Table 9: Minimum VM Sizing Recommendations, on page 32](#) lists the minimum recommended VM sizing configurations per VNF component. Your specific requirements for CF and SF VM sizing may vary based on your deployment scenario.

**Table 9: Minimum VM Sizing Recommendations**

Functions	Minimum Required	vCPU	RAM (GB)	Root Disk (GB)
AutoVNF	1	2	4	40
ESC (VNFM)*	2	2	8	40
UEM**	2	2	4	40
CF	2	8	16	6
SF	3	12	16	6
* Though ESC VM sizing recommendations are provided, ESC deployment information and instructions is beyond the scope of this document. Refer to the ESC product documentation for details.				
** UEM VM requirement is dependent on VNFD configuration. It can be either 2 or 3 instances.				

The VMs identified in [Table 9: Minimum VM Sizing Recommendations, on page 32](#) are deployed in your NFVI as shown in [Figure 15: VM Distribution for VNF Deployments Using AutoVNF, on page 33](#).

**Figure 15: VM Distribution for VNF Deployments Using AutoVNF**



**Note** The above figure depicts an example deployment scenario. The placement of the VMs is based on your deployment requirements.

## Software Requirements

[Table 10: Software Requirements, on page 34](#) identifies the software that must be installed on the prerequisite hardware before installing the USP.

Table 10: Software Requirements

Purpose	Software
Cloud Computing Platform	OpenStack Platform 10 (OSP 10 - Newton)  <b>Note</b> OpenStack Keystone API versions 2 and 3 are supported. Ensure that all aspects of your deployment are configured to use the same API version.
VNFM	In releases prior to 6.0: Cisco Elastic Services Controller 3.1.0.116  In releases 6.0 and 6.1: Cisco Elastic Services Controller 3.1.0.145  In release 6.2: Cisco Elastic Services Controller 4.0.0.104  In release 6.3: Cisco Elastic Services Controller 4.2  In release 6.4 : Cisco Elastic Services Controller 4.3.0.121
UAS Component Operating System	In releases prior to 6.3: Ubuntu 14.04  In release 6.4 and later: Ubuntu 16.04

In addition to the preceding software, it is assumed that you have downloaded the latest USP software ISO.

## Hardware Requirements

### Server Functions and Quantities

The servers host the VMs required by the USP-based VNF. Though server functions and quantity differ depending on your deployment scenario, the following server functions are required for VNF installation using AutoVNF:

- **Staging Server Node:** This server hosts the AutoVNF VM.
- **OpenStack Controller Nodes:** These servers host the high availability (HA) cluster that serves as the VIM. In addition, they facilitate the Ceph storage monitor function required by the OSD Compute Nodes.
- **OSD Compute Nodes:** Required only for deployments based on the Hyper-Converged architecture, these servers containing a Ceph Object Storage Daemon (OSD) providing storage capacity for the VNF. In addition to hosting the following:
  - Ultra Element Manager (UEM) HA cluster VMs
  - Ultra Service Platform (USP) Control Function (CF) active and standby VMs
- **Compute Nodes:** For all deployments, these servers host the active, standby, and demux USP Service Function (SF) VMs.



**Important** The above information assumes that the VNFM (ESC) was previously deployed.

[Table 11: Server Quantities by Function, on page 35](#) provides information on server quantity requirements per function. Your specific server/node requirements may vary based on your deployment scenario.

**Table 11: Server Quantities by Function**

Server Quantity (min)	Red Hat Bare Metal/ Staging Server Node	Controller Nodes	OSD Compute Nodes	Compute Nodes (min)	Additional Specifications
15	1	3	3	8	Based on node type as described in  <a href="#">Table 12: Minimum Server Specifications by Node Type, on page 35.</a>

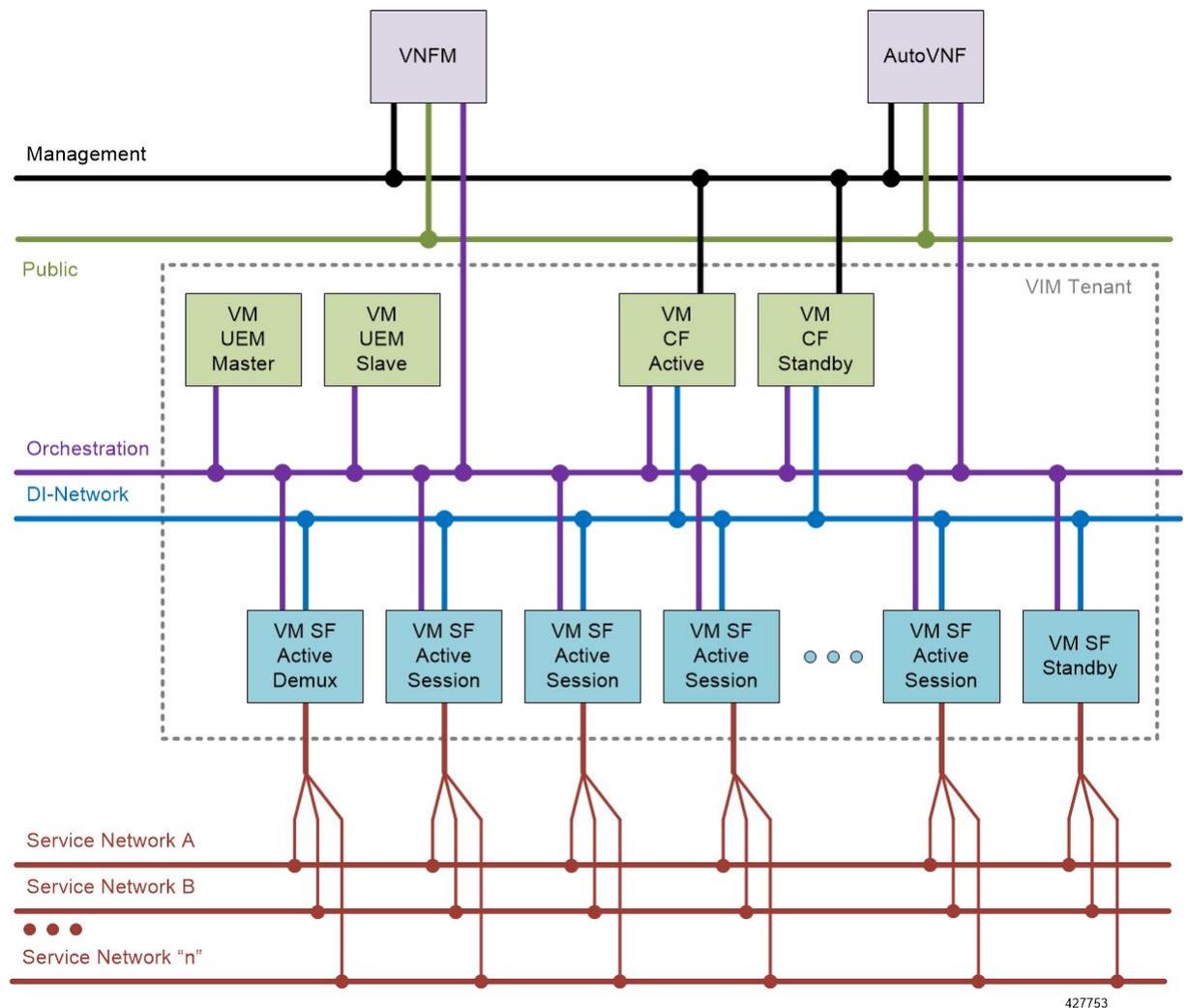
**Table 12: Minimum Server Specifications by Node Type**

Node Type	CPU	RAM	Storage
Staging Server	2x 2.60 GHz	4x 32GB DDR4-2400-MHz RDIMM/PC4	2x 1.2 TB 12G SAS HDD
Controller	2x 2.60 GHz	4x 32GB DDR4-2400-MHz RDIMM/PC4	2x 1.2 TB 12G SAS HDD
Compute	2x 2.60 GHz	8x 32GB DDR4-2400-MHz RDIMM/PC4	2x 1.2 TB 12G SAS HDD
OSD Compute	2x 2.60 GHz	8x 32GB DDR4-2400-MHz RDIMM/PC4	4x 1.2 TB 12G SAS HDD 2x 300G 12G SAS HDD HDD 1x 480G 6G SAS SATA SSD

## Network Requirements

While specific VNF network requirements are described in the documentation corresponding to the VNF, [Figure 16: Typical USP-based VNF Networks, on page 36](#) displays the types of networks typically required by USP-based VNFs.

Figure 16: Typical USP-based VNF Networks



The USP-based VNF networking requirements and the specific roles are described here:

- **Public:** External public network. The router has an external gateway to the public network. All other networks (except DI-Internal and ServiceA-n) have an internal gateway pointing to the router. And the router performs secure network address translation (SNAT).
- **DI-Internal:** This is the DI-internal network which serves as a 'backplane' for CF-SF and CF-CF communications. Since this network is internal to the UGP, it does not have a gateway interface to the router in the OpenStack network topology. A unique DI internal network must be created for each instance of the UGP. The interfaces attached to these networks use performance optimizations.
- **Management:** This is the local management network between the CFs and VNFM. To allow external access, an OpenStack floating IP address from the Public network must be associated with the UGP VIP (CF) address.



---

**Note** Prior to assigning floating and virtual IP addresses, make sure that they are not already allocated through OpenStack. If the addresses are already allocated, then they must be freed up for use or you must assign a new IP address that is available in the VIM.

---

- **Orchestration:** This is the network used for VNF deployment and monitoring. It is used by the VNFM to onboard the USP-based VNF.
- **ServiceA-n:** These are the service interfaces to the SF. Up to 12 service interfaces can be provisioned for the SF with this release. The interfaces attached to these networks use performance optimizations.

VNFCs can be assigned a floating IP address from a fixed pool of IP addresses configured for each network type. This is done using the **ip-allocation-pool** parameter in the Virtual Link Descriptor's Network Descriptor. Refer to the *Ultra Services Platform NETCONF API Guide* for more information.

## Password Requirements and Login Security

All passwords configured for and/or through UAS components (AutoIT, AutoDeploy, and/or AutoVNF) and UEM must meet the following criteria:

- They must be a minimum of 8 alpha and/or numeric characters.
- They must contain at least one uppercase letter.
- They must contain at least one lowercase letter.
- They must contain at least one number.
- They must contain at least one special character (e.g. @, #, \$, etc.) with an exception of using exclamation (!) character.

The specified password criteria is applicable to all deployment scenarios — UAS-based Ultra M deployment, Standalone Auto-VNF-based deployment, and UEM-based VNF deployment.

For UAS and UEM components, the following login security restrictions are supported:

- You will be locked out of the system for 10 minutes upon the third incorrect attempt to login to a UAS and UEM VM.
- Should you need/want to change your password, the new password must be different than any of the last five previously configured passwords.





## CHAPTER 3

# Deploying Hyper-Converged Ultra M Models Using UAS

---

This chapter provides information on the following topics:

- [Virtual Infrastructure Manager Installation Automation, on page 39](#)
- [VNF Deployment Automation, on page 60](#)

## Virtual Infrastructure Manager Installation Automation

### Introduction

Leveraging RedHat and OpenStack's TripleO project concepts, UAS supports the ability to automate the deployment of both the virtual infrastructure manager (VIM, the Triple O Overcloud) and the VIM Orchestrator (the TripleO Undercloud).

Installing the VIM Orchestrator and the VIM involves deploying the following components as VMs on a RedHat Enterprise Linux (RHEL) server:

- AutoIT
- AutoDeploy
- OpenStack Platform Director (OSP-D)

VIM Orchestrator and VIM settings are maintained in configuration files which are used by AutoDeploy.

AutoDeploy processes the VIM Orchestrator configuration and works with AutoIT to automate the deployment of a VM running OSP-D which serves as the Undercloud. Once this operation is successful, AutoDeploy processes the VIM configuration and works with AutoIT to deploy the OpenStack Overcloud.

Notes:

- This functionality is supported only with Ultra M deployments based on OSP 10 and that leverage the Hyper-Converged architecture.
- Refer to [Pre-Virtual Infrastructure Manager Installation Verification, on page 41](#) for pre-requisites pertaining to this feature.

## VIM Installation Automation Overview

Figure 17: NFVI Deployment Automation Workflow, on page 40 provides an overview of the deployment automation process. Details are provided in Table 13: Virtual Infrastructure Manager Installation Automation Workflow Descriptions, on page 40. This information assumes that all prerequisite hardware has been installed, cabled, and configured.



### Important

The workflow information in this section assumes a new deployment scenario. If you are using this feature in relation with an upgrade process, please contact your support representative for complete details.

Figure 17: NFVI Deployment Automation Workflow

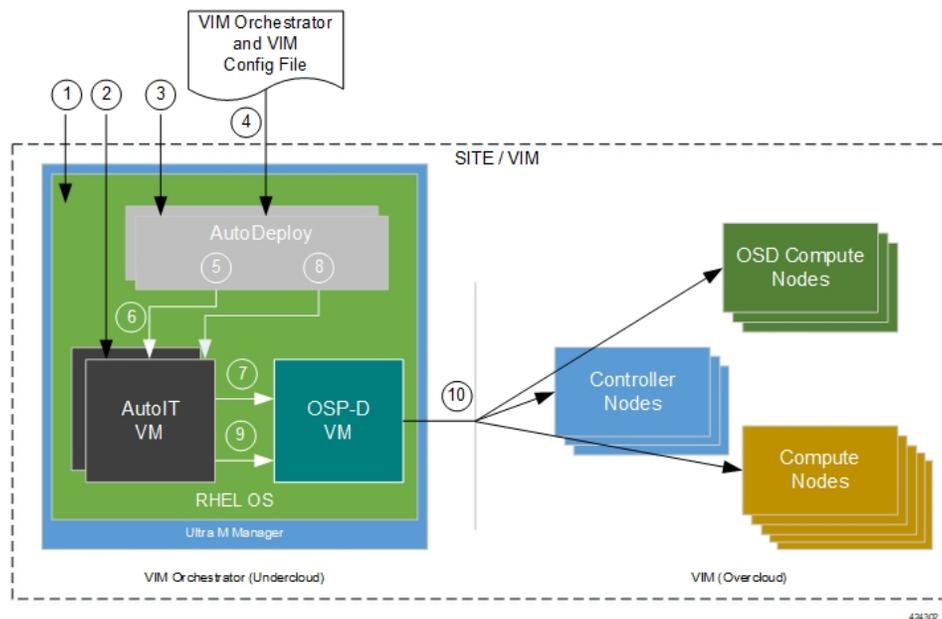


Table 13: Virtual Infrastructure Manager Installation Automation Workflow Descriptions

Callout	Description
1	Install RedHat Enterprise Linux (RHEL) operating system on bare metal hardware (Ultra M Manager Node).
2	Deploy the AutoIT VMs.
3	Deploy the AutoDeploy VMs.
4	<p>Prepare the file containing the VIM Orchestrator and VIM. This file is used by AutoDeploy to initiate the OSP-D VM deployment process and to bring up the VIM.</p> <p>This file includes all the configuration information required to deploy OSP-D VM and VIM including configurations for constructs such as secure tokens, package images, NFVI point-of-presence descriptors (nfvi-popd), the VIM Orchestrator descriptor (vim-orchd), and VIM role and node information. Refer to <a href="#">Sample VIM Orchestrator and VIM Configuration File, on page 183</a> for more information.</p>

Callout	Description
5	On the AutoDeploy VM, load, commit, and then activate the configuration file prepared in the previous step.
6	AutoDeploy passes data from the activated configuration to AutoIT requesting that it deploy the OSP-D VM for the Undercloud. Refer to <a href="#">Activate the VIM Orchestrator and VIM Deployment, on page 58</a> for more information.
7	AutoIT deploys the OSP-D VM which serves as the Undercloud.
8	AutoDeploy passes VIM data from the activated configuration to AutoIT for delivery to the OSP-D VM responsible for installing the VIM.
9	AutoIT initiates the VIM installation by passing parameters received from AutoDeploy to the OSP-D VM.
10	The OSP-D VM installs the VIM per the configuration requirements.

Once all the VIM servers have been successfully deployed, the process of deploying the VNF can begin as described in [VNF Deployment Automation, on page 60](#).

## Pre-Virtual Infrastructure Manager Installation Verification

Prior to installing the virtual infrastructure manager (VIM) and the VIM Orchestrator, please ensure that the following is true:

- Ensure that all required hardware is installed, powered on, cabled and configured according to the information and instructions in the *Ultra M Solutions Guide*. Refer to the following sections in that document:
  - *Hardware Specifications*
  - *Install and Cable the Hardware*
  - *Configure the Switches*
  - *Prepare the UCS C-Series Hardware*
- Ensure that all required software is available and that you have access to the Cisco-provided USP ISO image. See the *Software Specifications* section of the *Ultra M Solutions Guide* for more details.
- Ensure that the following repos are always enabled for Satellite Server and CDN Server:
  - rhel-7-server-rpms
  - rhel-7-server-rh-common-rpms
  - rhel-7-server-extras-rpms
  - rhel-ha-for-rhel-7-server-rpms
  - rhel-7-server-optional-rpms
  - rhel-7-server-rhscon-2-installer-rpms
  - rhel-7-server-openstack-10-rpms

- [rhel-7-server-rhceph-2-mon-rpms](#)
- [rhel-7-server-rhceph-2-osd-rpms](#)
- [rhel-7-server-rhceph-2-tools-rpms](#)

## Install the VIM Orchestrator

The initial part of the Virtual Infrastructure Manager installation automation process is to install the VIM Orchestrator. You cannot install the VIM until after the VIM Orchestration installation is successful.



---

**Important**

Before proceeding, ensure that all of the items in [Pre-Virtual Infrastructure Manager Installation Verification, on page 41](#) have been verified.

---

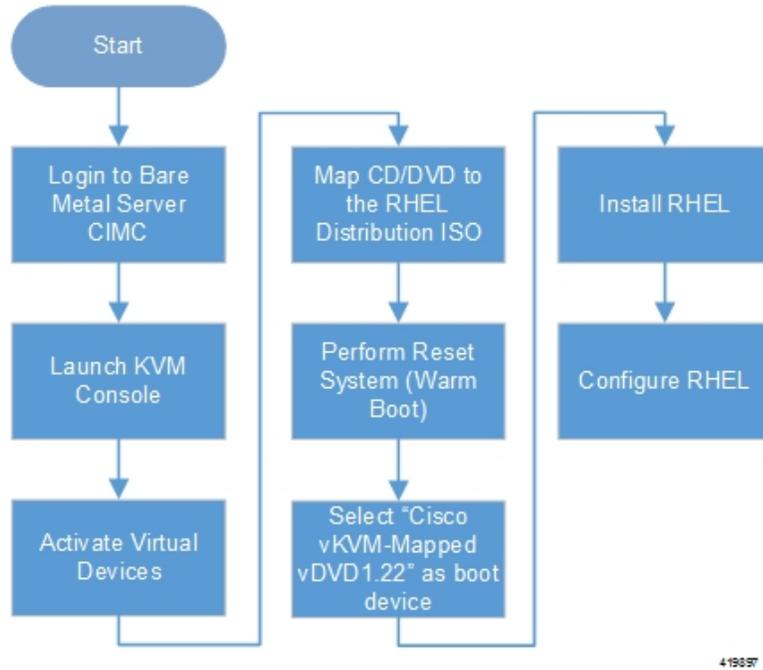
To install the VIM Orchestrator:

1. [Install and Configure RHEL, on page 42.](#)
2. [Onboard the USP ISO, on page 48.](#)
3. [Extract the UAS Bundle, on page 50.](#)
4. [Deploy AutoIT, on page 51.](#)
5. [Deploy AutoDeploy, on page 54.](#)
6. [Prepare the VIM Orchestrator and VIM Configuration File, on page 58](#) based on your deployment requirements.
7. [Activate the VIM Orchestrator and VIM Deployment, on page 58.](#)

## Install and Configure RHEL

As described in [VIM Installation Automation Overview, on page 40](#), the VIM Orchestrator (OSP-D) is deployed as a VM on top of RHEL. [Figure 18: Installation Process for RHEL Bare Metal Server, on page 43](#) illustrates the process for installing RHEL.

Figure 18: Installation Process for RHEL Bare Metal Server



General RHEL installation information and procedures are located in the product documentation:

- <https://access.redhat.com/documentation/en/red-hat-enterprise-linux/>

Prior to installing RHEL, refer to [Table 14: Red Hat Installation Settings, on page 43](#) for settings required for the VIM Orchestrator installation in Ultra M.



**Note** [Table 14: Red Hat Installation Settings, on page 43](#) assumes that you are using the product’s graphical user interface (GUI) for Red Hat installation.

Table 14: Red Hat Installation Settings

Parameters and Settings	Description
Installation Summary > Language Support	
English > English (United States)	Sets the language to English and the region to United States.
Installation Summary > Software Selection	
Base Environment = Virtualization Host Add-Ons for Selected Environment = Virtualization Platform	
Installation Summary > Network & Host Name	

Parameters and Settings	Description
Host name	Configure the desired host name.
Installation Summary > Network & Host Name > Ethernet (eno2) > Configure > IPv4 Setting	
IP Address Netmask Gateway DNS Server Search Domain	Configure and save settings for the network interface by which the server can be accessed externally.
Installation Summary > Installation Destination > CiscoUCSC-MRAID12G (sda) > I will configure partitioning > Click here to create them automatically	
Select all partitions, then click “-“ / = 100GB /var = 500GB /swap = 100GB /home = remaining space /boot = 1GB	Removes any previously configured partitions and creates partitions with the required sizes. <b>Note</b> You must use LVM-based partitioning.
Installation Summary > KDUMP	
kdump = disabled	It is recommended that kdump be disabled.
Installation Summary > Begin Installation > User Settings	
Root Password	Configure and confirm the root user password.
Create user “nfvi”	Creates a new user account. This account is used during the VIM Orchestration installation to log onto the Ultra M Manager Node. <b>Note</b> Ensure that a strong password is used. It must be a minimum of 8 alpha and/or numeric characters and must contain at least 1 uppercase letter, 1 lowercase letter, 1 number, and 1 special character (e.g. @, #, \$, etc.).

To install and configure RHEL:

1. Follow the CIMC processes on the bare metal server as identified in [Figure 18: Installation Process for RHEL Bare Metal Server, on page 43](#).
2. Select the option to install Red Hat Enterprise Linux to begin the installation.
3. Configure the settings identified in [Table 14: Red Hat Installation Settings, on page 43](#).
4. Begin the installation and configure the User Setting identified in [Table 14: Red Hat Installation Settings, on page 43](#).
5. Click **Reboot** once the installation is complete.
6. Log in to RedHat as the **nfvi** user.
7. Set password-less sudo access for **nfvi**.  

```
echo "nfvi ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/nfvi
chmod 0440 /etc/sudoers.d/nfvi
```
8. Configure the network interfaces and network bridges.

**Important**

If any of the network interface or bridge configuration files do not exist, create the related configuration files. Example configuration files are provided in [Example RedHat Network Interface and Bridge Configuration Files, on page 233](#).

- a. Configure the eno2 interface by appending the following parameters to the `/etc/sysconfig/network-scripts/ifcfg-eno2` file.

```
<--SNIP-->
DEVICE=eno2
ONBOOT=yes
BRIDGE=br-ex
NM_CONTROLLED=no
NETMASK=<netmask>
GATEWAY=<gateway_address>
```

- b. Configure the eno1 interface by appending the following parameters to the `/etc/sysconfig/network-scripts/ifcfg-eno1` file.

```
<--SNIP-->
DEVICE=eno1
ONBOOT=yes
BRIDGE=br-ctlplane
NM_CONTROLLED=no
```

- c. Configure the br-ex network bridge by adding the following parameters to the `/etc/sysconfig/network-scripts/ifcfg-br-ex` file.

```
<--SNIP-->
DEVICE=br-ex
DEFROUTE=yes
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=static
NM_CONTROLLED=no
DELAY=0
IPADDR=<external_ip_address>
NETMASK=<netmask>
GATEWAY=<gateway_address>
PREFIX="24"
DNS1="<DNS_server_address>"
DOMAIN="<domain_name>"
IPV4_FAILURE_FATAL="yes"
```

- d. Configure the br-ctlplane bridge by adding the following parameters to the `/etc/sysconfig/network-scripts/ifcfg-br-ctlplane` file.

```
<--SNIP-->
DEFROUTE=yes
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=static
```

```
NM_CONTROLLED=no
DELAY=0
DEVICE=br-ctlplane
```

**Caution**

Once configured, it is recommended that you do not make any changes to the network interface or bridge configuration. Doing so will require that you redeploy AutoIT and AutoDeploy.

9. Create and prepare the directories required for installing the UAS components.

```
sudo mkdir -p /var/cisco/isos
sudo mkdir -p /var/cisco/disks
sudo chmod 777 -R /var/cisco
```

10. Reboot the bare metal server.

```
sudo reboot
```

11. Login as a root user upon reboot.

**Important**

If the server is not accessible via the configured IP address, login into the server's KVM console and troubleshoot the configuration.

12. Validate the network configuration.

```
ifconfig | more
```

Example output:

```
br-ctlplane: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet6 fe80::22c:c8ff:fed9:f176 prefixlen 64 scopeid 0x20<link>
  ether 00:2c:c8:d9:f1:76 txqueuelen 1000 (Ethernet)
  RX packets 52 bytes 7044 (6.8 KiB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 8 bytes 648 (648.0 B)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

br-ex: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 172.25.22.59 netmask 255.255.255.0 broadcast 172.25.22.255
  inet6 fe80::22c:c8ff:fed9:f177 prefixlen 64 scopeid 0x20<link>
  ether 00:2c:c8:d9:f1:77 txqueuelen 1000 (Ethernet)
  RX packets 1394 bytes 122906 (120.0 KiB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 717 bytes 71762 (70.0 KiB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet6 fe80::22c:c8ff:fed9:f176 prefixlen 64 scopeid 0x20<link>
  ether 00:2c:c8:d9:f1:76 txqueuelen 1000 (Ethernet)
  RX packets 57 bytes 8072 (7.8 KiB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 16 bytes 1296 (1.2 KiB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
  device memory 0xc7000000-c70fffff

eno2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
```

```

        inet6 fe80::22c:c8ff:fed9:f177 prefixlen 64 scopeid 0x20<link>
        ether 00:2c:c8:d9:f1:77 txqueuelen 1000 (Ethernet)
        RX packets 1497 bytes 148860 (145.3 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 726 bytes 72476 (70.7 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
        device memory 0xc6f00000-c6ffffff

enp6s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        ether 00:2c:c8:68:3b:ec txqueuelen 1000 (Ethernet)
        RX packets 1 bytes 68 (68.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp7s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        ether 00:2c:c8:68:3b:ed txqueuelen 1000 (Ethernet)
        RX packets 1 bytes 68 (68.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1 (Local Loopback)
        RX packets 84 bytes 6946 (6.7 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 84 bytes 6946 (6.7 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255
[root@rhel-baremetal nfvi]# brctl show
bridge name bridge id STP enabled interfaces
br-ctlplane 8000.002cc8d9f176 no eno1
br-ex 8000.002cc8d9f177 no eno2
virbr0 8000.5254003d7549 yes virbr0-nic

```

### 13. Perform the RHEL subscription-manager registration.

From Content Delivery Network (CDN) servers:

```

sudo subscription-manager config --server.proxy_hostname=<proxy_url>
--server.proxy_port=80

subscription-manager register --username <username> --password <password>

subscription-manager attach -auto

sudo subscription-manager status

```

From Satellite Servers:

```

rpm -Uvh
http://<satellite_server_domain>/pub/katello-ca-consumer-latest.noarch.rpm

subscription-manager register --org="<organization>"
--activationkey="<activation_key>"

```

Example output:

```

+-----+
  System Status Details

```

```
+-----+
Overall Status: Current
```

14. Install the virtualization packages.

```
yum install virt-install -y
```

Example output:

```
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
rhel-7-server-rpms | 3.5 kB
00:00:00
(1/3): rhel-7-server-rpms/7Server/x86_64/group | 709 kB
00:00:01
(2/3): rhel-7-server-rpms/7Server/x86_64/updateinfo | 2.3 MB
00:00:02
(3/3): rhel-7-server-rpms/7Server/x86_64/primary_db | 42 MB
00:00:16
Resolving Dependencies
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
rhel-7-server-rpms | 3.5 kB 00:00:00
(1/3): rhel-7-server-rpms/7Server/x86_64/group | 709 kB 00:00:01
(2/3): rhel-7-server-rpms/7Server/x86_64/updateinfo | 2.3 MB 00:00:02
(3/3): rhel-7-server-rpms/7Server/x86_64/primary_db | 42 MB 00:00:16
Resolving Dependencies
```

```
yum install virt-viewer -y
```

```
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package virt-viewer.x86_64 0:5.0-7.e17 will be installed
```

15. Install the Python bindings to the OpenStack Compute API.

```
yum install python-novaclient -y
```

16. Install the OpenStack networking API client.

```
yum install python-neutronclient -y
```

17. Install the NETCONF client.

```
yum install python-ncclient -y
```

18. Install the python library for XML and HTML processing.

```
yum install python-lxml -y
```

19. Proceed to [Onboard the USP ISO](#), on page 48.

## Onboard the USP ISO

The files required to deploy the USP components are distributed as RPMs (called “bundles”) in a single ISO package. They are maintained using YUM on the Ultra M Manager Node. The following bundles are part of the ISO:

USP Bundle Name	Description
usp-em-bundle	The Element Manager (EM) Bundle RPM containing images and metadata for the Ultra Element Manager (UEM) module.

USP Bundle Name	Description
usp-uas-bundle	The Ultra Automation Services Bundle RPM containing AutoIT, AutoDeploy, AutoVNF, Ultra Web Services (UWS), and other automation packages.
usp-ugp-bundle	The Ultra Gateway Platform (UGP) Bundle RPM containing images for Ultra Packet core (VPC-DI). This bundle contains non-trusted images.
usp-vnfm-bundle	The VNFM Bundle RPM containing an image and a boot-up script for ESC (Elastic Service Controller).
usp-yang-bundle	The Yang Bundle RPM containing YANG data models including the VNFD and VNFR.
usp-auto-it-bundle	The bundle containing the AutoIT packages required to deploy the UAS.

**Important**

Release 6.4 will not be backward compatible with previous releases, i.e., you cannot deploy a 6.4 ISO from an AutoDeploy/AutoIT/AutoVNF running a pre-6.4 release, and vice-versa.

In addition to the bundles, the ISO bundle also includes scripts used to deploy the bundles including UAS.

**Important**

This procedure is not necessary if you are deploying a VNF on a Hyper-Converged Ultra M mode and have already deployed the VIM Orchestrator and the VIM using the information and instructions in [Virtual Infrastructure Manager Installation Automation, on page 39](#).

**Important**

Before attempting to deploy the Ultra M Manager Node, ensure that the [USP Installation Prerequisites, on page 29](#) have been met.

To onboard the ISO package:

1. Log on to the Ultra M Manager Node.
2. Download the USP ISO bundle and related files pertaining to the release.
3. Create a mount point on the Ultra M Manager Node and mount the ISO package:

```
mkdir /var/usp-iso
```

4. Mount the USP ISO.

```
sudo mount -t iso9660 -o loop <ISO_download_directory>/<ISO_package_name> /var/usp-iso
```

**Example:** The following command mounts the ISO bundle called *usp-5\_5\_0-1255.iso* located in a directory called *5\_5\_0-1283* to */var/usp-iso*:

```
sudo mount -t iso9660 -o loop 5_5_0-1064/usp-5_5_0-1064.iso /var/usp-iso  
mount: /dev/loop1 is write-protected, mounting read-only
```

5. Verify the mount configuration.

```
df -h
```

**Example output:**

```
Filesystem Size Used Avail Use% Mounted on
/dev/sda2 187G 178G 316M 100% /
devtmpfs 63G 0 63G 0% /dev
tmpfs 63G 4.0K 63G 1% /dev/shm
tmpfs 63G 1.4M 63G 1% /run
tmpfs 63G 0 63G 0% /sys/fs/cgroup
/dev/sda1 477M 112M 336M 25% /boot
tmpfs 13G 0 13G 0% /run/user/0
/dev/loop1 4.2G 4.2G 0 100% /var/usp-iso >>>>
```

6. Proceed to [Extract the UAS Bundle, on page 50](#).

## Extract the UAS Bundle

Once the USP ISO has been mounted, the UAS bundle must be extracted from the ISO in order to prepare the configuration files required for deployment.



### Important

These instructions assume you are already logged on to the server on which AutoIT, AutoDeploy, and VIM-Orchestrator VMs are to be installed and that the USP ISO has been mounted.

To extract the UAS bundle:

1. Navigate to the tools directory within the ISO mount.

```
cd /var/usp-iso/tools/
```

2. Launch the `usp-uas-installer.sh` script.

```
sudo ./usp-uas-installer.sh
```

The script extracts the files that comprise the UAS bundle to `/opt/cisco/usp/uas-installer`.

3. Verify that files have been extracted.

Example output:

```
ll /opt/cisco/usp/uas-installer
total 20
drwxr-xr-x 5 root root 4096 Aug 18 23:42 ./
drwxr-xr-x 6 root root 4096 Aug 18 23:42 ../
drwxr-xr-x 5 root root 4096 Aug 18 23:42 common/
drwxr-xr-x 2 root root 4096 Aug 18 23:42 images/
drwxr-xr-x 2 root root 4096 Aug 18 23:42 scripts/

ll /opt/cisco/usp/uas-installer/images/
total 711940
drwxr-xr-x 2 root root 4096 Aug 18 23:42 ./
drwxr-xr-x 5 root root 4096 Aug 18 23:42 ../
-rw-r--r-- 1 root root 729010688 Aug 17 23:29 usp-uas-1.0.0-1074.qcow2

ll /opt/cisco/usp/uas-installer/scripts/
total 80
-rwxr-xr-x. 1 root root 806 Aug 29 18:14 auto-deploy-booting.sh
-rwxr-xr-x. 1 root root 5460 Aug 29 18:14 autoit-user.py
-rwxr-xr-x. 1 root root 811 Aug 29 18:14 auto-it-vnf-staging.sh
-rwxr-xr-x. 1 root root 4762 Aug 29 18:14 encrypt_account.sh
```

```
-rwxr-xr-x. 1 root root 3945 Aug 29 18:14 encrypt_credentials.sh
-rwxr-xr-x. 1 root root 14031 Aug 29 18:14 start-ultram-vm.py
-rwxr-xr-x. 1 root root 14605 Aug 29 18:14 boot_uas.py
-rwxr-xr-x. 1 root root 5384 Aug 29 18:14 uas-check.py
-rwxr-xr-x. 1 root root 11283 Aug 29 18:14 usp-tenant.py
```

4. Proceed to [Deploy AutoIT, on page 51](#).

## Deploy AutoIT

AutoIT deployment is facilitated through a script. The script relies on user inputs to perform pre-requisite configurations including whether or not to deploy with HA support and account encryptions. Additionally, the script removes existing AutoIT deployments that may already exist.

The following information is required to execute the script:

- **AutoIT VM Login Password for ID 'ubuntu':** The password for the default user account, which is named ubuntu.
- **AutoIT API Access password for 'admin':** The password for the ConfD administrator user, which is named admin.
- **AutoIT API Access password for 'oper':** The password for the ConfD operator user, which is named oper.
- **AutoIT API Access password for 'security-admin':** The password for the ConfD security administrator user, which is named security-admin.
- **Hostname:** The hostname assigned to the AutoIT VM.
- **Image (QCOW2):** The path and file name for the UAS qcow2 file. For example:  
*/opt/cisco/usp/uas-installer/images/usp-uas-1.0.0-1074.qcow2*
- **External Network HA VIP :** The VIP address to be assigned to AutoIT's external network interface.
- **External Network Details:**
  - **IP Address:** The IP address to be assigned to AutoIT VMs' external network interface. If AutoIT is deployed with HA support, you are prompted to enter separate external IP addresses for both the active and redundant VMs.
  - **Gateway:** The gateway assigned to AutoIT's external network interface.
  - **Netmask:** The mask to be assigned to AutoIT's external network interface.
- **Provisional Network HA VIP:** The VIP address to be assigned to AutoIT's provisional network interface.
- **Provisioning Network Details:**
  - **IP Address:** The IP address to be assigned to the provisioning network interface. Within Hyper-Converged Ultra M models, this interface is used by the Ultra M Health Monitoring function. If AutoIT is deployed with HA support, you are prompted to enter separate IP provisioning addresses for both the active and redundant VMs.
  - **Netmask:** The netmask to be assigned to the provisioning network interface.

**Important**

- All passwords must meet the requirements specified in [Password Requirements and Login Security](#), on page 32.
- You may be asked for some of the above pieces of information twice, once for each VM when AutoIT is deployed with HA support.

---

The script allocates the following resources to the AutoIT VM:

- 2 VCPUs
- 8 GB RAM
- 80 GB Root Disk

**Important**


---

These instructions assume a bare-metal installation and that you are already logged on to the server on which AutoIT, AutoDeploy, and VIM-Orchestrator VMs are to be installed and on which the USP ISO has been mounted.

---

To deploy the AutoIT VM:

1. Navigate to the `/opt/cisco/usp/uas-installer/scripts` directory:

```
cd /opt/cisco/usp/uas-installer/scripts
```

2. Execute the `boot_uas.py` script with the desired options:

```
./boot_uas.py --kvm --autoit --ha
```

**Important**


---

The above command deploys AutoIT with HA support which is recommended for use within Ultra M solutions. Remove the `--ha` if you do not wish to implement HA support for AutoIT.

---

There are a number of options that can be specified when deploying AutoIT. Refer to [boot\\_uas.py Help](#), on page 179 for details. Some notes are below.

**Note**

- The above command deploys AutoIT with HA support. Remove the **--ha** if you do not wish to implement HA support for AutoIT.
- If you wish to configure syslog functionality for AutoIT, you must specify the IP address, TCP/UDP port, and severity level for one or more collection servers. The following command example configures two collection servers.

```
./boot_uas.py --kvm --autoit --ha --syslog-ip 192.168.2.1 --port 514
--severity 5 --syslog-ip 192.168.2.2 --port 514 --severity 5
```

You can set the severity level to one of the following values:

- 0: emerg, panic
- 1: alert
- 2: crit
- 3: err.error
- 4: warning, warn
- 5: notice
- 6: info
- 7: debug

### 3. Enter the information requested by the script for your deployment.

The script displays progress information. For example:

```
2018-01-24 16:06:17,355 - '/home' disk capacity is 1807 GB Loaded plugins: langpacks,
product-id
2018-01-24 16:06:17,397 - Package 'virt-install' is present
2018-01-24 16:06:17,397 - Package 'libvirt' is present
2018-01-24 16:06:17,397 - Package 'virt-viewer' is present
2018-01-24 16:06:17,397 - Interface 'br-ex' is UP
2018-01-24 16:06:17,397 - Interface 'br-ctlplane' is UP
2018-01-24 16:06:17,398 - Removing old deployment 'AutoIT_instance_0', if it exists
2018-01-24 16:06:19,921 - Removing old deployment 'AutoIT_instance_1', if it exists
2018-01-24 16:06:19,946 - Using instance 'AutoIT_instance_0' at location
'/home/cisco/AutoIT/instance_0'
2018-01-24 16:06:19,946 - Staging configuration ISO
2018-01-24 16:06:19,951 - Completed configuration ISO
/home/cisco/AutoIT/instance_0/cfg.iso
2018-01-24 16:06:19,951 - Preparing root disk '/home/cisco/AutoIT/instance_0/uas.qcow2'
2018-01-24 16:06:20,378 - Resizing disk to '80GB'
2018-01-24 16:06:33,417 - Starting deployment 'AutoIT_instance_0'
2018-01-24 16:06:34,124 - Started deployment 'AutoIT_instance_0' successfully
2018-01-24 16:06:34,125 - Using instance 'AutoIT_instance_1' at location
'/home/cisco/AutoIT/instance_1'
2018-01-24 16:06:34,125 - Staging configuration ISO
2018-01-24 16:06:34,130 - Completed configuration ISO
/home/cisco/AutoIT/instance_1/cfg.iso
2018-01-24 16:06:34,130 - Preparing root disk '/home/cisco/AutoIT/instance_1/uas.qcow2'
2018-01-24 16:06:34,557 - Resizing disk to '80GB'
2018-01-24 16:06:42,629 - Starting deployment 'AutoIT_instance_1'
2018-01-24 16:06:43,360 - Started deployment 'AutoIT_instance_1' successfully
```

4. Verify that the AutoIT VM is running.

```
virsh list -all
```

Example command output:

Id	Name	State
487	AutoIT_instance_0	running
488	AutoIT_instance_1	running

5. Check the status of AutoIT.

- a. Log on to the master AutoIT VM.

```
confd_cli -C -u admin
```

Example command output:

```
Welcome to the ConfD CLI
admin connected from 127.0.0.1 using console on autoit1-0
```

- b. Enter the *admin* user password when prompted.

- c. View the status.

```
show uas
```

Example command output:

```
uas version                6.0.0
uas state                   active
uas external-connection-point 172.28.185.132
INSTANCE IP      STATE  ROLE
-----
172.28.185.133  alive  CONF-D-MASTER
172.28.185.134  alive  CONF-D-SLAVE

NAME                LAST HEARTBEAT
-----
AutoIT-MASTER      2018-01-24 21:24:30
USPCFMWorker       2018-01-24 21:24:30
USPCHBWorker       2018-01-24 21:24:30
USPCWorker         2018-01-24 21:24:30
```

6. Proceed to [Deploy AutoDeploy, on page 54](#).

## Deploy AutoDeploy



### Important

The information and instructions provided here are only applicable when AutoDeploy is used in the VIM Orchestrator installation process.

AutoDeploy deployment is facilitated through a script. The script relies on user inputs to perform pre-requisite configurations including whether or not to deploy with HA support and account encryptions. Additionally, the script removes existing AutoDeploy deployments that may already exist.

The following information is required to execute the script:

- **AutoDeploy VM Login Password for ID 'ubuntu'** The password for the default user account, which is named ubuntu.

- **AutoDeploy API Access password for 'admin':** The password for the ConfD administrator user, which is named admin.
- **AutoDeploy API Access password for 'oper':** The password for the ConfD operator user, which is named oper.
- **AutoDeploy API Access password for 'security-admin':** The password for the ConfD security administrator user, which is named security-admin.
- **Hostname:** The hostname assigned to the AutoDeploy VM.
- **Image (QCOW2):** The path and file name for the UAS qcow2 file. For example:  
*/opt/cisco/usp/uas-installer/images/usp-uas-1.0.0-1074.qcow2*
- **External Network HA VIP :** The VIP address to be assigned to AutoDeploy's external network interface.
- **External Network Details:**
  - **IP Address:** The IP address to be assigned to AutoDeploy VMs' external network interface. If AutoDeploy is deployed with HA support, you are prompted to enter separate external IP addresses for both the active and redundant VMs.
  - **Gateway:** The gateway assigned to AutoDeploy's external network interface.
  - **Netmask:** The mask to be assigned to AutoDeploy's external network interface.




---

**Important**

- All passwords must meet the requirements specified in [Password Requirements and Login Security](#), on page 32.
- You may be asked for some of the above pieces of information twice, once for each VM when AutoDeploy is deployed with HA support.

---

The script allocates the following resources to the AutoDeploy VM:

- 2 VCPUs
- 8 GB RAM
- 80 GB Root Disk




---

**Important**

These instructions assume a bare-metal installation and that you are already logged on to the server on which AutoIT, AutoDeploy, and VIM-Orchestrator VMs are to be installed and on which the USP ISO has been mounted.

---

To deploy the AutoDeploy VM:

1. Navigate to the */opt/cisco/usp/uas-installer/scripts* directory:  
**cd /opt/cisco/usp/uas-installer/scripts**
2. Execute the *boot\_uas.py* script:

```
./boot_uas.py --kvm --autodeploy --ha
```



**Note** The above command deploys AutoDeploy with HA support. Remove the **--ha** if you do not wish to implement HA support for AutoDeploy.

There are a number of options that can be specified when deploying AutoDeploy. Refer to [boot\\_uas.py Help, on page 179](#) for details. Some notes are below.



**Note**

- The above command deploys AutoDeploy with HA support. Remove the **--ha** if you do not wish to implement HA support for AutoDeploy.
- If you wish to configure syslog functionality for AutoDeploy, you must specify the IP address, TCP/UDP port, and severity level for one or more collection servers. The following command example configures two collection servers.

```
./boot_uas.py --kvm --autodeploy --ha --syslog-ip 192.168.2.1 --port 514
--severity 5 --syslog-ip 192.168.2.2 --port 514 --severity 5
```

You can set the severity level to one of the following values:

- 0: emerg, panic
- 1: alert
- 2: crit
- 3: err.error
- 4: warning, warn
- 5: notice
- 6: info
- 7: debug

### 3. Enter the information requested by the script for your deployment.

The script displays progress information. For example:

```
2018-01-24 16:28:05,095 - '/home' disk capacity is 1807 GB Loaded plugins: langpacks,
product-id
2018-01-24 16:28:05,134 - Package 'virt-install' is present
2018-01-24 16:28:05,135 - Package 'libvirt' is present
2018-01-24 16:28:05,135 - Package 'virt-viewer' is present
2018-01-24 16:28:05,135 - Interface 'br-ex' is UP
2018-01-24 16:28:05,135 - Interface 'br-ctlplane' is UP
2018-01-24 16:28:05,135 - Removing old deployment 'AutoDeploy_instance_0', if it exists
2018-01-24 16:28:06,980 - Removing old deployment 'AutoDeploy_instance_1', if it exists
2018-01-24 16:28:07,005 - Using instance 'AutoDeploy_instance_0' at location
'/home/cisco/AutoDeploy/instance_0'
2018-01-24 16:28:07,006 - Staging configuration ISO
2018-01-24 16:28:07,010 - Completed configuration ISO
/home/cisco/AutoDeploy/instance_0/cfg.iso
2018-01-24 16:28:07,010 - Preparing root disk
```

```

'/home/cisco/AutoDeploy/instance_0/uas.qcow2'
2018-01-24 16:28:07,450 - Resizing disk to '80GB'
2018-01-24 16:28:15,965 - Starting deployment 'AutoDeploy_instance_0'
2018-01-24 16:28:16,649 - Started deployment 'AutoDeploy_instance_0' successfully
2018-01-24 16:28:16,650 - Using instance 'AutoDeploy_instance_1' at location
'/home/cisco/AutoDeploy/instance_1'
2018-01-24 16:28:16,650 - Staging configuration ISO
2018-01-24 16:28:16,655 - Completed configuration ISO
/home/cisco/AutoDeploy/instance_1/cfg.iso
2018-01-24 16:28:16,655 - Preparing root disk
'/home/cisco/AutoDeploy/instance_1/uas.qcow2'
2018-01-24 16:28:17,106 - Resizing disk to '80GB'
2018-01-24 16:28:30,204 - Starting deployment 'AutoDeploy_instance_1'
2018-01-24 16:28:30,892 - Started deployment 'AutoDeploy_instance_1' successfully

```

#### 4. Verify that the AutoDeploy VM is running.

```
virsh list -all
```

Id	Name	State
495	AutoDeploy_instance_0	running
496	AutoDeploy_instance_1	running



#### Important

It is recommended that you do not make any changes to the AutoIT network interface or bridge configuration. Doing so will require that you redeploy AutoDeploy.

#### 5. Check the status of AutoDeploy.

##### a. Log on to the master AutoDeploy VM.

```
confd_cli -C -u admin
```

Example command output:

```

Welcome to the ConfD CLI
admin connected from 127.0.0.1 using console on autodeploy-0

```

##### b. Enter the *admin* user password when prompted.

##### c. View the status.

```
show uas
```

Example command output:

```

uas version          6.0.0uas version          6.0.0
uas state            active
uas external-connection-point 172.28.185.132
INSTANCE IP        STATE  ROLE
-----
172.28.185.133    alive  CONFD-MASTER
172.28.185.134    alive  CONFD-SLAVE

NAME                LAST HEARTBEAT
-----
AutoDeploy-MASTER  2018-01-24 21:29:54
USPCFMWorker        2018-01-24 21:29:45
USPCHBWorker        2018-01-24 21:29:45
USPCWorker          2018-01-24 21:29:45

```

#### 6. Choose the desired method by which to continue the deployment process:

- Use the ConfD CLI/APIs to continue the deployment process. To use this method, proceed to [Prepare the VIM Orchestrator and VIM Configuration File, on page 58](#).




---

**Important** You will need access to both the OpenStack GUI and CLI to complete the configuration procedures.

---

## Prepare the VIM Orchestrator and VIM Configuration File

As described in [VIM Installation Automation Overview, on page 40](#), the VIM Orchestrator and VIM configuration file is used by AutoDeploy to activate the OSP-D VM and VIM deployment process.

This file includes all of the configuration information required to deploy OSP-D VM and VIM including configurations for constructs such as secure tokens, package images, NFVI point-of-presence descriptors (nfvi-popd), the VIM Orchestrator descriptor (vim-orchd), and VIM role and node information. Refer to [Sample VIM Orchestrator and VIM Configuration File, on page 183](#) for more information. Additional information on the constructs and parameters used in this file are located in the *Cisco Ultra Services Platform NETCONF API Guide*.

You can also refer to RedHat user documentation for information on how to install the satellite server if your deployment requires:

[https://access.redhat.com/documentation/en-US/Red\\_Hat\\_Network\\_Satellite/5.0/html/Installation\\_Guide/s1-intro-sat.html](https://access.redhat.com/documentation/en-US/Red_Hat_Network_Satellite/5.0/html/Installation_Guide/s1-intro-sat.html)




---

**Note** These instructions assume you are already logged on to the AutoDeploy VM as the root user.

---

To prepare the VIM Orchestrator and VIM configuration file:

1. Create and edit your VIM Orchestrator and VIM configuration file according to your deployment requirements. Use the sample provided in [Sample VIM Orchestrator and VIM Configuration File, on page 183](#) as a reference.
2. Save the VIM Orchestrator and VIM configuration file you have created to your home directory.
3. Proceed to [Activate the VIM Orchestrator and VIM Deployment, on page 58](#).

## Activate the VIM Orchestrator and VIM Deployment

Once you have completed preparing your VIM Orchestrator and VIM configuration file, you must load the configuration and activate the deployment in order to bring up the OSP-D VM and the VIM.




---

**Important** These instructions assume you are already logged on to the AutoDeploy VM as the root user and that your VIM Orchestrator and VIM configuration file has been prepared for your deployment as per the information and instructions in [Prepare the VIM Orchestrator and VIM Configuration File, on page 58](#).

---

To activate the OSP-D VM and VIM deployment using AutoDeploy:

1. Login to the ConfD CLI as the *admin* user.

```
confd_cli -u admin -C
```

2. Enter the *admin* user password when prompted.
3. Enter the ConfD configuration mode.

```
config
```

4. Load the VIM Orchestrator and VIM configuration file to provide the deployment artifacts to the VIM.

```
load merge <your_config_file_name>.cfg
commit
end
```




---

**Important**

If changes are made to the VIM Orchestrator and VIM configuration file after it was committed, you can apply the changes using the **load replace** command instead of the **load merge** command. You will also need to **commit** your changes.

---

5. Activate the VIM Orchestrator configuration aspects of the configuration file.

```
activate nsd-id <nsd_name>
```




---

**Important**

The output of this command is a transaction-id which can be used to monitor the deployment progress. If need be, the VIM deployment can be deactivated using the **deactivate** variant of this command.

---

6. Monitor the progress of the deployment.
  - a. List the transactions.

```
show transaction
```

Example command output:

TX ID	TX TYPE	ID	DEPLOYMENT	TIMESTAMP
STATUS	DETAIL	STATUS		
1510448403-721303	activate-ns-deployment		test	
2017-11-12T01:00:03.721334-00:00		requested	-	
1510448404-104189	activate-vim-orch-deployment		ph-vim-orch	
2017-11-12T01:00:04.104204-00:00		requested	-	

- b. Monitor the transaction log.

```
show log tx-id
```

Example command output:

```
show log tx-id
transaction 1510448403-721303
tx-type activate-ns-deployment
deployment-id test
timestamp 2017-11-12T01:00:03.721334-00:00
status success
transaction 1510448404-104189
tx-type activate-vim-orch-deployment
deployment-id ph-vim-orch
```

```
timestamp    2017-11-12T01:00:04.104204-00:00
status      success
```

- c. Check the VIM Orchestrator status.

**show vim-orchr**

Example command output:

```
vim-orch status success
vim-orch steps-total 84
vim-orch steps-completed 84
vim-orch version "Red Hat OpenStack Platform release 10.0 (Newton)"
```




---

**Important**

If there are any issues seen when executing the above commands, refer to [Monitoring and Troubleshooting the Deployment, on page 100](#) for information on collecting logs.

---

7. Upon successful completion of the VIM deployment, proceed to [VNF Deployment Automation, on page 60](#) for information and instructions on deploying your USP-based VNF.

## VNF Deployment Automation

### VNF Deployment Automation Overview

USP-based VNF deployment automation is performed through UAS.




---

**Important**

This information in these sections assume that all of the [USP Installation Prerequisites, on page 29](#) and/or that all requirements identified in the *Ultra M Solutions Guide* have been met.

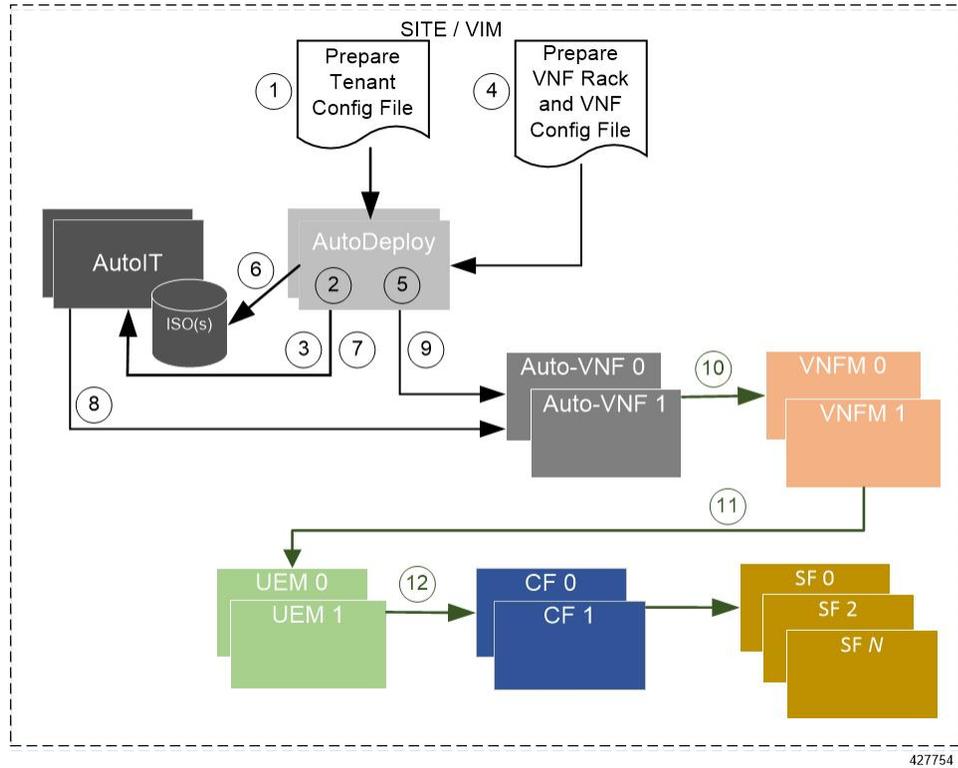
---

[Figure 19: UAS Deployment Automation Workflow for a Single VNF, on page 61](#) and [Figure 20: UAS Deployment Automation Workflow for a Multi-VNF, on page 62](#) provide an overview of the VNF deployment automation process for Hyper-Converged Ultra M deployments. Details are provided in [Table 15: VNF Deployment Automation Workflow Descriptions, on page 62](#).

Notes:

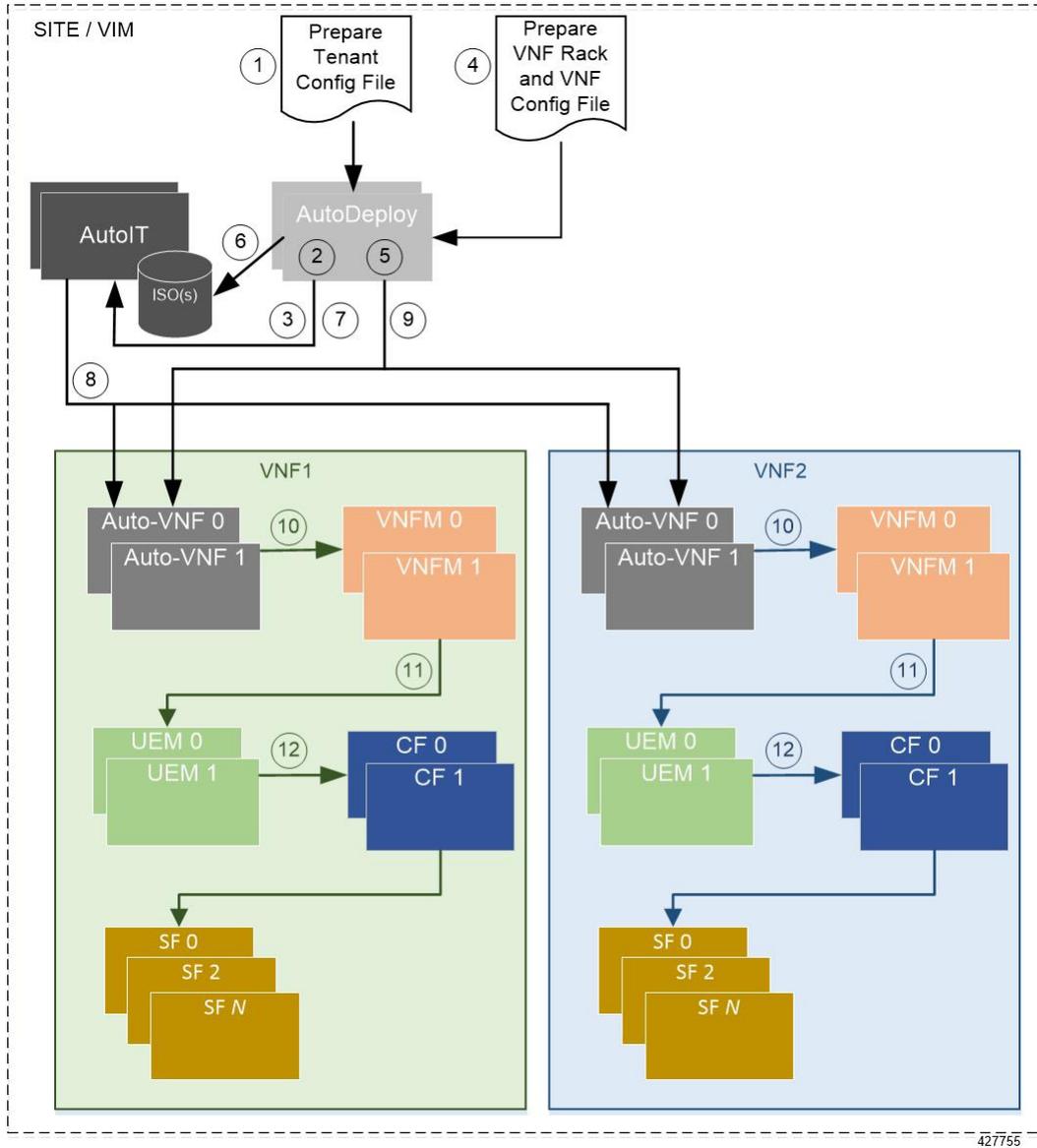
- The workflow described in this section is supported only with Ultra M deployments based on OSP 10 and that leverage the Hyper-Converged architecture.
- This information assumes that you have deployed the VIM Orchestrator and the VIM using the information and instructions in [Virtual Infrastructure Manager Installation Automation, on page 39](#).

Figure 19: UAS Deployment Automation Workflow for a Single VNF



427754

Figure 20: UAS Deployment Automation Workflow for a Multi-VNF



427755

Table 15: VNF Deployment Automation Workflow Descriptions

Callout	Description
1	Prepare the tenant configuration file. Tenants define the physical resources allocated to specific user groups.
2	Load and activate the tenant configuration file through AutoDeploy. Tenants must be configured after the VIM has been deployed and before deploying the VNF.
3	AutoDeploy works with AutoIT to provision the tenants within the VIM.

Callout	Description
4	<p>Prepare the VNF Rack and VNF descriptor configuration files. The parameters in this file identify the Compute Nodes to be used for VNF deployment and the VNFDs for AutoVNF, the VNFM, and for the UGP VNF.</p> <p>Refer to <a href="#">Sample VNF Rack and VNF Descriptor Configuration File, on page 199</a> for an example VNF Rack and VNF descriptor configuration file.</p>
5	<p>On the AutoDeploy VM, load and commit the configuration file prepared in previous step. Once committed, activate the previously loaded AutoDeploy configuration file. AutoDeploy processes this data to activate the Site and to deploy the functions needed to orchestrate the VNF deployment. Refer to <a href="#">Configure the VNF Rack and the VNF Descriptors, on page 68</a> for more information.</p>
6	<p>AutoDeploy loads the ISO on to AutoIT.</p>
7	<p>AutoDeploy passes data from the activated configuration to AutoIT requesting that it deploy the AutoVNF VM cluster for the initial VNF.</p>
8	<p>AutoIT deploys the AutoVNF VMs for the VNF.</p>
9	<p>Once the AutoVNF VMs are successfully deployed, AutoDeploy passes configuration information to the AutoVNF VMs. This is used by the AutoVNF to orchestrate the VNFM deployment.</p> <p>For deployments with multiple VNFs, AutoDeploy sends this information in parallel to the active AutoVNF VM for each VNF.</p> <p><b>Important</b> In 6.0, concurrent VNF deployment functionality was not fully qualified and was made available only for testing purposes. In 6.1 and later releases, this functionality is fully qualified. For more information, contact your Cisco Accounts representative.</p>
10	<p>The active AutoVNF software module leverages the network service descriptor (NSD) information to work with the VIM to deploy the VNFM VMs.</p> <p>Once the VNFM VMs are successfully deployed, AutoVNF also ensures that the various VM catalogs pertaining to other VNFCs are on-boarded by the VNFM. It accomplishes this through a number of YANG-based definitions which are then used to configure various aspects of the virtualized environment using REST and NETCONF APIs.</p> <p>The VNFM mounts the VNFC catalogs and works with AutoVNF to deploy the various components that comprise the desired VNF use-case (e.g. UGP or USF).</p>
11	<p>The VNFM leverages the VNFD information to deploy the UEM VM cluster.</p> <p>Though the USP architecture represents a single VNF to other network elements, it is comprised of multiple VM types each having their own separate catalogs. The UEM component of the USP works with the VNFM to deploy these catalogs based on the intended VNF use case (e.g. UGP, USF, etc.).</p>

Callout	Description
12	<p>The UEM processes the Day-0 configuration information it received from the VNFM and deploys the Control Function (CF) and Service Function (SF) VNFC VMs.</p> <p>Once all of the VNF components (VNFCs) have been successfully deployed, AutoVNF notifies AutoDeploy.</p>

## Pre-VNF Installation Verification

Prior to installing the USP, please ensure that the following is true:

- The prerequisite hardware is installed and operational with network connectivity.
- The prerequisite software is installed, configured and functioning properly.
  - You have administrative rights to the operating system.
  - VIM Orchestrator is properly installed and operational.
  - VIM components are properly installed and operational. This configuration includes networks, flavors, and sufficient quota allocations to the tenant.




---

**Important** Supported and/or required flavors and quota allocations are based on deployment models. Contact your Cisco representative for more information.

---

- You have administrative rights to the OpenStack setup.
- The Cisco USP software ISO has been downloaded and is accessible by you.

## Deploy the USP-based VNF

The software roles that comprise the Ultra Automation Services (UAS) are used to automate the USP-based VNF deployment. UAS is designed to support deployment automation for all USP-based VNF scenarios.




---

**Important** Cisco's Elastic Services Controller (ESC) is the only VNFM supported in this release.

---




---

**Important** These instructions assume that you have verified the requirements identified in [Pre-VNF Installation Verification, on page 64](#).

---

## Configure VIM Tenants

VIM tenants define the physical resources allocated to specific user groups. They are provisioned by executing an API through AutoDeploy which processes parameters provided in a configuration file. The parameters are

grouped into a tenant descriptor which is referenced within the VIM Artifact descriptor. Tenants must be configured after the VIM has been deployed and before deploying the VNF.

In 6.3, the multi-tenants are supported per VNF but this feature was not fully qualified. It was made available only for testing purposes. In release 6.4, this functionality has been fully qualified. Refer to [Sample Tenant Configuration File, on page 187](#) for an example tenant configuration file.




---

**Important** These instructions assume you are already logged on to the AutoDeploy VM as the root user.

---

To configure VIM tenants:

1. Prepare the tenant configuration file according to your deployment scenario.




---

**Caution** Ensure that the NSD name (nsd-id) specified in the configuration file is identical to the NSD name specified in the VIM Orchestrator and VIM configuration file. Additionally, ensure that only tenant information is referenced within the VIM artifact descriptor.

---

2. Login to the ConfD CLI as the *admin* user.

```
confd_cli -u admin -C
```

3. Enter the *admin* user password when prompted.

4. Enter the ConfD configuration mode.

```
config
```

5. Load the *addi-tenant.cfg* configuration file.

```
load merge <your_tenant_file_name>.cfg
```

```
commit
```

```
end
```

6. Activate the tenant configuration.

```
activate nsd-id <nsd_name>
```




---

**Important** The output of this command is a transaction-id which can be used to monitor the deployment progress. If need be, the tenant configuration can be deactivated using the **deactivate** variant of this command.

---

7. Monitor the progress of the tenant creation by viewing transaction logs:

```
show log <transaction_id> | display xml
```

*transaction\_id* is the ID displayed as a result of the **activate** command executed in step 6, on page 65.

8. Verify that the tenants have been created properly by checking the network service record (NSR).

```
show nsr
```

Example command output:

NSR ID	NSD	VNFR	VNF PACKAGE	VNF RACK	VIM ORCH	VIM	VIM ARTIFACT	VLR ID	NETWORK	VNFR
sjc-instance	sjc	-	-	-	underc	overc	sjccore			

- Proceed to [Configure OpenStack Prerequisites, on page 66](#).

## Configure OpenStack Prerequisites

Prior to beginning the USP deployment process, there are several items as described in this section that must first be configured in OpenStack. The deployment automation process requires these items be configured in order for the UAS processes to run properly.



### Important

This procedure is not necessary if you are deploying a VNF on a Hyper-Converged Ultra M mode and have already deployed the VIM Orchestrator and the VIM using the information and instructions in [Virtual Infrastructure Manager Installation Automation, on page 39](#).



### Important

The information in this section assumes that your Undercloud and Overcloud were previously installed and are operational as identified in [Pre-VNF Installation Verification, on page 64](#).

Ensure that the following items are configured within the OpenStack CLI interface on the OSP-D Server / Staging Server:

- Login to OSP-D and make sure to “su - stack” and “source stackrc”. Determine the name of the heat stack\_name.
 

```
heat stack-list
```
- Source the rc file for the stack.
 

```
source ~/<stack_name> rc
```
- Login to OpenStack Horizon with the tenant and username created in [Configure VIM Tenants, on page 64](#) and download the credential file.
- Source the “stack\_namerc-core” file.
 

```
source ~/<stack_name> rc-core
```
- Create the volume type for your deployment.
 

```
cinder type-create LUKS
```
- Determine the tenant ID for the OpenStack core project.
 

```
openstack project list | grep core
```
- Create a neutron router for the core tenant called “main” and associate it with the core project tenant ID.
 

```
neutron router-create main --tenant-id <core_tenant_id>
```
- Create a public/“external” network.

```
neutron net-create public --router:external True
--provider:physical_network datacentre --provider:network_type vlan
--provider:segmentation_id <vlan_segmentation_id>
```

**Important**

<vlan\_segmentation\_id> is based on your OpenStack configuration and must match the VLAN ID specified for the network.

```
neutron subnet-create public <network_address>/<mask-bits> --name
public-subnet --allocation-pool start= <start_address>, end=<end_address>
--disable-dhcp --gateway <gateway_address>
neutron router-gateway-set main public
```

**Important**

It is recommended that you assign a static IP address to your router if your VNF configuration uses floating IP addresses in order to avoid potential IP address conflicts. The IP address is assigned based on the subnet created for floating IPs on the network. Floating IP address support is configured at the VNFD-level within the AutoDeploy configuration using the **floating-ip enabled** and **floating-ip ip-address** parameters. Static addresses can be assigned to the router using the following command:

```
neutron router-gateway-set main public --fixed-ip subnet_id=`neutron
subnet-list |grep public | awk '{print $2}'`,ip_address= <static_ip_address>
```

9. Create SRIOV networks for use with the DI-internal and Service networks.

- a. Create the SR-IOV network for DI-internal network.

```
neutron net-create di-internal1 --provider:physical_network
phys_pcie1_0 --provider:network_type flat --shared
neutron subnet-create di-internal1 <network_address>/<mask-bits>--name
di-internal1-subnet --enable-dhcp
```

- b. Repeat step 9.a, on page 67 for the redundant DI-network in the case where NIC bonding is used.

```
neutron net-create di-internal2 --provider:physical_network
phys_pcie4_1 --provider:network_type flat --shared
neutron subnet-create di-internal2 <network_address>/<mask-bits>--name
di-internal2-subnet --enable-dhcp
```

- c. Create the SR-IOV network for Service 1.

```
neutron net-create service1 --provider:physical_network phys_pcie1_1
--provider:network_type flat --shared
neutron subnet-create service1 <network_address>/<mask-bits>--name
service1-subnet --enable-dhcp
```

- d. Repeat step 9.d, on page 67 for the redundant Service in the case where NIC bonding is used.

```
neutron net-create service2 --provider:physical_network phys_pcie4_0
--provider:network_type flat --shared
neutron subnet-create service2 <network_address>/<mask-bits>--name
service2-subnet --enable-dhcp
```

- e. Repeat steps [9.c, on page 67](#) and [9.d, on page 67](#) for each additional Service network required for your deployment.
10. Proceed to [Configure the VNF Rack and the VNF Descriptors, on page 68](#).

## Configure the VNF Rack and the VNF Descriptors

Once the VIM tenants and OpenStack prerequisites have been configured, the VNF Rack and VNF descriptors must be configured. Once these items are configured, your VNF can be deployed.

The VNF Rack descriptor is a logical grouping of Compute Nodes. It is used to map the nodes to specific VNFs. It is equivalent to Availability Zones and/or Host Aggregates in OpenStack. Like tenants, VNF Rack descriptors are configured at the network service descriptor (NSD) level and is referenced within the VIM artifact descriptor.

The VNF descriptor (VNFD) defines the deployment flavor for a specific VNF including all the aspects of VNF resources and associated networking. A single NSD can contain multiple VNFDs. For example, a configuration for deploying a UGP VNF on Ultra M will have separate VNFDs for:

- AutoVNF (one instance per VNF)
- VNFM
- UGP VNF

These VNFDs are defined under nested NSDs, one per VNF, within the file. Each VNF must be defined by its own NSD. The file also contains additional parameters related to and required by your specific deployment scenario. These are a mix of basic, operational parameters and enhanced features supported within the USP VNF deployment on the Ultra M solution. For more information on enhanced features, refer to:

- [Monitoring and Recovering AutoVNF Through AutoIT, on page 167](#)
- [Monitoring and Recovering VNFC Through AutoVNF, on page 168](#)
- [Configuring Fully-Defined VM Names for ESC, on page 70](#)
- The "Health Monitoring Within the Ultra M Solution" chapter of the *Ultra M Solutions Guide*

Refer to [Sample VNF Rack and VNF Descriptor Configuration File, on page 199](#) for an example VNF Rack and VNF descriptor configuration file. Detailed information for the parameters used in the configuration constructs within the file is provided in the *Cisco Ultra Services Platform NETCONF API Guide*.



### Important

User credentials are configured through Secure Tokens specified in the configuration file. Ensure that passwords configured with Secure Token meet the requirements specified in [Password Requirements and Login Security, on page 32](#).



### Important

These instructions assume you are already logged on to the AutoDeploy VM as the root user.

To configure the VNF Rack and VNF descriptor file:

1. Prepare the VNF Rack and VNF descriptor configuration file according to your deployment scenario.

**Caution**

Ensure that the NSD name (nsd-id) specified in the configuration file is identical to the NSD name specified in the VIM Orchestrator/VIM and tenant configuration files.

- a. Add the VNF Rack descriptor parameter configuration to the file and reference it within the VIM Artifact descriptor.
- b. Add the VNFD and other construct parameter configurations to the file as required for your deployment.
- c. *Optional.* If required, add any other additional parameters to configure the enhanced features like - AutoVNF Health Check, VNFC Health Check, Passing Fully-Defined VM Names, syslog proxy functionality, and so on.

Example Configuration for Health Check features:

```
nsd <nsd_name>
...
vnfd <vnfd_name>
...
  vnfc <vnfc_name>
    health-check enabled
    health-check probe-frequency 10
    health-check probe-max-miss 6
    health-check retry-count 6
    health-check recovery-type restart-then-redeploy
    health-check boot-time 300
...

```

Example Configuration for Passing VM Names:

```
nsd <nsd_name>
<---SNIP--->
vnfd <uas_vnfd_name>
vnf-type usp-uas
...
configuration set-vim-instance-name true
<---SNIP--->
vnfd <vnfm_vnfd_name>
vnf-type esc
...
configuration set-vim-instance-name true
<---SNIP--->

```

Example syslog proxy functionality:

```
nsd <nsd_name>
...
vnfd <vnfd_name>
...
  vnfc <vnfc_name>
    uas-proxy
...

```

- d. Save the file.
2. Login to the ConfD CLI as the *admin* user.  
**confd\_cli -u admin -C**
  3. Enter the *admin* user password when prompted.
  4. Enter the ConfD configuration mode.

```
config
```

5. Load the VNF Rack and VNF descriptor configuration file.

```
load merge <your_config_file_name>.cfg
```

```
commit
```

```
end
```

6. Activate the VNF rack and VNF descriptor configuration.

```
activate nsd-id<nsd_name>
```



### Important

The output of this command is a transaction-id which can be used to monitor the deployment progress. If needed, the VIM deployment can be deactivated using the **deactivate** variant of this command.

7. Monitor the progress of the tenant creation by viewing transaction logs:

```
show log <transaction_id> | display xml
```

*transaction\_id* is the ID displayed as a result of the command executed in step 6, on page 70.

8. Verify that the VNFs have been created properly by checking the network service record (NSR).

```
show nsr
```

Example command output:

NSR ID	VLR		VNF	VIM	NSR ID	VLR NETWORK	VNFR
	NSD	VNF RACK	VIM ORCH	VIM ARTIFACT			
ab-autovnf-instance ab-autovnf-vpc-up1 ]	ab-autovnf [ usp_6_0 ]	-	[ ab-autovnf-vnfl-em -	ab-autovnf-vnfl-esc vim_art_rack	-	-	-
ss-autoit-instance [ usp_6_0 ]	ss-autoit -	-	[ ss-autoit-f-autovnf ] -	vim_art_rack	-	-	-

## Configuring Fully-Defined VM Names for ESC

Leveraging capabilities in the VNFM (ESC version 3.0 and later), UAS supports the ability to generate and pass VM names to the VNFM. This is applicable to all VMs deployed on OpenStack including ESC and AutoVNF.

VM name generation is based on known algorithms using the following parameters:

- VNF Component (VNFC) name
- Network Service Descriptor (NSD) name
- VNF Descriptor (VNFD) name

- VIM tenant name

VM instance names are assembled as follows:

For CF/SF:

```
<nsd_name>-<vnfd_name>-<vim_tenant_name>-<vdu_id>-X
```

For example: abcUAS-LBPCF401-UGP-core-LBPCF401-CF-VDU-0

For UEM:

```
<nsd_name>-<vnfd_name>-<vim_tenant_name>-<vnfc_instance_id>-X
```

For example: abcUAS-LBPCF401-UGP-core-EM1-1




---

**Important**

There may be one or more VNFC instances depending on the redundancy and scaling.

This functionality is controlled in the UAS through a YANG-based configurable in the VNFD.




---

**Important**

In 6.0, this functionality was not fully qualified and was made available only for testing purposes. In 6.1 and later releases, this functionality is fully qualified. For more information, contact your Cisco Accounts representative.

```
nsd <nsd_name>
<---SNIP--->
  vnfd <uas_vnfd_name>
    vnf-type      usp-uas
    ...
configuration set-vim-instance-name true
<---SNIP--->
  vnfd <vnfm_vnfd_name>
    vnf-type      esc
    ...
    configuration set-vim-instance-name true
<---SNIP--->
```

This functionality is enabled (e.g. set to “true”) by default.





## CHAPTER 4

# Deploying VNFs Using AutoVNF

This chapter describes the following topics:

- [Introduction, on page 73](#)
- [VNF Deployment Automation Overview, on page 73](#)
- [Pre-VNF Installation Verification, on page 77](#)
- [Deploy the USP-based VNF, on page 77](#)
- [Upgrading/Redeploying the Stand-alone AutoVNF VM Instance, on page 87](#)

## Introduction

USP-based VNFs can be deployed using a stand-alone AutoVNF instance in environments with a pre-installed network function virtualization orchestrator (NFVO). In this scenario, a single AutoVNF VM is deployed on the VIM and communicates with a pre-existing VNFM installation to deploy the VNF(s). The VNFM can be installed in a tenant other than the one in which AutoVNF is installed.



### Important

Cisco Elastic Services Controller (ESC) is the only VNFM supported in this release.

A single AutoVNF VM can deploy one or more VNFs in one or more tenants within the same VIM.

## VNF Deployment Automation Overview

[Figure 21: AutoVNF Deployment Automation Workflow for a Single VNF, on page 74](#) and [Figure 22: AutoVNF Deployment Automation Workflow for a Multi-VNF, on page 75](#) provide an overview of the VNF deployment automation process for when using a stand-alone AutoVNF instance. Details are provided in [Table 16: VNF Deployment Automation Workflow Descriptions, on page 75](#).

### NOTES:

- The workflow described in this section is supported only with VNF deployments performed through AutoVNF and that are based on OSP 10.
- This information assumes that you have deployed the NFVI, VIM, and VNFM.
- This information assumes that all artifacts required during configuration must be pre-created in OpenStack.

Figure 21: AutoVNF Deployment Automation Workflow for a Single VNF

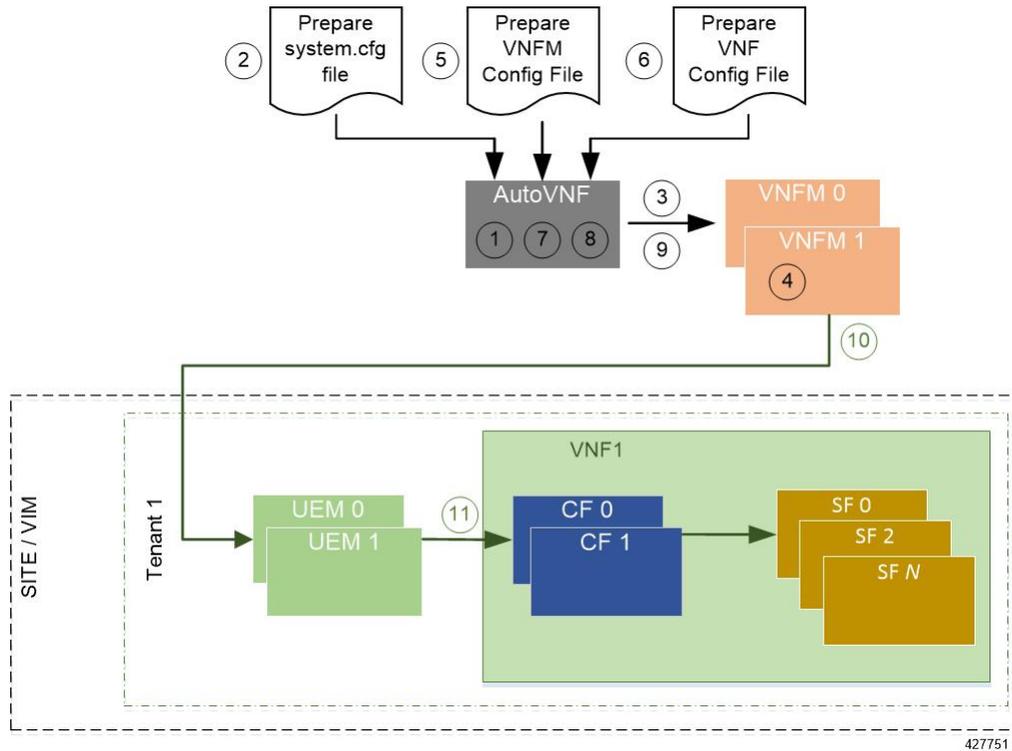
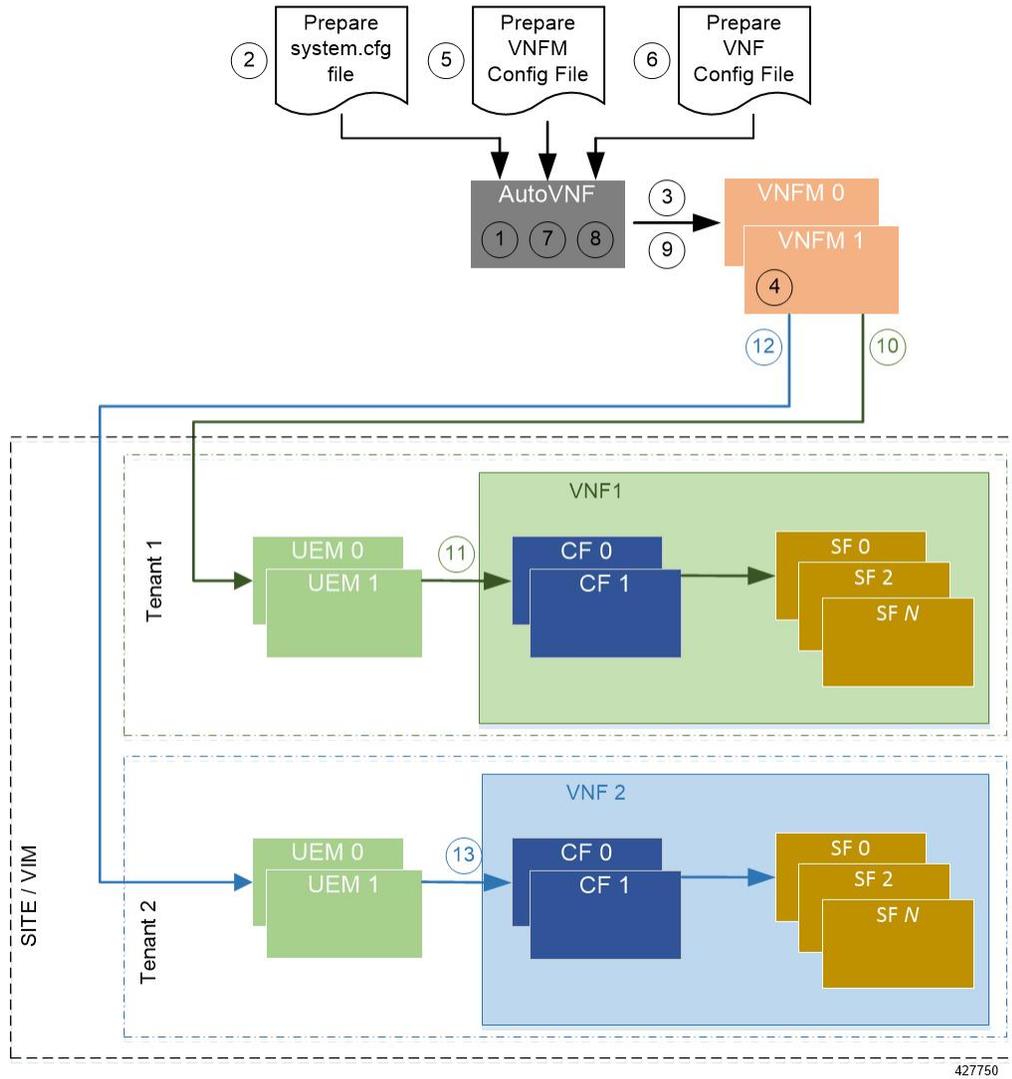


Figure 22: AutoVNF Deployment Automation Workflow for a Multi-VNF



427750

Table 16: VNF Deployment Automation Workflow Descriptions

Callout	Description
1	On the Onboarding Server, deploy AutoVNF using the <i>boot_uas.py</i> script provided as part of the release ISO.  Refer to <a href="#">Deploy the AutoVNF VM, on page 82</a> for more information. The release ISO also includes the software images for the VNFM and VNFs.
2	Prepare the system.cfg file to the AutoVNF VM. This file provides the VNF’s Day-0 configuration.  Refer to <a href="#">Sample system.cfg File, on page 211</a> for an example configuration file.

Callout	Description
3	Copy the VNFM scripts supplied in the UAS ISO from the AutoVNF VM to the VNFM VMs.
4	<p>Confirm that the VNFM has been configured with the VIM connectors for the VNF tenants. A connector is needed for each tenant.</p> <p>Refer to <a href="#">Sample ESC VIM Connector Configuration, on page 213</a> for an example configuration.</p>
5	<p>Prepare the VNFM configuration file that provides AutoVNF with the necessary information for communicating with a pre-existing VNFM installation.</p> <p>Refer to <a href="#">Sample AutoVNF VNFM Configuration File, on page 215</a> for more information.</p>
6	<p>Prepare the VNF configuration file that is used by AutoVNF to initiate the VNF deployment process.</p> <p>This file includes all of the configuration information required to deploy all of the VNF components (VNFCs) such as secure tokens, network catalogs, VDU catalogs, and VDUs.</p> <p>Refer to <a href="#">Sample AutoVNF VNF Configuration File, on page 217</a> for more information.</p>
7	<p>On the AutoVNF VM, load and commit the VNFM configuration file prepared in the previous step. Once committed, activate the loaded AutoVNF VNFM configuration file.</p> <p>AutoVNF processes this data to deploy the VNFCs. Refer to <a href="#">Activate the AutoVNF Configuration Files, on page 85</a> for more information.</p>
8	<p>On the AutoVNF VM, load and commit the VNF configuration file prepared in the previous step. Once committed, activate the loaded AutoVNF VNF configuration file.</p> <p>AutoVNF processes this data to deploy the VNFCs. Refer to <a href="#">Activate the AutoVNF Configuration Files, on page 85</a> for more information.</p>
9	<p>AutoVNF passes the VNF configuration to the pre-created VNFM VM instance.</p> <p>It ensures that the various VM catalogs pertaining to other VNFCs are on-boarded by the VNFM. It accomplishes this through a number of YANG-based definitions which are then used to configure various aspects of the virtualized environment using REST and NETCONF APIs.</p> <p>That VNFM mounts the VNFC catalogs and works with AutoVNF to deploy the various components that comprise the desired VNF use-case (e.g. UGP or USF).</p>
10, 12	<p>The VNFM leverages the VNFC information to deploy the UEM VMs cluster.</p> <p>Though the USP architecture represents a single VNF to other network elements, it is comprised of multiple VM types each having their own separate catalogs. The UEM component of the USP works with the VNFM to deploy these catalogs based on the intended VNF use case (e.g. UGP, USF, etc.).</p>

Callout	Description
11, 13	<p>The UEM processes the Day-0 configuration information it received from the VNFM and deploys the Control Function (CF) and Service Function (SF) VNFC VMs.</p> <p>Once all of the VNF components (VNFCs) have been successfully deployed, AutoVNF notifies AutoDeploy.</p> <p><b>Important</b> In multi-VNF deployments, AutoVNF waits until it receives confirmation that all of the VNFCs have been on-boarded successfully for the current VNF deployment before it initiates the deployment of the next VNF.</p>

## Pre-VNF Installation Verification

Prior to installing the USP, please ensure that the following is true:

- The prerequisite hardware is installed and operational with network connectivity.
- The prerequisite software is installed and configured and functioning properly:
  - You have administrative rights to the operating system.
  - VIM Orchestrator is properly installed and operational.
  - VIM components are properly installed and operational. This configuration includes networks, flavors, and sufficient quota allocations to the tenant.




---

**Note** Supported and/or required flavors and quota allocations are based on deployment models. Contact your Cisco representative for more information.

---

- You have administrative rights to the OpenStack setup.
- The VNFM software is in properly installed and operational.




---

**Note** Cisco's Elastic Services Controller (ESC) is the only VNFM supported in this release.

---

- The Cisco USP software ISO has been downloaded and is accessible by you.

## Deploy the USP-based VNF

The AutoVNF software roles within the Ultra Automation Services (UAS) is used to automate the USP-based VNF deployment. The automated deployment process through AutoVNF is described in [VNF Deployment Automation Overview, on page 73](#).

To deploy the USP-based VNF using AutoDeploy:

1. [Onboard the USP ISO, on page 78.](#)
2. [Extract the UAS Bundle, on page 79.](#)
3. [Extract the UEM VM Image, on page 80.](#)
4. [Extract the UGP VM Image, on page 81.](#)
5. [Upload the USP VM Images to Glance, on page 82.](#)
6. [Deploy the AutoVNF VM, on page 82.](#)
7. [Activate the AutoVNF Configuration Files, on page 85.](#)

## Onboard the USP ISO

The files required to deploy the USP components are distributed as RPMs (called “bundles”) in a single ISO package. They are maintained using YUM on the Onboarding Server. The following bundles are part of the ISO:

USP Bundle Name	Description
usp-em-bundle	The Element Manager (EM) Bundle RPM containing images and metadata for the Ultra Element Manager (UEM) module.
usp-uas-bundle	The Ultra Automation Services Bundle RPM containing AutoIT, AutoDeploy, AutoVNF, Ultra Web Services (UWS), and other automation packages.
usp-ugp-bundle	The Ultra Gateway Platform (UGP) Bundle RPM containing images for Ultra Packet core (VPC-DI). This bundle contains non-trusted images.
usp-vnfm-bundle	The VNFM Bundle RPM containing an image and a boot-up script for ESC (Elastic Service Controller).
usp-yang-bundle	The Yang Bundle RPM containing YANG data models including the VNFD and VNFR.
usp-auto-it-bundle	The bundle containing the AutoIT packages required to deploy the UAS.
ultram-manager	This package contains the script and relevant files needed to deploy the Ultra Health Service.



### Important

Release 6.4 will not be backward compatible with previous releases, i.e., you cannot deploy a 6.4 ISO from an AutoDeploy/AutoIT/AutoVNF running a pre-6.4 release, and vice-versa.

In addition to the bundles, the ISO bundle also includes scripts used to deploy the bundles including UAS.

Before proceeding with these instructions, ensure that the prerequisites identified in [USP Installation Prerequisites, on page 29](#) have been met.

To onboard the ISO package:

1. Log on to the Onboarding Server.
2. Download the USP ISO bundle and related files pertaining to the release.
3. Create a mount point on the Onboarding Server and mount the ISO package:

```
mkdir /var/usp-iso
```

4. Mount the USP ISO.

```
sudo mount -t iso9660 -o loop <ISO_download_directory>/<ISO_package_name>  
/var/usp-iso
```

**Example:** The following command mounts the ISO bundle called *usp-5\_5\_0-1255.iso* located in a directory called *5\_5\_0-1283* to */var/usp-iso*:

```
sudo mount -t iso9660 -o loop 5_5_0-1064/usp-5_5_0-1064.iso /var/usp-iso  
mount: /dev/loop1 is write-protected, mounting read-only
```

5. Verify the mount configuration.

```
df -h
```

**Example output:**

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda2	187G	178G	316M	100%	/
devtmpfs	63G	0	63G	0%	/dev
tmpfs	63G	4.0K	63G	1%	/dev/shm
tmpfs	63G	1.4M	63G	1%	/run
tmpfs	63G	0	63G	0%	/sys/fs/cgroup
/dev/sda1	477M	112M	336M	25%	/boot
tmpfs	13G	0	13G	0%	/run/user/0
/dev/loop1	4.2G	4.2G	0	100%	/var/usp-iso

6. Proceed to [Extract the UAS Bundle, on page 79](#).

## Extract the UAS Bundle

Once the USP ISO has been mounted, the UAS bundle must be extracted from the ISO in order to prepare the configuration files required for deployment.

These instructions assume you are already logged on to the Onboarding Server.

To extract the UAS bundle:

1. Navigate to the tools directory within the ISO mount.

```
cd /var/usp-iso/tools/
```

2. Launch the *usp-uas-installer.sh* script.

```
sudo ./usp-uas-installer.sh
```

The script extracts the files that comprise the UAS bundle to */opt/cisco/usp/uas-installer*.

3. Verify that files have been extracted.

Example output:

```
ll /opt/cisco/usp/uas-installer
```

```
total 12
drwxr-xr-x. 5 root root 4096 May 11 08:04 common
drwxr-xr-x. 2 root root 4096 May 11 08:04 images
drwxr-xr-x. 2 root root 4096 May 11 08:04 scripts

ll /opt/cisco/usp/uas-installer/images/

total 707580
-rw-r--r--. 1 root root 723898880 May 10 15:40 usp-uas-1.0.0-601.qcow2

ll /opt/cisco/usp/uas-installer/scripts/

total 56
-rwxr-xr-x. 1 root root 5460 May 11 08:04 autoit-user.py
-rwxr-xr-x. 1 root root 4762 May 11 08:04 encrypt_account.sh
-rwxr-xr-x. 1 root root 3945 May 11 08:04 encrypt_credentials.sh
-rwxr-xr-x. 1 root root 13846 May 11 08:04 uas-boot.py
-rwxr-xr-x. 1 root root 5383 May 11 08:04 uas-check.py
-rwxr-xr-x. 1 root root 10385 May 11 08:04 usp-tenant.py
```

4. Proceed to [Extract the UEM VM Image, on page 80](#).

## Extract the UEM VM Image

The image files required to deploy the UEM are distributed as part of an RPM bundle. The bundle is called “usp-em-bundle-*<version>*-1.x86\_64.rpm” and it is distributed as part of the USP ISO image.

The UEM image file is called “em-*<version>*.qcow2”. Prior to installing the UGP VNF, you must extract this file from the ISO.



### Important

These instructions assume that you have already mounted the USP ISO.

To extract the files:

1. Navigate to the directory containing the rpm bundles.

```
cd /var/usp-iso/repo
```

2. View the contents of, and information about the bundle.

```
rpm -qilp usp-em-bundle-<version>-1.x86_64.rpm
```

Example output:

```
<---SNIP--->
/opt/cisco/usp/bundles/em-bundle/em-5_7_0_1481.qcow
/opt/cisco/usp/bundles/em-bundle/usp-build-info.json
/opt/cisco/usp/bundles/em-bundle/usp-bundle-manifest.yml
<---SNIP--->
```

3. Extract the required artifacts from the bundle.

- a. Make a directory and ensure that it is empty.

```
mkdir -p /tmp/artifacts; rm -rf /tmp/artifacts/*
```

- b. Navigate to the directory just created.

```
cd /tmp/artifacts/
```

- c. Extract the UEM image file.

```
rpm2cpio /var/usp-iso/repo/usp-em-bundle-<version>-1.x86_64.rpm | cpio
-idmv ./opt/cisco/usp/bundles/em-bundle/em-<version>.qcow2
```

- d. Ensure that the image has been extracted.

```
ls -l ./opt/cisco/usp/bundles/em-bundle/em-<version>.qcow2
```

4. Proceed to [Extract the UGP VM Image, on page 81](#).

## Extract the UGP VM Image

The image files required to deploy the UGP are distributed as part of an RPM bundle. The bundle is called “usp-ugp-bundle-<version>.x86\_64.rpm” and it is distributed as part of the USP ISO image.

The UGP image file is called “qvpc-di-<version>.qcow2.tgz”. Prior to installing the UGP VNF, you must extract the image files from the ISO.



### Important

These instructions assume that you have already mounted the USP ISO.

To extract the files:

1. Navigate to the directory containing the rpm bundles.

```
cd /var/usp-iso/repo
```

2. View the contents of, and information about the bundle.

```
rpm -qilp usp-ugp-bundle-<version>-1.x86_64.rpm
```

Example output:

```
<---SNIP--->
/opt/cisco/usp/bundles/ugp-bundle/qvpc-di-21.4.v0.qcow2.tgz
/opt/cisco/usp/bundles/ugp-bundle/qvpc-di-21.4.v0.qcow2.tgz.md5
/opt/cisco/usp/bundles/ugp-bundle/qvpc-di-21.4.v0.qcow2.tgz.sha1
/opt/cisco/usp/bundles/ugp-bundle/qvpc-di-21.4.v0.qcow2.tgz.sha512
<---SNIP--->
```

3. Extract the required artifacts from the bundle.

- a. Make a directory and ensure that it is empty.

```
mkdir -p /tmp/artifacts; rm -rf /tmp/artifacts/*
```

- b. Navigate to the directory just created.

```
cd /tmp/artifacts/
```

- c. Extract the UGP image files.

```
rpm2cpio /var/usp-iso/repo/usp-ugp-bundle-<version>-1.x86_64.rpm |
cpio -idmv
./opt/cisco/usp/bundles/ugp-bundle/qvpc-di-<version>.qcow2.tgz
```

- d. Ensure that the images have been extracted.

```
ls -l ./opt/cisco/usp/bundles/ugp-bundle/qvpc-di-<version>.qcow2.tgz
```

4. Extract the CF qcow2 image.

```
tar -zxvf qvpc-di-<version>.qcow2.tgz qvpc-di-cf-<version>.qcow2
```

5. Extract the SF qcow2 image.

```
tar -zxvf qvpc-di-<version>.qcow2.tgz qvpc-di-sf-<version>.qcow2
```

6. Proceed to [Upload the USP VM Images to Glance, on page 82](#).

## Upload the USP VM Images to Glance

The UAS, UEM, and UGP VM images extracted from the USP ISO must be uploaded into OpenStack Glance.

To upload the images to Glance:

1. Login to OSP-D and make sure to “su - stack” and “source stackrc”. Determine the name of the heat *stack\_name*.

```
heat stack-list
```

2. Source the rc file for the stack.

```
source ~/ <stack_name> rc
```

3. Upload the UAS image.

```
glance image-create --file usp-uas- <version> .qcow2 --container-format bare --disk-format qcow2 --name ultra-autovnf- <version>
```

4. Upload the UEM image.

```
glance image-create --file em- <version> .qcow2 --container-format bare --disk-format qcow2 --name ultra-em- <version>
```

5. Upload the CF image.

```
glance image-create --file qvpc-di-cf- <version> .qcow2 --container-format bare --disk-format qcow2 --name ultra-cf- <version>
```

6. Upload the SF image.

```
glance image-create --file qvpc-di-xf- <version> .qcow2 --container-format bare --disk-format qcow2 --name ultra-sf- <version>
```

7. Proceed to [Deploy the AutoVNF VM, on page 82](#).

## Deploy the AutoVNF VM

The VM for AutoVNF is deployed using *boot\_uas.py* script provided with the UAS bundle. The script is located in the following directory:

```
/opt/cisco/usp/bundles/uas-bundle/tools
```

This script includes a number of deployment parameters for the VM. These parameters are described in the help information pertaining to the script which can be accessed by executing the following command:

```
./boot_uas.py -h
```

The help information is provided as an appendix in this document. Refer to [boot\\_uas.py Help, on page 179](#).




---

**Important**

These instructions assume you are already logged on to the Onboarding Server.

---

To deploy the AutoVNF VM:

1. Navigate to the directory containing the `boot_uas.py` file.

```
cd /opt/cisco/usp/bundles/uas-bundle/tools
```

2. Deploy the AutoVNF VM.

```
./boot_uas.py --autovnf --openstack --image <image_name> --flavor
<flavor_name> --net <network_name>
```

There are additional arguments that can be executed with this script based on your deployment scenario. Refer to [boot\\_uas.py Help, on page 179](#) for details.




---

**Important**

Both version 2 and 3 of OpenStack Keystone APIs are supported. You can specify the desired version using the `--os_identity_api_version` argument with this script. For example to specify the use of version 3, add the argument `--os_identity_api_version 3`. The default is version 2.

---

Upon executing the script, you are prompted to enter user credentials for performing operations within the AutoVNF VM.

3. Provide the requested information.

- **AutoVNF VM Login Password:** The password for the default user account, which is named `ubuntu`.
- **AutoVNF API Access password for "admin":** The password for the ConfD administrator user, which is named `admin`.
- **AutoVNF API Access password for "oper":** The password for the ConfD operator user, which is named `oper`.
- **AutoVNF API Access password for "security":** The password for the ConfD security administrator user, which is named `security-admin`.




---

**Important**

Ensure that all passwords meet the requirements specified in [Password Requirements and Login Security, on page 32](#).

---

4. Log on to the AutoVNF VM as `ubuntu`. Use the password that was created earlier for this user.
5. Become the root user.

```
sudo -i
```

6. Prepare the `system.cfg` file. This will serve as the Day-0 config for the VNF. Refer to [Sample system.cfg File, on page 211](#) for an example configuration file.

**Important**

Though administrative user credentials can be specified in clear text in the `system.cfg` file, it is not recommended. For security purposes, it is recommended that you configure a secure token for the user account in the VNF configuration file and reference that file as part of the VDU catalog pertaining to the CF using the **login-credential** parameter. In the `system.cfg` file, use the `$CF_LOGIN_USER` and `$CF_LOGIN_PASSWORD` variables as follows to call the values configured for the secure token:

```
configure
context local
  administrator $CF_LOGIN_USER password $CF_LOGIN_PASSWORD ftp
```

7. Upload the `system.cfg` to the `/opt/cisco/usp/uploads/` directory on the AutoVNF VM.
8. Copy the ESC scripts from the `/opt/cisco/usp/uas/autovnf/vnfms/esc-scripts` directory on the AutoVNF VM to the VNFM (ESC) VMs.

These are custom scripts which aid in the VNF instantiation.

- a. Connect to the master VNFM (ESC) VM and copy the scripts.

```
cd /opt/cisco/usp/uas/autovnf/vnfms/esc-scripts
scp esc-vpc-di-internal-keys.sh <esc_user>@<master_esc_vm_address>:
opt/cisco/esc/esc-scripts/esc-vpc-di-internal-keys.sh
scp esc_vpc_chassis_id.py <esc_user>@<master_esc_vm_address>:
opt/cisco/esc/esc-scripts/esc_vpc_chassis_id.py
scp esc_volume_em_staging.sh <esc_user>@<master_esc_vm_address>:
/opt/cisco/esc/esc-scripts/esc_volume_em_staging.sh
```

- b. Connect to the standby VNFM (ESC) VM and copy the scripts.

```
scp esc-vpc-di-internal-keys.sh <esc_user>@<standby_esc_vm_address>:
opt/cisco/esc/esc-scripts/esc-vpc-di-internal-keys.sh
scp esc_vpc_chassis_id.py <esc_user>@<standby_esc_vm_address>:
opt/cisco/esc/esc-scripts/esc_vpc_chassis_id.py
scp esc_volume_em_staging.sh <esc_user>@<standby_esc_vm_address>:
/opt/cisco/esc/esc-scripts/esc_volume_em_staging.sh
```

9. Confirm that the VNFM has been configured with the VIM connectors for the VNF tenants. A connector is needed for each tenant. Refer to [Sample ESC VIM Connector Configuration, on page 213](#) for an example configuration.

- a. Connect to the master VNFM (ESC) VM.

- b. Log on to the ConfD command line.

```
/opt/cisco/esc/confd/bin/confd_cli -C
```

- c. Confirm the VIM connector configuration.

```
show running-config esc_system_config vim_connectors vim_connector
  <vim_connector_name>
```

If the connectors have not been configured, refer to the documentation for the appropriate version of ESC software. ESC product documentation is available here: <https://www.cisco.com/c/en/us/support/cloud-systems-management/elastic-services-controller-esc/tsd-products-support-series-home.html>



---

**Important** The OpenStack Keystone configuration version specified for the authentication URL in the connector must match the version used when deploying AutoVNF and the version specified in the AutoVNF configuration file.

---

- d. Repeat step [9.c, on page 84](#) for each VIM connector.



---

**Important** If the ESC VMs are upgraded or redeployed at any time, ensure that you reload the VIM connectors on the new or upgraded ESC VM deployment.

---

10. Prepare the AutoVNF VNFM configuration file.

This file provides the information necessary to allow AutoVNF to communicate with the VNFM (ESC).

A sample configuration file is provided for reference in [Sample AutoVNF VNFM Configuration File, on page 215](#).



---

**Important** The OpenStack Keystone configuration version specified in the VNFM configuration file used by AutoVNF must match the version used when deploying AutoVNF and the version specified in the ESC VIM connector(s). Set the **api-version** parameter to the appropriate version type.

---



---

**Important** If the ESC VMs are upgraded or redeployed at any time after the AutoVNF is deployed, you may need to change the ESC endpoint details in the AutoVNF VNFM configuration file and reload it.

---

11. Save the AutoVNF VNFM configuration file to your home directory on the AutoVNF VM.
12. Prepare the AutoVNF VNF configuration file.

This file provides the VNF configuration information used by AutoVNF during the deployment process. A sample configuration file is provided for reference in [Sample AutoVNF VNF Configuration File, on page 217](#).



---

**Caution** Ensure that the network service descriptor (NSD) identified in the AutoVNF VNF configuration file is identical to the one specified in the AutoVNF VNFM configuration file.

---

13. Save the AutoVNF VNF configuration file to your home directory on the AutoVNF VM.
14. Proceed to [Activate the AutoVNF Configuration Files, on page 85](#).

## Activate the AutoVNF Configuration Files

Once you have completed preparing your AutoVNF VNFM and VNF configuration files, you must load the configuration and activate the deployment.




---

**Important** User credentials are configured through Secure Tokens specified in the configuration file. Ensure that passwords configured with Secure Token meet the requirements specified in [Password Requirements and Login Security, on page 32](#).

---

Once activated, AutoVNF proceeds with the deployment automation workflow as described in [VNF Deployment Automation Overview, on page 73](#).

---




---

**Important** These instructions assume you are already logged on to the AutoVNF VM as the *root* user and that your configuration files have been prepared for your deployment as per the information and instructions in [Deploy the AutoVNF VM, on page 82](#). These instructions also assume that AutoVNF has access to the VNFC image files (either locally or on a remote server) provided with the USP ISO.

---

To activate the USP deployment using AutoVNF:

1. Login to the ConfD CLI as the admin user.

```
confd_cli -u admin -C
```

2. Enter the *admin* user password when prompted.
3. Enter the ConfD configuration mode.

```
config
```

4. Load the AutoVNF VNFM configuration file to load the VNFM information into the AutoVNF database.

```
load merge <your_vnfm_file_name> .cfg
commit
end
```

---




---

**Important** If you are performing this process as a result of an upgrade or redeployment, you must use the load replace variant of this command:

```
load replace <your_vnfm_file_name> .cfg
commit
end
```

---

5. Activate the AutoVNF VNFM configuration file.

```
activate nsd <nsd_name>
```

---




---

**Important** The output of this command is a transaction-id which can be used to monitor the deployment progress. If need be, the VIM deployment can be deactivated using the **deactivate** variant of this command.

---

6. Load the AutoVNF VNF configuration file to load the deployment name and its attributes in the AutoVNF database.
-

```
load merge <your_vnf_file_name> .cfg
commit
end
```



**Important** If you are performing this process as a result of an upgrade or redeployment, you must use the load replace variant of this command:

```
load replace <your_vnf_file_name> .cfg
commit
end
```

7. Activate the AutoVNF VNF configuration file.

```
activate nsd <nsd_name>
```



**Important** The output of this command is a transaction-id which can be used to monitor the deployment progress. If need be, the VIM deployment can be deactivated using the **deactivate** variant of this command.

8. Monitor the progress of the deployment by viewing transaction logs:

```
show log <transaction_id> | display xml
```

*transaction\_id* is the ID displayed as a result of the **activate-deployment** command.

The logs display status messages for each node in each VNF that the configuration file defines. Example success messages for the different components deployed through AutoVNF are shown below:

- VNF:

```
Fri May 12 21:44:35 UTC 2017 [Task: 1494624612779/tblvnfd2] Successfully completed
all Vnf Deployments.
```

- Entire Deployment:

```
Fri May 12 21:57:38 UTC 2017 [Task: 1494624612779] Success
```



**Important** If there are any issues seen when executing the above commands, refer to [Monitoring and Troubleshooting the Deployment, on page 100](#).

## Upgrading/Redeploying the Stand-alone AutoVNF VM Instance

Use the following procedure to upgrade or redeploy the AutoVNF software image in scenarios where AutoVNF was brought up as stand-alone instance.



**Important** These instructions assume you are already logged on to the Onboarding Server.

1. Delete the AutoVNF VM instance.

```
./boot_uas.py --openstack --autovnf --delete <transaction_id>
```

2. *Optional.* If required remove the OpenStack artifacts which were created manually to bring up AutoVNF.
3. Follow the procedures in [Deploy the USP-based VNF, on page 77](#) to redeploy AutoVNF with the new software version.



## CHAPTER 5

# Upgrading the Ultra Services Platform Deployment

---

- [Upgrading/Redeploying USP for Hyper-Converged Ultra M Models](#), on page 89
- [Upgrading/Redeploying VNFs Deployed Through a Stand-alone AutoVNF Instance](#), on page 92

## Upgrading/Redeploying USP for Hyper-Converged Ultra M Models

The specific process to upgrade or redeploy your Ultra M deployment depends on the USP version you're upgrading from. Information and guidelines for the different upgrade scenarios are provided in the following sections:

- [Upgrading from 5.1.x to This Release](#), on page 89
- [Upgrading from 5.5.x to This Release](#), on page 89
- [Upgrading from 5.7.x to This Release](#), on page 90
- [Upgrading from 5.8.x to This Release](#), on page 91
- [Upgrading from 6.0.x OR 6.1.x to This Release](#), on page 91



---

**Important**

If you're upgrading to 6.2 or later releases, follow the upgrade instructions in the [Patch Upgrade Procedure](#), on page 235 appendix of this guide.

---

### Upgrading from 5.1.x to This Release

There are customer-specific considerations for this upgrade scenario. Please contact your local Sales or Support representative for details.

### Upgrading from 5.5.x to This Release

Use the information and guidelines below to upgrade from a 5.5.x release to this release.

- Deactivate your current deployment using the information and instructions in: [https://www.cisco.com/c/dam/en/us/td/docs/wireless/asr\\_5000/21-3\\_N5-5/USP\\_DAG/N5-5-USP-Deploy-Automation-Guide.pdf](https://www.cisco.com/c/dam/en/us/td/docs/wireless/asr_5000/21-3_N5-5/USP_DAG/N5-5-USP-Deploy-Automation-Guide.pdf)
  - Deactivate the VNF. Refer to the section on *Deactivating the USP Deployment* for details.




---

**Caution** Ensure that you perform the steps recommended in the Caution statement at the beginning of this section prior to performing the deactivation.

---

- Deactivate the VIM and VIM Orchestrator. Refer to the section on *Deactivating the USP Deployment* for details.
- Terminate the AutoDeploy VM. Refer to the section on *Terminating the AutoDeploy VM* for details.
- Terminate the AutoIT-VNF VM. Refer to the section on *Terminating the AutoIT-VNF VM* for details.
- Deploy this release using the information and instructions in this document. Refer to [Deploying Hyper-Converged Ultra M Models Using UAS, on page 39](#) and follow the instructions for deploying both the VIM and the VNF for Hyper-Converged Ultra M Models.

### Upgrading from 5.7.x to This Release

Use the information and guidelines below to upgrade from a 5.7.x release to this release.

- Deactivate your current deployment using the information and instructions in: [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-4\\_N5-7/USP\\_DAG/N5-7\\_USP\\_Deploy\\_Automation\\_Guide.html](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-4_N5-7/USP_DAG/N5-7_USP_Deploy_Automation_Guide.html)
  - Deactivate the VNF. Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-4\\_N5-7/USP\\_DAG/N5-7\\_USP\\_Deploy\\_Automation\\_Guide/N5-5\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_010010.html#reference\\_bpn\\_hyv\\_kbb](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-4_N5-7/USP_DAG/N5-7_USP_Deploy_Automation_Guide/N5-5_USP_Deploy_Automation_Guide_chapter_010010.html#reference_bpn_hyv_kbb) for details.




---

**Caution** Ensure that you perform the steps recommended in the Caution statement at the beginning of this section prior to performing the deactivation.

---

- Deactivate the VIM and VIM Orchestrator. Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-4\\_N5-7/USP\\_DAG/N5-7\\_USP\\_Deploy\\_Automation\\_Guide/N5-5\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_010010.html#reference\\_bpn\\_hyv\\_kbb](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-4_N5-7/USP_DAG/N5-7_USP_Deploy_Automation_Guide/N5-5_USP_Deploy_Automation_Guide_chapter_010010.html#reference_bpn_hyv_kbb) for details.
- Terminate the AutoDeploy VM. Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-4\\_N5-7/USP\\_DAG/N5-7\\_USP\\_Deploy\\_Automation\\_Guide/N5-5\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_010010.html#reference\\_acl\\_5yv\\_kbb](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-4_N5-7/USP_DAG/N5-7_USP_Deploy_Automation_Guide/N5-5_USP_Deploy_Automation_Guide_chapter_010010.html#reference_acl_5yv_kbb) for details.
- Terminate the AutoIT-VNF VM. Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-4\\_N5-7/USP\\_DAG/N5-7\\_USP\\_Deploy\\_Automation\\_Guide/N5-5\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_010010.html#reference\\_jsk\\_xyv\\_kbb](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-4_N5-7/USP_DAG/N5-7_USP_Deploy_Automation_Guide/N5-5_USP_Deploy_Automation_Guide_chapter_010010.html#reference_jsk_xyv_kbb) for details.
- Deploy this release using the information and instructions in this document. Refer to [Deploying Hyper-Converged Ultra M Models Using UAS, on page 39](#) and follow the instructions for deploying both the VIM and the VNF for Hyper-Converged Ultra M Models.

## Upgrading from 5.8.x to This Release

Use the information and guidelines below to upgrade from a 5.8.x release to this release.

- Deactivate your current deployment using the information and instructions in: [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-5\\_N5-8/USP\\_DAG/N5-8\\_USP\\_Deploy\\_Automation\\_Guide.html](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-5_N5-8/USP_DAG/N5-8_USP_Deploy_Automation_Guide.html)
- Deactivate the VNF. Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-5\\_N5-8/USP\\_DAG/N5-8\\_USP\\_Deploy\\_Automation\\_Guide/N5-8\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_0111.html#reference\\_bpn\\_hyv\\_kbb](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-5_N5-8/USP_DAG/N5-8_USP_Deploy_Automation_Guide/N5-8_USP_Deploy_Automation_Guide_chapter_0111.html#reference_bpn_hyv_kbb) for details.



---

**Caution** Ensure that you perform the steps recommended in the Caution statement at the beginning of this section prior to performing the deactivation.

---

- Deactivate the VIM and VIM Orchestrator. Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-5\\_N5-8/USP\\_DAG/N5-8\\_USP\\_Deploy\\_Automation\\_Guide/N5-8\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_0111.html#reference\\_bpn\\_hyv\\_kbb](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-5_N5-8/USP_DAG/N5-8_USP_Deploy_Automation_Guide/N5-8_USP_Deploy_Automation_Guide_chapter_0111.html#reference_bpn_hyv_kbb) for details.
  - Terminate the AutoDeploy VM. Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-5\\_N5-8/USP\\_DAG/N5-8\\_USP\\_Deploy\\_Automation\\_Guide/N5-8\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_0111.html#reference\\_ac1\\_5yv\\_kbb](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-5_N5-8/USP_DAG/N5-8_USP_Deploy_Automation_Guide/N5-8_USP_Deploy_Automation_Guide_chapter_0111.html#reference_ac1_5yv_kbb) for details.
  - Terminate the AutoIT-VNF VM. Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-5\\_N5-8/USP\\_DAG/N5-8\\_USP\\_Deploy\\_Automation\\_Guide/N5-8\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_0111.html#reference\\_jsk\\_xyv\\_kbb](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-5_N5-8/USP_DAG/N5-8_USP_Deploy_Automation_Guide/N5-8_USP_Deploy_Automation_Guide_chapter_0111.html#reference_jsk_xyv_kbb) for details.
- Deploy this release using the information and instructions in this document. Refer to [Deploying Hyper-Converged Ultra M Models Using UAS, on page 39](#) and follow the instructions for deploying both the VIM and the VNF for Hyper-Converged Ultra M Models.

## Upgrading from 6.0.x OR 6.1.x to This Release

Use the information and guidelines below to upgrade from a 6.0.x or 6.1.x release to this release.

- Deactivate your current deployment using the information and instructions in: [Deactivating the USP Deployment, on page 95](#)
- Deactivate the VNF. Refer to [Deactivating the USP Deployment, on page 95](#) for details.



---

**Caution** Ensure that you perform the steps recommended in the Caution statement at the beginning of this section prior to performing the deactivation.

---

- Deactivate the VIM and VIM Orchestrator. Refer to [Deactivating the USP Deployment, on page 95](#) for details.
- Terminate the AutoDeploy VM. Refer to [Terminating the AutoDeploy VM, on page 96](#) for details.
- Terminate the AutoIT VM. Refer to [Terminating the AutoIT VM, on page 96](#) for details.

- Deploy this release using the information and instructions in this document. Refer to [Deploying Hyper-Converged Ultra M Models Using UAS, on page 39](#) and follow the instructions for deploying both the VIM and the VNF for Hyper-Converged Ultra M Models.

## Upgrading/Redeploying VNFs Deployed Through a Stand-alone AutoVNF Instance

The specific process to upgrade or redeploy your stand-alone AutoVNF deployment depends on the USP version you're upgrading from. Information and guidelines for the different upgrade scenarios are provided in the following sections:

- [Upgrading from 5.7.x to This Release, on page 92](#)
- [Upgrading from 5.8.x to This Release, on page 93](#)
- [Upgrading from 6.0.x OR 6.1.x to This Release, on page 93](#)



### Important

If you're upgrading to 6.2 or later releases, follow the upgrade instructions in the [Patch Upgrade Procedure, on page 235](#) appendix of this guide.

### Upgrading from 5.7.x to This Release

Use the information and guidelines below to upgrade from a 5.7.x release to this release.

- Deactivate your current deployment using the information and instructions in: [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-4\\_N5-7/USP\\_DAG/N5-7\\_USP\\_Deploy\\_Automation\\_Guide.html](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-4_N5-7/USP_DAG/N5-7_USP_Deploy_Automation_Guide.html)
- Deactivate the VNF. Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-4\\_N5-7/USP\\_DAG/N5-7\\_USP\\_Deploy\\_Automation\\_Guide/N5-5\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_010010.html#reference\\_bpn\\_hyv\\_kbb](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-4_N5-7/USP_DAG/N5-7_USP_Deploy_Automation_Guide/N5-5_USP_Deploy_Automation_Guide_chapter_010010.html#reference_bpn_hyv_kbb) for details.



### Caution

Ensure that you perform the steps recommended in the Caution statement at the beginning of this section prior to performing the deactivation.

- Terminate the AutoVNF VMs by executing the following command on the primary AutoVNF VM:

```
./boot_autovnf.py --delete
```

Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-4\\_N5-7/USP\\_DAG/N5-7\\_USP\\_Deploy\\_Automation\\_Guide/N5-5\\_USP\\_Deploy\\_Automation\\_Guide\\_appendix\\_011001.html](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-4_N5-7/USP_DAG/N5-7_USP_Deploy_Automation_Guide/N5-5_USP_Deploy_Automation_Guide_appendix_011001.html) for details.

- Deploy this release using the information and instructions in this document. Refer to [Deploying Hyper-Converged Ultra M Models Using UAS, on page 39](#) and follow the instructions for deploying both the VIM and the VNF for Hyper-Converged Ultra M Models.

## Upgrading from 5.8.x to This Release

Use the information and guidelines below to upgrade from a 5.8.x release to this release.

- Deactivate your current deployment using the information and instructions in: [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-5\\_N5-8/USP\\_DAG/N5-8\\_USP\\_Deploy\\_Automation\\_Guide.html](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-5_N5-8/USP_DAG/N5-8_USP_Deploy_Automation_Guide.html)
- Deactivate the VNF. Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-5\\_N5-8/USP\\_DAG/N5-8\\_USP\\_Deploy\\_Automation\\_Guide/N5-8\\_USP\\_Deploy\\_Automation\\_Guide\\_chapter\\_0111.html#reference\\_bpn\\_hyv\\_kbb](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-5_N5-8/USP_DAG/N5-8_USP_Deploy_Automation_Guide/N5-8_USP_Deploy_Automation_Guide_chapter_0111.html#reference_bpn_hyv_kbb) for details.



---

**Caution** Ensure that you perform the steps recommended in the Caution statement at the beginning of this section prior to performing the deactivation.

---

- Terminate the AutoVNF VMs by executing the following command on the primary AutoVNF VM:  

```
./boot_autovnf.py --delete
```

Refer to [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-5\\_N5-8/USP\\_DAG/N5-8\\_USP\\_Deploy\\_Automation\\_Guide/N5-8\\_USP\\_Deploy\\_Automation\\_Guide\\_appendix\\_010100.html](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-5_N5-8/USP_DAG/N5-8_USP_Deploy_Automation_Guide/N5-8_USP_Deploy_Automation_Guide_appendix_010100.html) for details.
- Deploy this release using the information and instructions in this document. Refer to [Deploying Hyper-Converged Ultra M Models Using UAS, on page 39](#) and follow the instructions for deploying both the VIM and the VNF for Hyper-Converged Ultra M Models.

## Upgrading from 6.0.x OR 6.1.x to This Release

Use the information and guidelines below to upgrade from a 6.0.x or 6.1.x release to this release.

- Deactivate your current deployment using the information and instructions in: [Deactivating the USP Deployment, on page 95](#)
- Deactivate the VNF. Refer to [Deactivating the USP Deployment, on page 95](#) for details.



---

**Caution** Ensure that you perform the steps recommended in the Caution statement at the beginning of this section prior to performing the deactivation.

---

- Terminate the AutoVNF VMs by executing the following command on the primary AutoVNF VM:  

```
./boot_uas.py --autovnf --openstack --delete
```

Refer to [boot\\_uas.py Help, on page 179](#) for details.
- Deploy this release using the information and instructions in this document. Refer to [Deploying VNFs Using AutoVNF, on page 73](#) and follow the instructions for deploying both the VIM and the VNF for Hyper-Converged Ultra M Models.





## CHAPTER 6

# Post Deployment Operations

---

- [Deactivating the USP Deployment, on page 95](#)
- [Terminating the AutoDeploy VM, on page 96](#)
- [Terminating the AutoIT VM, on page 96](#)
- [Deploy and Undeploy the Card with the NCS CLI, on page 97](#)
- [Monitoring and Troubleshooting the Deployment, on page 100](#)
- [Monitoring AutoDeploy Operations, on page 127](#)
- [Monitoring AutoIT Operations, on page 132](#)
- [Monitoring AutoVNF Operations, on page 137](#)
- [UAS Log Collection, on page 153](#)
- [Monitoring VNFM Operations, on page 160](#)
- [Monitoring VNF Operations, on page 164](#)
- [Monitoring and Recovering AutoVNF Through AutoIT, on page 167](#)
- [Monitoring and Recovering VNFC Through AutoVNF, on page 168](#)
- [Troubleshooting Deactivation Process and Issues, on page 171](#)
- [Troubleshooting UEM Issues, on page 175](#)

## Deactivating the USP Deployment



### Caution

It is recommended that you perform the checks identified in [Pre-Deactivation/Post-Activation Health Check Summary, on page 100](#) before performing any deactivations. It is also recommended that you back up relevant data before proceeding. Refer to [Backing Up Deployment Information, on page 225](#) for more information.

Execute the following command to deactivate the entire USP deployment:

```
deactivate nsd <nsd_name>
```

The output of this command is a transaction-id which can be used to monitor the deactivation progress using the following command

```
show log <transaction_id> | display xml
```

Example output for a successful USP deactivation:

## Terminating the AutoDeploy VM

Terminating the AutoDeploy VM leverages the same *boot\_uas.py* script used to instantiate the AutoDeploy VM.



### Important

- Ensure that no changes have been made to this file since it was used to deploy AutoDeploy.
- Be sure to take a backup of the VM content if you are terminating the VM in order to upgrade with a new ISO.
- If AutoDeploy was deployed with HA support, this process terminates both VMs.

To terminate the AutoDeploy VM:

1. Log on to the Ultra M Manager Node.
2. Terminate the AutoDeploy VM.

```
./boot_uas.py --kvm --autodeploy --ha --delete-uas
```

Example command output:

```
2018-01-24 16:30:23,821 - Removing old deployment 'AutoDeploy_instance_0', if it exists
2018-01-24 16:30:24,176 - Removing old deployment 'AutoDeploy_instance_1', if it exists
```

3. View the status.

```
show uas
```

Example command output:

```
Id      Name                               State
-----
```

## Terminating the AutoIT VM

Terminating the AutoIT VM leverages the same *boot\_uas.py* script used to instantiate the AutoIT-VNF VM.



### Important

- Ensure that no changes have been made to this file since it was used to deploy AutoIT.
- Be sure to take a backup of the VM content if you are terminating the VM in order to upgrade with a new ISO.
- If AutoIT was deployed with HA support, this process terminates both VMs.

To terminate the AutoIT VM:

1. Log on to the Ultra M Manager Node.
2. Terminate the AutoIT VM.

```
./boot_uas.py --kvm --autoit --ha --delete-uas
```

Example command output:

```
2018-01-24 16:25:23,734 - Removing old deployment 'AutoIT_instance_0', if it exists
2018-01-24 16:25:24,056 - Removing old deployment 'AutoIT_instance_1', if it exists
```

3. View the status.

```
show uas
```

Example command output:

```
Id      Name                               State
-----
```

## Deploy and Undeploy the Card with the NCS CLI

To undeploy and redeploy the card (service or session function) using the NCS CLI:

1. Log on to the master UEM VM.
2. Access the NCS CLI.

```
sudo -i
ncs_cli -u admin -C
```

3. Undeploy or suspend the card.

```
suspend-vnfci vnfid <name> vdu <VDU> vnfci <VNFCI Instance>
```

For example:

```
suspend-vnfci vnfid abc vdu sf vnfci sf1 success true
```

4. Verify the operational status of VNF, card, VDUs. Suspending card removes the card, e.g. from CF.

In 6.2 and earlier releases:

```
show vnf-state
```

```
vnf-state running
```

In 6.3 and later releases:

```
show vnfproxy:vnfd <vnfd_name>vnf-state
```

```
vnf-state running
```

```
show card table
```

Slot	Card Type	Oper State	SPOF	Attach
1: CFC	Control Function Virtual Card	Active	No	
2: CFC	Control Function Virtual Card	Standby	-	
4: FC	1-Port Service Function Virtual Card	Standby	-	

In 6.2 and earlier releases:

```
show vdus
```

ID	CARD	TYPE	ID	DEVICE	MEMORY		CONSTITUENT		IS	INITIALIZED	VIM
					CPU	UTILS	STORAGE	ELEMENT			
ID				NAME	GROUP	BYTES	GROUP	GROUP	INFRA		
cf	control-function		cf1	scm-cf-nc	scm-cf-nc	ugp			true	true	
76e2f28a-4427-4b1d-9c44-72ff51e0d124						-	-	-			
			cf2	scm-cf-nc	scm-cf-nc	ugp			true	true	
b1155c6e-26f1-44c1-8832-0e9a02f7acd3						-	-	-			
sf	session-function		sf1	-	-	ugp			true	false	
7822cea9-1707-4790-abb3-33bb4d26b567						-	-	-			
			sf2	-	-	ugp			true	false	
7fd8f37f-59cf-4c9a-811f-faa0abd30b58						-	-	-			

In 6.3 and later releases:

```

show vnfmpoxy:vnfd <vnfd_name> vdus
vdus vdu cf1
card-type control-function
vnfci cf1
  device-name          vnf-d-1
  device-group         cf-nc
  constituent-element-group ugp-standalone
  is-infra             true
  initialized          true
  vim-id               1ca3fca7-8929-4830-9e35-4cac294b62bf
vnfci cf2
  device-name          vnf-d-2
  device-group         cf-nc
  constituent-element-group ugp-standalone
  is-infra             true
  initialized          true
  vim-id               6ce2a22c-1c03-4f5e-95a4-c791d09a1024
vdus vdu sf1
card-type session-function
vnfci sf1
  constituent-element-group ugp-standalone
  is-infra             true
  initialized          false
  vim-id               de5bec01-c3e4-4bbf-8f45-ac2354fc9fbc
vnfci sf2
  constituent-element-group ugp-standalone
  is-infra             true
  initialized          false
  vim-id               8fe9eebc-614a-464a-8e87-5288be0528c9

```

UEM changes the status of suspended card to *undeployed*. For example, UEM Zookeeper:

In 6.2 and earlier releases:

```

[zk: localhost:2181 (CONNECTED) 0] get /config/vnf
{"state":"run","name":"abcabc-autovnf-vpc-abcabc"}

[zk: localhost:2181 (CONNECTED) 1] get /config/vnfd
{"name":"abcabc-autovnf-vpc-abcabc","version":"6.0","deployment-flavor-id":["generic"],
"anti-affinity-cards":["control-function","session-function"],"card-type-to-vm":{"control-function":["cf"],"session-function":["sf"]}}

[zk: localhost:2181 (CONNECTED) 2] get /config/vdus/sf/sf1
{"cpts":[{"vnfc":"sf-vnfc-ugp","cpid":null,"vl":null}],"affinity":null,
"initvars":[{"best_path":"staros_param.cfg","path vars":[{"name":"CARD_TYPE_NM","val":"0x42020100"}, {"name":"SLOT_CARD_NUMBER","val":"3"}, {"name":"MEM_PROXY_ADDRS","val":"101.101.14.9,101.101.14.16,101.101.14.13"}]}],"operation":"create","ceg-id":"ugp","vnfci-id":"sf1",
"context-vars":null,"nat-pool":null,"vim-id":"abcabc-autovnf-vpc-abcabc-sf-1","volume":null}

```



# Monitoring and Troubleshooting the Deployment

## Pre-Deactivation/Post-Activation Health Check Summary

Table 17: Pre-deactivation/Post-activation Health Checks, on page 100 contains a summary of items to check/verify before performing a deactivation and/or after an activation.

Table 17: Pre-deactivation/Post-activation Health Checks

Item to Check	Notes
<a href="#">Checking OSP-D Server Health</a>	Perform all identified checks.
<a href="#">Checking Controller Server Health</a>	Perform all identified checks.
<a href="#">Checking OSD Compute Server Health</a>	Perform all identified checks.
<a href="#">Viewing AutoVNF Operational Data</a>	In particular, check the outputs of the following commands: <ul style="list-style-type: none"> <li>• <b>show uas</b></li> <li>• In releases prior to 6.0: <b>show autovnf-oper:vip-port</b> In 6.0 and later releases: <b>show vnfr</b></li> <li>• In releases prior to 6.0: <b>show autovnf-oper:vnf-em</b> In 6.0 and later releases: <b>show vnfr</b></li> <li>• In releases prior to 6.0: <b>show autovnf-oper:vnfm</b> In 6.0 and later releases: <b>show vnfr</b></li> </ul>
<a href="#">Viewing ESC Status</a>	Perform all identified checks.
<a href="#">Viewing ESC Health</a>	Perform all identified checks.
<a href="#">Viewing UEM Service Status</a>	Perform all identified checks.
<a href="#">Viewing VNF Information through the Control Function</a>	Perform all identified checks.

## Checking OSP-D Server Health

### Viewing Stack Status

Log on to the server on which OSP-D is running to view the stack status by executing the following command:

```
openstack stack list
```

Example output:

ID	Stack Name	Stack Status	Creation Time
db229d67-212d-4086-a266-e635b2902708	tb3-ultram	CREATE_COMPLETE	2017-06-20T02:31:31Z



**Note** Prior to an update, the stack status may be “CREATE\_COMPLETE” at the beginning of the update procedure. The stack status should read “UPDATE\_COMPLETE” and list and update time at the successful completion of the update procedure.

## Viewing the Bare Metal Node List

Log on to the server on which OSP-D is running to view the node list by executing the following command:

```
openstack baremetal node list
```

**Example command output:**

UUID	Name	Instance UUID	Power State	Provisioning State	Maintenance
6725bb18-2895-4a8a-86ad-96b00cc9df4d	None	bc903f51-8483-4522-bcd7-ac396ac626b1	power on	active	False
flaa6356-40a0-41de-belb-fa6033c9affb	None	05fbfb44-ccd9-475d-b263-58b2deaf8554	power on	active	False
f02357a3-6f9b-46ae-b31f-1a21f6d33543	None	dd0596b1-bd35-451a-85bc-c635e7fa6d14	power on	active	False
ca1153d6-ffaf-481a-ac9b-bc2afc450152	None	96d2725c-9c70-4a66-9d3c-4a0356faf1c0	power on	active	False
8f338102-c114-4a7a-94f0-9e1a54494519	None	85a9a708-5eae-4ea2-8b29-dc2acd6e515d	power on	active	False
5d3d3525-2528-4801-b885-6c4b340393a6	None	315c7aea-acef-4341-aa9e-bcd594cae592	power on	active	False
ac21208b-36fd-4404-8e68-53a90df3a29f	None	9f0b2ff3-5234-42e9-81dd-c0ef5e454137	power on	active	False
a6d92bfc-0136-4c22-9988-0108df775a03	None	2a3e2086-3516-40ac-a584-3714e91858f5	power on	active	False
5f0593b7-31de-4291-b43f-a549699cd470	None	f4cc50d4-441e-4728-9984-53df29f0b7f7	power on	active	False
99225e1b-085e-4ef7-8173-4687900b741a	None	200a918e-abb3-4539-a1c4-7e30f2d8ebc2	power on	active	False
c6ec143b-a522-4d69-ab31-5b4934ad3c42	None	7c675ed5-17d9-47ad-a2ef-592353e27713	power on	active	False
e1026c43-f2a3-44ad-a385-4d4462552977	None	45b45041-656f-4ee1-8be2-976c71a35b1f	power on	active	False
122188ea-09ae-486c-b225-17cf0defe025	None	bd38818e-36ca-4fd9-a65f-c4b0e5b34977	power on	active	False
f6ecf896-6e5e-4735-8727-942478dee58a	None	82a79351-5520-4e89-ae19-48c7b6f6b39f	power on	active	False
e6db159e-008e-4186-8967-92a9faeee368	None	986affe6-23ba-48ba-ae4e-0d2226aabf55	power on	active	False
44f3a434-eaf8-4b1a-97e5-6340d277fa4e	None	1f385454-3ddb-40bd-bc6e-a55ad69fff47	power on	active	False
7ab70571-64ea-439b-a0f4-34147d01dfbf	None	6f9f76ac-3cf7-4002-94ba-39bc6f0b4c40	power on	active	False
6d478a22-874c-4611-834d-21f4809f90ce	None	8e37407f-c784-4f5f-942f-2e2c36aa3fa4	power on	active	False

## Viewing the OpenStack Server List

0a57a5ad-d160-477e-807f-11997307bc9c	None	25b53356-9f02-4810-b722-efb6fd887879	power on
active	False		
6fff3d83-ed37-4934-89e0-d632aeb37b15	None	0ea048c0-6f4b-460d-99b2-796dd694c226	power on
active	False		
5496919c-c269-4860-b49a-e0d103a6a460	None	6a8e05aa-26fe-43bb-b464-ede86b9f4639	power on
active	False		
513b936d-1c52-4b0a-9ac4-4101fe812f07	None	b92c5720-7db9-417b-b3d5-023046788c8e	power on
active	False		

## Viewing the OpenStack Server List

Log on to the server on which OSP-D is running to ensure that stack components and verify they are active and running the same image by executing the following command:

```
openstack server list
```

**Example command output:**

ID	Image Name	Name	Status	Networks
9f0b2ff3-5234-42e9-81dd-c0ef5e454137	overcloud-full_20170620T011048	tb3-ultram-compute-3	ACTIVE	
ctlplane=192.200.0.133				
25b53356-9f02-4810-b722-efb6fd887879	overcloud-full_20170620T011048	tb3-ultram-compute-15	ACTIVE	
ctlplane=192.200.0.131				
986affe6-23ba-48ba-ae4e-0d2226aabf55	overcloud-full_20170620T011048	tb3-ultram-compute-11	ACTIVE	
ctlplane=192.200.0.128				
45b45041-656f-4ee1-8be2-976c71a35b1f	overcloud-full_20170620T011048	tb3-ultram-compute-8	ACTIVE	
ctlplane=192.200.0.130				
bd38818e-36ca-4fd9-a65f-c4b0e5b34977	overcloud-full_20170620T011048	tb3-ultram-compute-9	ACTIVE	
ctlplane=192.200.0.127				
82a79351-5520-4e89-ae19-48c7b6f6b39f	overcloud-full_20170620T011048	tb3-ultram-compute-10	ACTIVE	
ctlplane=192.200.0.126				
1f385454-3ddb-40bd-bc6e-a55ad69fff47	overcloud-full_20170620T011048	tb3-ultram-compute-12	ACTIVE	
ctlplane=192.200.0.118				
8e37407f-c784-4f5f-942f-2e2c36aa3fa4	overcloud-full_20170620T011048	tb3-ultram-compute-14	ACTIVE	
ctlplane=192.200.0.117				
315c7aea-acef-4341-aa9e-bcd594cae592	overcloud-full_20170620T011048	tb3-ultram-compute-2	ACTIVE	
ctlplane=192.200.0.114				
2a3e2086-3516-40ac-a584-3714e91858f5	overcloud-full_20170620T011048	tb3-ultram-compute-4	ACTIVE	
ctlplane=192.200.0.120				
b92c5720-7db9-417b-b3d5-023046788c8e	overcloud-full_20170620T011048	tb3-ultram-osd-compute-2	ACTIVE	
ctlplane=192.200.0.110				
7c675ed5-17d9-47ad-a2ef-592353e27713	overcloud-full_20170620T011048	tb3-ultram-compute-7	ACTIVE	
ctlplane=192.200.0.111				
0ea048c0-6f4b-460d-99b2-796dd694c226	overcloud-full_20170620T011048	tb3-ultram-osd-compute-0	ACTIVE	
ctlplane=192.200.0.112				
f4cc50d4-441e-4728-9984-53df29f0b7f7	overcloud-full_20170620T011048	tb3-ultram-compute-5	ACTIVE	
ctlplane=192.200.0.108				
dd0596b1-bd35-451a-85bc-c635e7fa6d14	overcloud-full_20170620T011048	tb3-ultram-controller-2	ACTIVE	
ctlplane=192.200.0.115				
85a9a708-5eae-4ea2-8b29-dc2acd6e515d	overcloud-full_20170620T011048	tb3-ultram-compute-1	ACTIVE	
ctlplane=192.200.0.102				
bc903f51-8483-4522-bcd7-ac396ac626b1	overcloud-full_20170620T011048	tb3-ultram-controller-0	ACTIVE	
ctlplane=192.200.0.105				
6a8e05aa-26fe-43bb-b464-ede86b9f4639	overcloud-full_20170620T011048	tb3-ultram-osd-compute-1	ACTIVE	
ctlplane=192.200.0.106				
200a918e-abb3-4539-a1c4-7e30f2d8ebc2	overcloud-full_20170620T011048	tb3-ultram-compute-6	ACTIVE	
ctlplane=192.200.0.109				
05fbfb44-ccd9-475d-b263-58b2deaf8554	overcloud-full_20170620T011048	tb3-ultram-controller-1	ACTIVE	
ctlplane=192.200.0.113				

```
| 96d2725c-9c70-4a66-9d3c-4a0356faf1c0 | tb3-ultram-compute-0 | ACTIVE |
ctlplane=192.200.0.107 | overcloud-full_20170620T011048 |
| 6f9f76ac-3cf7-4002-94ba-39bc6f0b4c40 | tb3-ultram-compute-13 | ACTIVE |
ctlplane=192.200.0.103 | overcloud-full_20170620T011048 |
```

## Viewing the OpenStack Stack Resource List

Log on to the server on which OSP-D is running to view the stack resources and their status by executing the following command:

```
openstack stack resource list name
```

**Example command output:**

```
-----
| resource_name | physical_resource_id |
resource_type | resource_status | updated_time |
-----
| UpdateWorkflow | 94270702-cd8b-4441-a09e-5c9da0c2d02b |
OS::TripleO::Tasks::UpdateWorkflow | CREATE_COMPLETE | 2017-06-27T22:04:00Z |
| CephStorageHostsDeployment | 196dbba7-5d66-4a9c-9308-f47ff4d8be2d |
OS::Heat::StructuredDeployments | CREATE_COMPLETE | 2017-06-27T22:04:00Z |
| OsdComputeAllNodesDeployment | 6a5775c0-03d8-453f-92d8-be6ea5aed853 |
OS::Heat::StructuredDeployments | CREATE_COMPLETE | 2017-06-27T22:04:00Z |
| BlockStorageHostsDeployment | 97b2f70a-c295-4437-9222-8248ec30badf |
OS::Heat::StructuredDeployments | CREATE_COMPLETE | 2017-06-27T22:04:00Z |
| CephStorage | 1bc20bb0-516a-4eb5-85e2-be9d30e2f6e8 |
OS::Heat::ResourceGroup | CREATE_COMPLETE | 2017-06-27T22:04:00Z |
| AllNodesDeploySteps | da9ead69-b83e-4cc9-86e8-8d823c02843b |
OS::TripleO::PostDeploySteps | CREATE_COMPLETE | 2017-06-27T22:04:00Z |
| CephStorageAllNodesDeployment | e5ee9df8-fae1-4641-9cfb-038c8f4eca85 |
OS::Heat::StructuredDeployments | CREATE_COMPLETE | 2017-06-27T22:04:00Z |
```

## Verifying Node Reachability

Log on to the server on which OSP-D is running to ensure the node reachability and availability by executing the following command:

```
for i in $(nova list| grep ACTIVE| awk '{print $12}' | sed 's\ctlplane=\g'
) ; do ssh heat-admin@${i} uptime ; done
```

This command establishes an SSH session with each node and report the system uptime. Investigate any node that does not reply or has an unexpected uptime.

**Example command output:**

```
14:47:10 up 18:15, 0 users, load average: 0.01, 0.02, 0.05
14:47:11 up 18:14, 0 users, load average: 9.50, 9.15, 12.32
14:47:11 up 18:14, 0 users, load average: 9.41, 9.09, 12.26
14:47:11 up 18:14, 0 users, load average: 10.41, 10.28, 10.49
14:47:12 up 18:15, 0 users, load average: 0.00, 0.02, 0.05
14:47:12 up 18:14, 0 users, load average: 0.18, 0.06, 0.06
14:47:12 up 18:15, 0 users, load average: 0.00, 0.03, 0.05
14:47:12 up 18:15, 0 users, load average: 0.00, 0.01, 0.05
14:47:13 up 18:14, 0 users, load average: 0.02, 0.02, 0.05
14:47:13 up 18:14, 0 users, load average: 8.23, 8.66, 12.29
14:47:13 up 18:14, 0 users, load average: 8.76, 8.87, 12.14
14:47:14 up 18:15, 0 users, load average: 0.01, 0.04, 0.05
14:47:14 up 18:15, 0 users, load average: 9.30, 9.08, 10.12
14:47:14 up 18:15, 0 users, load average: 0.01, 0.06, 0.05
14:47:14 up 18:14, 0 users, load average: 8.31, 8.61, 11.96
```

```

14:47:15 up 18:14,  0 users,  load average: 17.08, 12.09, 11.06
14:47:15 up 17:09,  0 users,  load average:  1.64,  1.33,  1.10
14:47:15 up 17:04,  0 users,  load average:  1.02,  0.77,  0.79
14:47:16 up 16:58,  0 users,  load average:  0.55,  0.63,  0.72
14:47:16 up 23:46,  0 users,  load average:  2.68,  3.46,  3.89
14:47:16 up 1 day, 5 min,  0 users,  load average:  4.10,  4.27,  4.44
14:47:17 up 23:53,  0 users,  load average:  1.90,  2.32,  2.24

```

## Verify NTP is running

To verify the operational status of NTP server:

1. Log on to the server on which OSP-D is running to ensure that NTP is running on all nodes in the cluster by executing the following command:

```

for i in $(nova list | grep ACTIVE | awk '{print $12}' | sed
's/ctlplane=\\g' ) ; do ssh heat-admin@${i} systemctl status ntpd |grep
Active; done

```

This command establishes an SSH session with each node and lists the ntpd status.

### Example command output:

```

Active: active (running) since Tue 2017-07-11 20:32:25 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:28 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:50 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:28 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:14 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:30 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:22 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:16 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:35 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:31 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:30 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:25 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:19 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:14 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:41 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 20:32:30 UTC; 18h ago
Active: active (running) since Tue 2017-07-11 21:37:32 UTC; 17h ago
Active: active (running) since Tue 2017-07-11 21:43:16 UTC; 17h ago
Active: active (running) since Tue 2017-07-11 21:48:57 UTC; 17h ago
Active: active (running) since Tue 2017-07-11 15:01:30 UTC; 23h ago
Active: active (running) since Tue 2017-07-11 14:42:10 UTC; 24h ago
Active: active (running) since Tue 2017-07-11 14:54:06 UTC; 23h ago

```

2. Verify that all the OpenStack nodes are synced to NTP server.

```

for i in $(nova list | grep -i overc- | awk '{print $12}' | sed
's/ctlplane=//g') ; do (ssh -o StrictHostKeyChecking=no heat-admin@${i}
sudo ntpstat | grep NTP) ; done

```

```

[stack@j19bxb-ospd ~]$ for i in $(nova list | grep -i overc- | awk '{print $12}' | sed
's/ctlplane=//g') ; do (ssh -o StrictHostKeyChecking=no heat-admin@${i} sudo ntpstat |
grep NTP) ; done
synchronised to NTP server (10.84.96.130) at stratum 3

```

```
synchronised to NTP server (10.84.96.130) at stratum 3
```

3. Check the NTP status on the server on which OSP-D is running by executing the following command:

```
systemctl status ntpd |grep Active
```

Investigate any node that is not actively running NTP.

## Checking OSP-D Server Health

### Verifying VM and Other Service Status and Quotas

Log on to the server on which OSP-D is running to verify that Overcloud VMs are active and running by executing the following commands:

```
cd /home/stack
source ~/<stack_name>rc-core
nova list
```



**Note** Overcloud VM status can also be checked through the Horizon GUI.

#### Example command output:

```

+-----+-----+-----+-----+-----+
| ID                                     | Name                                     |
+-----+-----+-----+-----+-----+
|                                     | Status | Task State | Power State | Networks |
+-----+-----+-----+-----+-----+
| 407891a2-85bb-4b84-a023-bca4ff304fc5 | auto-deploy-vm-uas-0 |
| ACTIVE | - | Running | mgmt=172.16.181.21, 10.84.123.13 |
+-----+-----+-----+-----+-----+
| bb4c06c5-b328-47bd-ac57-a72a9b4bb496 | auto-it-vm-uas-0 |
| ACTIVE | - | Running | mgmt=172.16.181.19, 10.84.123.12 |
+-----+-----+-----+-----+-----+
| fc0e47d3-e59e-41a3-9d8d-99371de1c4c5 | tb3-bxb-autovnf1-uas-0 |
| ACTIVE | - | Running |                                     |
tb3-bxb-autovnf1-uas-orchestration=172.17.180.10; tb3-bxb-autovnf1-uas-management=172.17.181.8 |
+-----+-----+-----+-----+-----+

```

```

| 8056eff1-913e-479a-ac44-22eba42ceee1 | tb3-bxb-autovnf1-uas-1
| ACTIVE | - | Running |
tb3-bxb-autovnf1-uas-orchestration=172.17.180.6; tb3-bxb-autovnf1-uas-management=172.17.181.12

|
| 4e9fab14-dad0-4789-bc52-1fac3e40b7cc | tb3-bxb-autovnf1-uas-2
| ACTIVE | - | Running |
tb3-bxb-autovnf1-uas-orchestration=172.17.180.13; tb3-bxb-autovnf1-uas-management=172.17.181.3

|
| 1a4e65e3-9f9d-429f-a604-6dfb45ef2a45 | tb3-bxb-vnfm1-ESC-0
| ACTIVE | - | Running |
tb3-bxb-autovnf1-uas-orchestration=172.17.180.3; tb3-bxb-autovnf1-uas-management=172.17.181.4

|
| 7f4ec2dc-e8a8-4f6c-bfce-8f29735e9fca | tb3-bxb-vnfm1-ESC-1
| ACTIVE | - | Running |
tb3-bxb-autovnf1-uas-orchestration=172.17.180.14; tb3-bxb-autovnf1-uas-management=172.17.181.5

|
| 1c9fc0bd-dc16-426f-b387-c2b75b3a1c16 |
tb3-bxb-vnfm1-em_tb3-bx_0_190729a1-c703-4e15-b0b3-795e2e876f55 | ACTIVE | - |
Running | tb3-bxb-autovnf1-uas-orchestration=172.17.180.4;
tb3-bxb-autovnf1-uas-management=172.17.181.9

|
| 9a407a06-929a-49ce-8bae-4df35b5f8b40 |
tb3-bxb-vnfm1-em_tb3-bx_0_92c5224b-1f1f-4f3f-8ac8-137be69ce473 | ACTIVE | - |
Running | tb3-bxb-autovnf1-uas-orchestration=172.17.180.5;
tb3-bxb-autovnf1-uas-management=172.17.181.10

|
| e4528022-6e7b-43f9-94f6-a6ab6289478d |
tb3-bxb-vnfm1-em_tb3-bx_0_d9f7ecb2-a7dc-439b-b492-5ce0402264ea | ACTIVE | - |
Running | tb3-bxb-autovnf1-uas-orchestration=172.17.180.2;
tb3-bxb-autovnf1-uas-management=172.17.181.7

|
| 2calle5b-8eec-456d-9001-1f2600605ad4 |
vnfd1-deployment_c1_0_5b287829-6a9d-4c0a-97d0-a5e0f645b767 | ACTIVE | - |
Running | tb3-bxb-autovnf1-uas-orchestration=172.17.180.16;
tb3-bxb-vnfm1-di-internal1=192.168.1.4; tb3-bxb-autovnf1-uas-management=172.17.181.15;
tb3-bxb-vnfm1-di-internal2=192.168.2.5

|
| 0bdbd9e3-926a-4abe-81b3-95dc42ea0676 |
vnfd1-deployment_c2_0_7074a450-5268-4c94-965b-8fb809410d14 | ACTIVE | - |
Running | tb3-bxb-autovnf1-uas-orchestration=172.17.180.15;
tb3-bxb-vnfm1-di-internal1=192.168.1.2; tb3-bxb-autovnf1-uas-management=172.17.181.18;
tb3-bxb-vnfm1-di-internal2=192.168.2.6

|
| 8b07a9b1-139f-4a12-b16e-d35cb17f6668 |
vnfd1-deployment_s10_0_f6d110f9-9e49-43fe-be14-4ab87ca3334c | ACTIVE | - |
Running | tb3-bxb-autovnf1-uas-orchestration=172.17.180.7;
tb3-bxb-vnfm1-di-internal1=192.168.1.8; tb3-bxb-vnfm1-service-network1=10.10.10.3,
10.10.10.10; tb3-bxb-vnfm1-service-network2=20.20.20.5, 20.20.20.4;
tb3-bxb-vnfm1-di-internal2=192.168.2.12 |
| 4ff0ce2e-1d97-4056-a7aa-018412c0385d |
vnfd1-deployment_s3_0_5380ef6c-6fe3-4e92-aa44-d94ef6e94235 | ACTIVE | - |
Running | tb3-bxb-autovnf1-uas-orchestration=172.17.180.19;
tb3-bxb-vnfm1-di-internal1=192.168.1.5; tb3-bxb-vnfm1-service-network1=10.10.10.7, 10.10.10.2;
tb3-bxb-vnfm1-service-network2=20.20.20.9, 20.20.20.6; tb3-bxb-vnfm1-di-internal2=192.168.2.8

|
| 3954cd6e-0f12-4d4b-8558-2e035c126d9a |
vnfd1-deployment_s4_0_e5ae4aa9-a90e-4bfe-aaff-82ffd8f7fe34 | ACTIVE | - |

```

```

Running      | tb3-bxb-autovnf1-uas-orchestration=172.17.180.8;
tb3-bxb-vnfml-di-internal1=192.168.1.9; tb3-bxb-vnfml-service-network1=10.10.10.13,
10.10.10.8; tb3-bxb-vnfml-service-network2=20.20.20.12, 20.20.20.10;
tb3-bxb-vnfml-di-internal2=192.168.2.3      |
| 2cc6728c-2982-42bf-bb8b-198a14fdbcb31 |
vnfd1-deployment_s5_0_1d57c15d-alde-40d4-aac2-1715f01ac50a      | ACTIVE | -
Running      | tb3-bxb-autovnf1-uas-orchestration=172.17.180.17;
tb3-bxb-vnfml-di-internal1=192.168.1.7; tb3-bxb-vnfml-service-network1=10.10.10.5,
10.10.10.18; tb3-bxb-vnfml-service-network2=20.20.20.11, 20.20.20.2;
tb3-bxb-vnfml-di-internal2=192.168.2.4      |
| 876cc650-ae8b-497b-805a-24a305be6c13 |
vnfd1-deployment_s6_0_05e13a62-623c-4749-ae2a-15c70dd12e16      | ACTIVE | -
Running      | tb3-bxb-autovnf1-uas-orchestration=172.17.180.11;
tb3-bxb-vnfml-di-internal1=192.168.1.6; tb3-bxb-vnfml-service-network1=10.10.10.12,
10.10.10.9; tb3-bxb-vnfml-service-network2=20.20.20.13, 20.20.20.18;
tb3-bxb-vnfml-di-internal2=192.168.2.16      |
| 89f7245e-c2f7-4041-b5e6-1eee48641cfd |
vnfd1-deployment_s7_0_3a4d7273-e808-4b5f-8877-7aa182483d93      | ACTIVE | -
Running      | tb3-bxb-autovnf1-uas-orchestration=172.17.180.24;
tb3-bxb-vnfml-di-internal1=192.168.1.12; tb3-bxb-vnfml-service-network1=10.10.10.14,
10.10.10.6; tb3-bxb-vnfml-service-network2=20.20.20.20, 20.20.20.8;
tb3-bxb-vnfml-di-internal2=192.168.2.7      |
| 535b0bca-d3c5-4d99-ba41-9953da6339f4 |
vnfd1-deployment_s8_0_1e0f3ebf-b6e0-4bfe-9b1c-985dc32e1519      | ACTIVE | -
Running      | tb3-bxb-autovnf1-uas-orchestration=172.17.180.18;
tb3-bxb-vnfml-di-internal1=192.168.1.14; tb3-bxb-vnfml-service-network1=10.10.10.17,
10.10.10.11; tb3-bxb-vnfml-service-network2=20.20.20.17, 20.20.20.15;
tb3-bxb-vnfml-di-internal2=192.168.2.9      |
| dfdffafb-a624-4063-bae6-63c4a757473f |
vnfd1-deployment_s9_0_26db8332-8dac-43fc-84c5-71a8b975fd17      | ACTIVE | -
Running      | tb3-bxb-autovnf1-uas-orchestration=172.17.180.22;
tb3-bxb-vnfml-di-internal1=192.168.1.10; tb3-bxb-vnfml-service-network1=10.10.10.21,
10.10.10.24; tb3-bxb-vnfml-service-network2=20.20.20.23, 20.20.20.22;
tb3-bxb-vnfml-di-internal2=192.168.2.19      |
+-----+-----+-----+-----+

```

## Checking Cinder Type

Log on to the server on which OSP-D is running to check the Cinder volume type by executing the following commands:

```

cd /home/stack
source ~/<stack_name>rc-core
cinder type-list

```

Example command output:

```

+-----+-----+-----+-----+
| ID                | Name | Description | Is_Public |
+-----+-----+-----+-----+
| 208ef179-dfe4-4735-8a96-e7beee472944 | LUKS | -           | True      |
+-----+-----+-----+-----+

```

```

cinder type-show LUKS

```

Example command output:

```

+-----+-----+-----+
| Property          | Value
+-----+-----+-----+
| description       | None
| extra_specs       | {}
| id                | bf855b0f-8b3f-42bf-9497-05013b4ddad9
+-----+-----+-----+

```

```

| is_public          | True          |
| name              | LUKS         |
| os-volume-type-access:is_public | True        |
| qos_specs_id     | None         |
+-----+-----+

```

## Checking Core Project (Tenant) and User Core

Log on to the server on which OSP-D is running to check the core projects and users by executing the following commands:

```

cd /home/stack
source ~/<stack_name> rc-core
openstack project list

```

**Example command output:**

```

+-----+-----+
| ID              | Name        |
+-----+-----+
| 271ab207a197465f9d166c2dc7304b18 | core       |
| 52547e0fca994cd682aa733b941d0f68 | service   |
| 9543ad9db4dd422ea5aedf04756d3682 | admin     |
+-----+-----+

```

```
openstack project show core
```

**Example command output:**

```

+-----+-----+
| Field          | Value      |
+-----+-----+
| description    | core tenant |
| enabled        | True       |
| id             | 271ab207a197465f9d166c2dc7304b18 |
| name           | core       |
| properties     |            |
+-----+-----+

```

```
openstack project show service
```

**Example command output:**

```

+-----+-----+
| Field          | Value      |
+-----+-----+
| description    | Tenant for the openstack services |
| enabled        | True       |
| id             | 52547e0fca994cd682aa733b941d0f68 |
| name           | service    |
| properties     |            |
+-----+-----+

```

```
openstack project show admin
```

**Example command output:**

```

+-----+-----+
| Field          | Value      |
+-----+-----+
| description    | admin tenant |
| enabled        | True       |
| id             | 9543ad9db4dd422ea5aedf04756d3682 |
| name           | admin       |
+-----+-----+

```

```
| properties |
+-----+
```

### openstack user list

#### Example command output:

```
+-----+
| ID | Name |
+-----+
| 1ac7208b033a41ccba805d86bf60dbb7 | admin |
| a6adac4ee79c4206a29de5165d7c7a6a | neutron |
| 79da40fe88c64de7a93bc691a42926ea | heat |
| 525048a99816474d91d692d9516e951c | nova |
| 8d6688db8d19411080eeb4c84c1d586b | glance |
| 9aadd12171474d1e8bcacf890e070ab | cinder |
| d2ee641a72c4493995de70a1a9671f2b | heat-cfn |
| 7fbb088c15e1428ab6ce677aad5415f4 | swift |
| 828cbf69cf564747a81bb313208a1c21 | core |
| 40563efc469d4c1295de0d6d4cf545c2 | tom |
+-----+
```

### openstack user show core

#### Example command output:

```
+-----+
| Field | Value |
+-----+
| email | None |
| enabled | True |
| id | 828cbf69cf564747a81bb313208a1c21 |
| name | core |
| project_id | 271ab207a197465f9d166c2dc7304b18 |
| username | core |
+-----+
```

### openstack role list

#### Example command output:

```
+-----+
| ID | Name |
+-----+
| 315d3058519a4b1a9385e11aa5ffe25b | admin |
| 585de968688e4257bc76f6dec13752cb | ResellerAdmin |
| 9717fe8079ba49e9ba9eadd5a37689e7 | swiftoperator |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
| d75dcf507bfa4a6abee3aee3bb0323c6 | heat_stack_user |
+-----+
```

### openstack role show admin

#### Example command output:

```
+-----+
| Field | Value |
+-----+
| domain_id | None |
| id | 315d3058519a4b1a9385e11aa5ffe25b |
| name | admin |
+-----+
```

## Checking Nova/Neutron Security Groups

Log on to the server on which OSP-D is running to check Nova and Neutron security groups by executing the following commands:

**nova secgroup-list**

### Example command output:

WARNING: Command secgroup-list is deprecated and will be removed after Nova 15.0.0 is released. Use python-neutronclient or python-openstackclient instead.

```

+-----+-----+-----+
| Id                | Name    | Description                |
+-----+-----+-----+
| ce308d67-7645-43c1-a83e-89d3871141a2 | default | Default security group |
+-----+-----+-----+

```

**neutron security-group-list**

### Example command output:

```

+-----+-----+-----+
| id                | name    | security_group_rules      |
+-----+-----+-----+
| 4007a7a4-e7fa-4ad6-bc74-fc0b20f0b60c | default | egress, IPv4              |
|                                     |         | egress, IPv6              |
|                                     |         | ingress, IPv4, remote_group_id: |
| 4007a7a4-e7fa-4ad6-bc74-fc0b20f0b60c |         | ingress, IPv6, remote_group_id: |
| 4007a7a4-e7fa-4ad6-bc74-fc0b20f0b60c |         |                               |
| 8bee29ae-88c0-4d5d-b27a-a123f20b6858 | default | egress, IPv4              |
|                                     |         | egress, IPv6              |
|                                     |         | ingress, IPv4, 1-65535/tcp,    |
| remote_ip_prefix: 0.0.0.0/0          |         | ingress, IPv4, 1-65535/udp,    |
| remote_ip_prefix: 0.0.0.0/0          |         |                               |
| 0.0.0.0/0                             |         | ingress, IPv4, icmp, remote_ip_prefix: |
|                                     |         | ingress, IPv4, remote_group_id: |
| 8bee29ae-88c0-4d5d-b27a-a123f20b6858 |         | ingress, IPv6, remote_group_id: |
| 8bee29ae-88c0-4d5d-b27a-a123f20b6858 |         |                               |
| b6b27428-35a3-4be4-af9b-38559132d28e | default | egress, IPv4              |
|                                     |         | egress, IPv6              |
|                                     |         | ingress, IPv4, remote_group_id: |
| b6b27428-35a3-4be4-af9b-38559132d28e |         | ingress, IPv6, remote_group_id: |
| b6b27428-35a3-4be4-af9b-38559132d28e |         |                               |
| ce308d67-7645-43c1-a83e-89d3871141a2 | default | egress, IPv4              |
|                                     |         | egress, IPv6              |
|                                     |         | ingress, IPv4, 1-65535/tcp,    |
| remote_ip_prefix: 0.0.0.0/0          |         | ingress, IPv4, 1-65535/udp,    |
| remote_ip_prefix: 0.0.0.0/0          |         |                               |
| 0.0.0.0/0                             |         | ingress, IPv4, icmp, remote_ip_prefix: |
+-----+-----+-----+

```

```

|                                     | ingress, IPv4, remote_group_id:
ce308d67-7645-43c1-a83e-89d3871141a2 |
|                                     | ingress, IPv6, remote_group_id:
ce308d67-7645-43c1-a83e-89d3871141a2 |
+-----+-----+-----+-----+

```

**neutron security-group-show ce308d67-7645-43c1-a83e-89d3871141a2**

**Example command output:**

```

+-----+-----+-----+-----+
| Field          | Value
+-----+-----+-----+-----+
| created_at     | 2017-06-03T04:57:01Z
| description    | Default security group
| id             | ce308d67-7645-43c1-a83e-89d3871141a2
| name           | default
| project_id     | 271ab207a197465f9d166c2dc7304b18
| revision_number | 4
| security_group_rules | {
|               |   "remote_group_id": null,
|               |   "direction": "egress",
|               |   "protocol": null,
|               |   "description": null,
|               |   "ethertype": "IPv4",
|               |   "remote_ip_prefix": null,
|               |   "port_range_max": null,
|               |   "updated_at": "2017-06-03T04:57:01Z",
|               |   "security_group_id": "ce308d67-7645-43c1-a83e-89d3871141a2",
|               |   "port_range_min": null,
|               |   "revision_number": 1,
|               |   "tenant_id": "271ab207a197465f9d166c2dc7304b18",
|               |   "created_at": "2017-06-03T04:57:01Z",
|               |   "project_id": "271ab207a197465f9d166c2dc7304b18",
|               |   "id": "337838dd-0612-47f8-99e8-7d4f58dc09d6"
|               | }
|               | {
|               |   "remote_group_id": null,

```





```

|                                     | "id": "88320991-5232-44f6-b74b-8cfe934165d0"
|                                     | }
|                                     | {
|                                     |
|                                     | "remote_group_id": "ce308d67-7645-43c1-a83e-89d3871141a2",
|                                     | "direction": "ingress",
|                                     | "protocol": null,
|                                     | "description": null,
|                                     | "ethertype": "IPv4",
|                                     | "remote_ip_prefix": null,
|                                     | "port_range_max": null,
|                                     | "updated_at": "2017-06-03T04:57:01Z",
|                                     | "security_group_id": "ce308d67-7645-43c1-a83e-89d3871141a2",
|                                     | "port_range_min": null,
|                                     | "revision_number": 1,
|                                     | "tenant_id": "271ab207a197465f9d166c2dc7304b18",
|                                     | "created_at": "2017-06-03T04:57:01Z",
|                                     | "project_id": "271ab207a197465f9d166c2dc7304b18",
|                                     | "id": "ba306ee2-d21f-48be-9de2-7f04bea5e43a"
|                                     | }
|                                     | {
|                                     |
|                                     | "remote_group_id": "ce308d67-7645-43c1-a83e-89d3871141a2",
|                                     | "direction": "ingress",
|                                     | "protocol": null,
|                                     | "description": null,
|                                     | "ethertype": "IPv6",
|                                     | "remote_ip_prefix": null,
|                                     | "port_range_max": null,
|                                     | "updated_at": "2017-06-03T04:57:01Z",
|                                     | "security_group_id": "ce308d67-7645-43c1-a83e-89d3871141a2",
|                                     | "port_range_min": null,
|                                     | "revision_number": 1,
|                                     | "tenant_id": "271ab207a197465f9d166c2dc7304b18",

```

```

|                                     |      "created_at": "2017-06-03T04:57:01Z",
|                                     |
|                                     |      "project_id": "271ab207a197465f9d166c2dc7304b18",
|                                     |
|                                     |      "id": "deb7752c-e642-462e-92f0-5dff983f0739"
|                                     | }
| tenant_id                          | 271ab207a197465f9d166c2dc7304b18
| updated_at                          | 2017-06-03T04:57:33Z
+-----+-----+

```

## Checking Tenant Project Default Quotas

Log on to the server on which OSP-D is running to check default project quotas by executing the following commands:

**nova quota-show**

**Example command output:**

```

+-----+-----+
| Quota                | Limit |
+-----+-----+
| instances            | 1000  |
| cores                | 1000  |
| ram                  | 5120000 |
| metadata_items      | 128   |
| injected_files       | 100   |
| injected_file_content_bytes | 1024000 |
| injected_file_path_bytes | 255   |
| key_pairs            | 100   |
| server_groups        | 10    |
| server_group_members | 10    |
+-----+-----+

```

**openstack project list | grep core**

**Example command output:**

```
| 271ab207a197465f9d166c2dc7304b18 | core |
```

**nova quota-class-show 271ab207a197465f9d166c2dc7304b18**

**Example command output:**

```

+-----+-----+
| Quota                | Limit |
+-----+-----+
| instances            | 10    |
| cores                | 20    |
| ram                  | 51200 |
| floating_ips         | 10    |
| fixed_ips            | -1    |
| metadata_items      | 128   |
| injected_files       | 5     |
| injected_file_content_bytes | 10240 |
| injected_file_path_bytes | 255   |
| key_pairs            | 100   |
| security_groups      | 10    |
| security_group_rules | 20    |
+-----+-----+

```

**neutron quota-show****Example command output:**

```
+-----+-----+
| Field          | Value |
+-----+-----+
| floatingip     | 100   |
| network        | 1000  |
| port           | 4092  |
| rbac_policy    | 10    |
| router         | 100   |
| security_group | 100   |
| security_group_rule | 300  |
| subnet         | 1000  |
| subnetpool     | -1    |
| trunk          | -1    |
+-----+-----+
```

**openstack project list | grep core****Example command output:**

```
| 271ab207a197465f9d166c2dc7304b18 | core |
```

**cinder quota-show 271ab207a197465f9d166c2dc7304b18****Example command output:**

```
+-----+-----+
| Property       | Value |
+-----+-----+
| backup_gigabytes | 1000  |
| backups         | 10    |
| gigabytes       | 8092  |
| gigabytes_LUKS  | -1    |
| per_volume_gigabytes | -1   |
| snapshots       | 300   |
| snapshots_LUKS  | -1    |
| volumes         | 500   |
| volumes_LUKS    | -1    |
+-----+-----+
```

## Checking the Nova Hypervisor List

Log on to the server on which OSP-D is running to check the status of nova api on all compute nodes by executing the following command:

**nova hypervisor-list****Example command output:**

```
+-----+-----+-----+-----+
| ID | Hypervisor hostname | State | Status |
+-----+-----+-----+-----+
| 3  | tb3-ultram-compute-7.localdomain | up   | enabled |
| 6  | tb3-ultram-compute-6.localdomain | up   | enabled |
| 9  | tb3-ultram-osd-compute-0.localdomain | up   | enabled |
| 12 | tb3-ultram-compute-9.localdomain | up   | enabled |
| 15 | tb3-ultram-compute-0.localdomain | up   | enabled |
| 18 | tb3-ultram-compute-14.localdomain | up   | enabled |
| 21 | tb3-ultram-compute-2.localdomain | up   | enabled |
| 24 | tb3-ultram-compute-8.localdomain | up   | enabled |
| 27 | tb3-ultram-compute-13.localdomain | up   | enabled |
| 30 | tb3-ultram-compute-15.localdomain | up   | enabled |
+-----+-----+-----+-----+
```

```

| 33 | tb3-ultram-compute-12.localdomain | up | enabled |
| 36 | tb3-ultram-compute-5.localdomain | up | enabled |
| 39 | tb3-ultram-osd-compute-1.localdomain | up | enabled |
| 42 | tb3-ultram-compute-10.localdomain | up | enabled |
| 45 | tb3-ultram-compute-11.localdomain | up | enabled |
| 48 | tb3-ultram-compute-3.localdomain | up | enabled |
| 51 | tb3-ultram-osd-compute-2.localdomain | up | enabled |
| 54 | tb3-ultram-compute-4.localdomain | up | enabled |
| 57 | tb3-ultram-compute-1.localdomain | up | enabled |
+-----+-----+-----+-----+

```

## Checking the Router Main Configuration

Log on to the server on which OSP-D is running to check the Neutron router by entering the following commands:

```
neutron router-list
```

**Example command output:**

```

+-----+-----+-----+-----+-----+
| id | distributed | ha | name | external_gateway_info |
+-----+-----+-----+-----+-----+
| 2d0cdee4-bb5e-415b-921c-97caf0aa0cd1 | main | {"network_id": |
"1c46790f-cab5-4b1d-afc7-a637fe2dbe08", | False | True |
| | | | "enable_snat": true, "external_fixed_ips": |
| [{"subnet_id": | | | | "a23a740e-3ad0-4fb1-8526-3353dfd0010f", |
"ip_address": | | | | "10.169.127.176"}]} |
| | | | |
+-----+-----+-----+-----+-----+

```

```
[stack@lbucs001-ospd ~]$ neutron router-show
2d0cdee4-bb5e-415b-921c-97caf0aa0cd1
```

**Example command output:**

```

+-----+-----+-----+-----+-----+
| Field | Value |
+-----+-----+-----+-----+-----+
| admin_state_up | True |
| availability_zone_hints | |
| availability_zones | nova |
| created_at | 2017-06-03T05:05:08Z |
| description | |
| distributed | False |
| external_gateway_info | {"network_id": "1c46790f-cab5-4b1d-afc7-a637fe2dbe08", |
"enable_snat": true, "external_fixed_ips": [{"subnet_id": |
| | "a23a740e-3ad0-4fb1-8526-3353dfd0010f", "ip_address": |
"10.169.127.176"}]} |
| flavor_id | |
| ha | True |
+-----+-----+-----+-----+-----+

```

## Checking the External Network Using the core-project-id

```

| id | 2d0cdee4-bb5e-415b-921c-97caf0aa0cd1 |
| name | main |
| project_id | 271ab207a197465f9d166c2dc7304b18 |
| revision_number | 94 |
| routes | |
| status | ACTIVE |
| tenant_id | 271ab207a197465f9d166c2dc7304b18 |
| updated_at | 2017-07-28T00:44:27Z |

```

## Checking the External Network Using the core-project-id

Log on to the server on which OSP-D is running to check the external network configuration by entering the following commands:

```
neutron net-list
```

**Example command output:**

```

+-----+-----+-----+
| id | name |
| subnets |
+-----+-----+-----+
| 1236bd98-5389-42f9-bac8-433997525549 | LBUCS001-AUTOIT-MGMT |
| c63451f2-7e44-432e-94fc-167f6a31e4aa | 172.16.182.0/24 |
| 1c46790f-cab5-4b1d-afc7-a637fe2dbe08 | LBUCS001-EXTERNAL-MGMT |
| a23a740e-3ad0-4fb1-8526-3353dfd0010f | 10.169.127.160/27 |
| 1c70a9ab-212e-4884-b7d5-4749c44a87b6 | LBPGW101-DI-INTERNAL1 |
| e619b02e-84e0-48d9-9096-f16adc84f1cc | HA network tenant 271ab207a197465f9d166c2dc7304b18 |
| cefd5f5f-0c97-4027-b385-ca1a57f2cfac | 169.254.192.0/18 |
+-----+-----+-----+

```

```
neutron net-show 1c46790f-cab5-4b1d-afc7-a637fe2dbe08
```

**Example command output:**

```

+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | True |
| availability_zone_hints | |
| availability_zones | |
| created_at | 2017-06-05T07:18:59Z |
| description | |
| id | 1c46790f-cab5-4b1d-afc7-a637fe2dbe08 |
| ipv4_address_scope | |
| ipv6_address_scope | |
| is_default | False |
| mtu | 1500 |
| name | LBUCS001-EXTERNAL-MGMT |
| port_security_enabled | True |
| project_id | 271ab207a197465f9d166c2dc7304b18 |
| provider:network_type | vlan |

```

```

| provider:physical_network | datacentre |
| provider:segmentation_id | 101 |
| qos_policy_id | |
| revision_number | 6 |
| router:external | True |
| shared | False |
| status | ACTIVE |
| subnets | a23a740e-3ad0-4fb1-8526-3353dfd0010f |
| tags | |
| tenant_id | 271ab207a197465f9d166c2dc7304b18 |
| updated_at | 2017-06-05T07:22:51Z |
+-----+-----+

```

Note down the **provider:segmentation\_id**. In this example, 101 is the vlan for the external interface.

### neutron subnet-list

#### Example command output:

```

+-----+-----+-----+-----+
| id | allocation_pools | name | cidr |
+-----+-----+-----+-----+
| a23a740e-3ad0-4fb1-8526-3353dfd0010f | LBUCS001-EXTERNAL-MGMT | 10.169.127.160/27 | {"start": "10.169.127.168", "end": "10.169.127.190"} |
| c63451f2-7e44-432e-94fc-167f6a31e4aa | LBUCS001-AUTOIT-MGMT | 172.16.182.0/24 | {"start": "172.16.182.2", "end": "172.16.182.254"} |
| cefd5f5f-0c97-4027-b385-cala57f2cfac | HA subnet tenant | 169.254.192.0/18 | {"start": "169.254.192.1", "end": "169.254.255.254"} |
| | | 271ab207a197465f9d166c2dc7304b18 | |
+-----+-----+-----+-----+

```

### neutron subnet-show a23a740e-3ad0-4fb1-8526-3353dfd0010f

#### Example command output:

```

+-----+-----+
| Field | Value |
+-----+-----+
| allocation_pools | {"start": "10.169.127.168", "end": "10.169.127.190"} |
| cidr | 10.169.127.160/27 |
| created_at | 2017-06-05T07:22:51Z |
| description | |
| dns_nameservers | |
| enable_dhcp | False |
| gateway_ip | 10.169.127.163 |
| host_routes | |
| id | a23a740e-3ad0-4fb1-8526-3353dfd0010f |
| ip_version | 4 |
| ipv6_address_mode | |
| ipv6_ra_mode | |
| name | LBUCS001-EXTERNAL-MGMT |
| network_id | 1c46790f-cab5-4b1d-afc7-a637fe2dbe08 |
| project_id | 271ab207a197465f9d166c2dc7304b18 |
| revision_number | 2 |
| service_types | |
| subnetpool_id | |
| tenant_id | 271ab207a197465f9d166c2dc7304b18 |
| updated_at | 2017-06-05T07:22:51Z |
+-----+-----+

```

## Checking the Staging Network Configuration

Log on to the server on which OSP-D is running to check the staging network configuration by entering the following commands:

```
neutron subnet-show <ext-mgmt-id>
```

<ext-mgmt-id> is the ID for the external management interface as obtained through the **neutron subnet-list** command output.

### Example output:

```
+-----+-----+
| Field          | Value                                                                 |
+-----+-----+
| allocation_pools | {"start": "10.169.127.168", "end": "10.169.127.190"} |
| cidr            | 10.169.127.160/27 |
| created_at      | 2017-06-05T07:22:51Z |
| description     | |
| dns_nameservers | |
| enable_dhcp     | False |
| gateway_ip      | 10.169.127.163 |
| host_routes     | |
| id              | a23a740e-3ad0-4fb1-8526-3353dfd0010f |
| ip_version      | 4 |
| ipv6_address_mode | |
| ipv6_ra_mode    | |
| name            | LBUCS001-EXTERNAL-MGMT |
| network_id      | 1c46790f-cab5-4b1d-afc7-a637fe2dbe08 |
| project_id      | 271ab207a197465f9d166c2dc7304b18 |
| revision_number | 2 |
| service_types   | |
| subnetpool_id   | |
| tenant_id       | 271ab207a197465f9d166c2dc7304b18 |
| updated_at      | 2017-06-05T07:22:51Z |
+-----+-----+
```

```
neutron subnet-show <autoit-mgmt-id>
```

<autoit-mgmt-id> is the ID for the AutoIT management interface as obtained through the **neutron subnet-list** command output.

### Example output:

```
+-----+-----+
| Field          | Value                                                                 |
+-----+-----+
| allocation_pools | {"start": "172.16.182.2", "end": "172.16.182.254"} |
| cidr            | 172.16.182.0/24 |
| created_at      | 2017-06-05T07:41:45Z |
| description     | |
| dns_nameservers | |
| enable_dhcp     | True |
| gateway_ip      | 172.16.182.1 |
| host_routes     | |
| id              | c63451f2-7e44-432e-94fc-167f6a31e4aa |
| ip_version      | 4 |
| ipv6_address_mode | |
| ipv6_ra_mode    | |
| name            | LBUCS001-AUTOIT-MGMT |
| network_id      | 1236bd98-5389-42f9-bac8-433997525549 |
| project_id      | 271ab207a197465f9d166c2dc7304b18 |
| revision_number | 2 |
| service_types   | |
+-----+-----+
```

```

| subnetpool_id      |
| tenant_id         | 271ab207a197465f9d166c2dc7304b18 |
| updated_at        | 2017-06-05T07:41:45Z             |
+-----+-----+

```

## Checking the DI-Internal and Service Network Configurations

Log on to the server on which OSP-D is running to check the DI-internal and service network configuration by entering the following commands:

```
neutron net-list
```

**Example command output:**

```

+-----+-----+-----+-----+
| id                | name                |
| subnets          |                     |
+-----+-----+-----+-----+
| 1236bd98-5389-42f9-bac8-433997525549 | LBUCS001-AUTOIT-MGMT |
| c63451f2-7e44-432e-94fc-167f6a31e4aa | 172.16.182.0/24      |
| 1c46790f-cab5-4b1d-afc7-a637fe2dbe08 | LBUCS001-EXTERNAL-MGMT |
| a23a740e-3ad0-4fb1-8526-3353dfd0010f | 10.169.127.160/27   |
| 1c70a9ab-212e-4884-b7d5-4749c44a87b6 | LBPGW101-DI-INTERNAL1 |
| e619b02e-84e0-48d9-9096-f16adc84f1cc | HA network tenant 271ab207a197465f9d166c2dc7304b18 |
| cefd5f5f-0c97-4027-b385-ca1a57f2cfac | 169.254.192.0/18   |
+-----+-----+-----+-----+

```

```
neutron net-show LBPGW101-DI-INTERNAL1
```

**Example command output:**

```

+-----+-----+
| Field                | Value                |
+-----+-----+
| admin_state_up      | True                 |
| availability_zone_hints |                     |
| availability_zones   |                     |
| created_at          | 2017-07-28T22:25:53Z |
| description         |                     |
| id                  | 1c70a9ab-212e-4884-b7d5-4749c44a87b6 |
| ipv4_address_scope  |                     |
| ipv6_address_scope  |                     |
| mtu                 | 1500                |
| name                | LBPGW101-DI-INTERNAL1 |
| port_security_enabled | True                 |
| project_id          | 271ab207a197465f9d166c2dc7304b18 |
| provider:network_type | flat                 |
| provider:physical_network | phys_pcie1_0        |
| provider:segmentation_id |                     |
| qos_policy_id       |                     |
| revision_number     | 3                   |
| router:external     | False                |
| shared              | True                 |
| status              | ACTIVE               |
| subnets            |                     |
| tags                |                     |
| tenant_id           | 271ab207a197465f9d166c2dc7304b18 |
| updated_at          | 2017-07-28T22:25:53Z |
+-----+-----+

```

```
neutron subnet-list
```

**Example command output:**

id	allocation_pools	name	cidr
96ae7e6e-f2e9-4fa5-a816-769c5a79f8f4	{ "start": "192.168.1.2", "end": "192.168.1.254" }	LBPGW101-DI-INTERNAL1-SUBNET	
a23a740e-3ad0-4fb1-8526-3353dfd0010f	{ "start": "10.169.127.168", "end": "10.169.127.190" }	LBUCS001-EXTERNAL-MGMT	
c63451f2-7e44-432e-94fc-167f6a31e4aa	{ "start": "172.16.182.2", "end": "172.16.182.254" }	LBUCS001-AUTOIT-MGMT	
cefd5f5f-0c97-4027-b385-ca1a57f2cfac	{ "start": "169.254.192.1", "end": "169.254.255.254" }	HA subnet tenant	271ab207a197465f9d166c2dc7304b18

## Checking the Flavor List

Log on to the server on which OSP-D is running to check the flavor list and to by entering the following command:

```
nova flavor-list
```

**Example command output:**

ID	Name	Memory_MB	Disk	Ephemeral
eff0335b-3374-46c3-a3de-9f4b1ccea04	DNUCS002-AUTOIT-FLAVOR	8192	80	0

## Checking Host Aggregate and Availability Zone Configuration

Log on to the server on which OSP-D is running to check the host aggregate and availability zone configurations for the OSD Compute and for the AutoDeploy and AutoIT VMs.



**Note** It is assumed that the AutoDeploy and AutoIT VMs reside on the same OSD Compute node.

This is done by executing the following commands:

```
cd /home/stack  
source ~/<stack_name>rc-core  
nova aggregate-list
```

**Example command output:**

Id	Name	Availability Zone
108	LBUCS001-AUTOIT	mgmt

```
| 147 | LBPGW101-EM-MGMT1 | - |
| 150 | LBPGW101-SERVICE1 | - |
| 153 | LBPGW101-CF-MGMT1 | - |
+-----+-----+-----+
```

#### **nova aggregate-show LBUCS001-AUTOIT**

```
+-----+-----+-----+-----+-----+
| Id | Name | Availability Zone | Hosts | Metadata |
+-----+-----+-----+-----+-----+
| 108 | LBUCS001-AUTOIT | mgmt | 'newtonoc-osd-compute-0.localdomain' | 'availability_zone=mgmt', 'mgmt=true' |
+-----+-----+-----+-----+-----+
```



**Note** This information can also be verified through the Horizon GUI. Login to Horizon as the user core and navigate to **Project > Compute > Instances**. Check each instance to verify that the status is Active and the power state is Running.

Correct any instance that does not meet these criteria before continuing.

## Checking Controller Server Health



**Note** The commands in this section should be executed on any one of the Controller nodes and do not need to be repeated on the other Controller nodes unless an issue is observed.

## Checking the Pacemaker Cluster Stack (PCS) Status

Log on to one of the Controller nodes and verify that the group of resources in the PCS cluster are active and in the expected state by executing the following command:

```
sudo pcs status
```

#### **Example command output:**

```
Cluster name: tripleo_cluster
Stack: corosync
Current DC: tb3-ultram-controller-0 (version 1.1.15-11.e17_3.4-e174ec8) - partition with quorum
Last updated: Wed Jul 12 13:28:56 2017 Last change: Tue Jul 11 21:45:09 2017 by root via crm_attribute on tb3-ultram-controller-0
```

```
3 nodes and 22 resources configured
```

```
Online: [ tb3-ultram-controller-0 tb3-ultram-controller-1 tb3-ultram-controller-2 ]
```

```
Full list of resources:
```

```
ip-192.200.0.104 (ocf::heartbeat:IPaddr2): Started tb3-ultram-controller-1
ip-10.84.123.6 (ocf::heartbeat:IPaddr2): Started tb3-ultram-controller-0
ip-11.119.0.42 (ocf::heartbeat:IPaddr2): Started tb3-ultram-controller-0
Clone Set: haproxy-clone [haproxy]
Started: [ tb3-ultram-controller-0 tb3-ultram-controller-1 tb3-ultram-controller-2 ]
Master/Slave Set: galera-master [galera]
```

```

Masters: [ tb3-ultram-controller-0 tb3-ultram-controller-1 tb3-ultram-controller-2 ]
ip-11.120.0.47 (ocf::heartbeat:IPAddr2):      Started tb3-ultram-controller-1
ip-11.118.0.49 (ocf::heartbeat:IPAddr2):      Started tb3-ultram-controller-0
Clone Set: rabbitmq-clone [rabbitmq]
Started: [ tb3-ultram-controller-0 tb3-ultram-controller-1 tb3-ultram-controller-2 ]
ip-11.120.0.48 (ocf::heartbeat:IPAddr2):      Started tb3-ultram-controller-1
Master/Slave Set: redis-master [redis]
Masters: [ tb3-ultram-controller-0 ]
Slaves: [ tb3-ultram-controller-1 tb3-ultram-controller-2 ]
openstack-cinder-volume (systemd:openstack-cinder-volume): Started
tb3-ultram-controller-0
my-ipmilan-for-controller-0 (stonith:fence_ipmilan): Started
tb3-ultram-controller-0
my-ipmilan-for-controller-1 (stonith:fence_ipmilan): Started
tb3-ultram-controller-1
my-ipmilan-for-controller-2 (stonith:fence_ipmilan): Started
tb3-ultram-controller-0

Daemon Status:
corosync: active/enabled
pacemaker: active/enabled
pcsd: active/enabled

```

From the output of this command, ensure that:

- All 3 controllers are listed as Online
- haproxy-clone is started on all 3 controllers
- galera-master lists all 3 controllers as Masters
- rabbitmq-clone is started on all 3 controllers
- redis-master lists one controller as master and the other 2 controllers as slaves
- openstack-cinder-volume is started on one node
- my-ipmilan/stonith is started on all 3 controllers
- Daemons corosync, pacemaker and pcsd are active and enabled




---

**Note** If the output displays any “Failed Actions”, execute the **sudo pcs resource cleanup** command and then re-execute the **sudo pcs status** command.

---

## Checking Ceph Storage Status

Log on to the Controller node and verify the health of the Ceph storage from the Controller node by executing the following command:

```
sudo ceph status
```

### Example command output:

```

cluster eb2bb192-b1c9-11e6-9205-525400330666
health HEALTH_OK
monmap e1: 3 mons at
{tb3-ultram-controller-0=11.118.0.10:6789/0,tb3-ultram-controller-1=11.118.0.11:6789/0,
tb3-ultram-controller-2=11.118.0.12:6789/0}
election epoch 152, quorum 0,1,2
tb3-ultram-controller-0,tb3-ultram-controller-1,tb3-ultram-controller-2

```

```

osdmap e158: 12 osds: 12 up, 12 in
      flags sortbitwise,require_jewel_osds
pgmap v1417251: 704 pgs, 6 pools, 321 GB data, 110 kobjects
      961 GB used, 12431 GB / 13393 GB avail
      704 active+clean
client io 53755 B/s wr, 0 op/s rd, 7 op/s wr

```

From the output of this command, ensure that:

- health is listed as HEALTH\_OK
- The correct number of monitors are listed in the monmap
- The correct number of OSDs are listed in the osdmap

## Checking Controller Node Services

Log on to the Controller node and check the status of all services by executing the following command:

```
sudo systemctl list-units "openstack*" "neutron*" "openvswitch"
```

**Example command output:**

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
neutron-dhcp-agent.service Agent	loaded	active	running	OpenStack Neutron DHCP
neutron-l3-agent.service 3 Agent	loaded	active	running	OpenStack Neutron Layer
neutron-metadata-agent.service Agent	loaded	active	running	OpenStack Neutron Metadata
neutron-openvswitch-agent.service vSwitch Agent	loaded	active	running	OpenStack Neutron Open
neutron-ovs-cleanup.service vSwitch Cleanup Utility	loaded	active	exited	OpenStack Neutron Open
neutron-server.service	loaded	active	running	OpenStack Neutron Server
openstack-cinder-api.service Server	loaded	active	running	OpenStack Cinder API
openstack-cinder-scheduler.service Server	loaded	active	running	OpenStack Cinder Scheduler
openstack-cinder-volume.service openstack-cinder-volume	loaded	active	running	Cluster Controlled
openstack-glance-api.service (code-named Glance) API server	loaded	active	running	OpenStack Image Service
openstack-glance-registry.service (code-named Glance) Registry server	loaded	active	running	OpenStack Image Service
openstack-heat-api-cfn.service CFN-compatible API Service	loaded	active	running	Openstack Heat
openstack-heat-api-cloudwatch.service API Service	loaded	active	running	OpenStack Heat CloudWatch
openstack-heat-api.service	loaded	active	running	OpenStack Heat API Service
openstack-heat-engine.service Service	loaded	active	running	Openstack Heat Engine
openstack-nova-api.service	loaded	active	running	OpenStack Nova API Server
openstack-nova-conductor.service Server	loaded	active	running	OpenStack Nova Conductor
openstack-nova-consoleauth.service auth Server	loaded	active	running	OpenStack Nova VNC console
openstack-nova-novncproxy.service Proxy Server	loaded	active	running	OpenStack Nova NoVNC
openstack-nova-scheduler.service Server	loaded	active	running	OpenStack Nova Scheduler
openstack-swift-account-auditor.service (swift) - Account Auditor	loaded	active	running	OpenStack Object Storage
openstack-swift-account-reaper.service	loaded	active	running	OpenStack Object Storage

```

(swift) - Account Reaper
openstack-swift-account-replicator.service loaded active running OpenStack Object Storage
(swift) - Account Replicator
openstack-swift-account.service loaded active running OpenStack Object Storage
(swift) - Account Server
openstack-swift-container-auditor.service loaded active running OpenStack Object Storage
(swift) - Container Auditor
openstack-swift-container-replicator.service loaded active running OpenStack Object Storage
(swift) - Container Replicator
openstack-swift-container-updater.service loaded active running OpenStack Object Storage
(swift) - Container Updater
openstack-swift-container.service loaded active running OpenStack Object Storage
(swift) - Container Server
openstack-swift-object-auditor.service loaded active running OpenStack Object Storage
(swift) - Object Auditor
openstack-swift-object-expirer.service loaded active running OpenStack Object Storage
(swift) - Object Expirer
openstack-swift-object-replicator.service loaded active running OpenStack Object Storage
(swift) - Object Replicator
openstack-swift-object-updater.service loaded active running OpenStack Object Storage
(swift) - Object Updater
openstack-swift-object.service loaded active running OpenStack Object Storage
(swift) - Object Server
openstack-swift-proxy.service loaded active running OpenStack Object Storage
(swift) - Proxy Server
openvswitch.service loaded active exited Open vSwitch

```

LOAD = Reflects whether the unit definition was properly loaded.

ACTIVE = The high-level unit activation state, i.e. generalization of SUB.

SUB = The low-level unit activation state, values depend on unit type.

43 loaded units listed. Pass --all to see loaded but inactive units, too.

To show all installed unit files use 'systemctl list-unit-files'.

## Check the RabbitMQ Database Status

From each of the controller nodes, determine if the rabbitmq database is in a good state by executing the following command:

```
sudo rabbitmqctl eval 'rabbit_diagnostics:maybe_stuck().'
```

### Example command output:

```

2017-07-20 01:58:02 There are 11020 processes.
2017-07-20 01:58:02 Investigated 0 processes this round, 5000ms to go.
2017-07-20 01:58:03 Investigated 0 processes this round, 4500ms to go.
2017-07-20 01:58:03 Investigated 0 processes this round, 4000ms to go.
2017-07-20 01:58:04 Investigated 0 processes this round, 3500ms to go.
2017-07-20 01:58:04 Investigated 0 processes this round, 3000ms to go.
2017-07-20 01:58:05 Investigated 0 processes this round, 2500ms to go.
2017-07-20 01:58:05 Investigated 0 processes this round, 2000ms to go.
2017-07-20 01:58:06 Investigated 0 processes this round, 1500ms to go.
2017-07-20 01:58:06 Investigated 0 processes this round, 1000ms to go.
2017-07-20 01:58:07 Investigated 0 processes this round, 500ms to go.
2017-07-20 01:58:07 Found 0 suspicious processes.
ok

```

If the database is healthy, the command returns “Found 0 suspicious processes.” If the database is not healthy, the command returns 1 or more suspicious processes. Contact your local support representative if suspicious processes are found.

## Checking OSD Compute Server Health

### Checking Ceph Status

Log on to the OSD Compute and check the Ceph storage status by executing the following command:

```
sudo ceph status
```

#### Example command output:

```
sudo ceph status
  cluster eb2bb192-b1c9-11e6-9205-525400330666
  health HEALTH_OK
  monmap e1: 3 mons at
{tb3-ultram-controller-0=11.118.0.10:6789/0,tb3-ultram-controller-1=11.118.0.11:6789/0,
tb3-ultram-controller-2=11.118.0.12:6789/0}
  election epoch 152, quorum 0,1,2
tb3-ultram-controller-0,tb3-ultram-controller-1,tb3-ultram-controller-2
  osdmap e158: 12 osds: 12 up, 12 in
  flags sortbitwise,require_jewel_osds
  pgmap v1417867: 704 pgs, 6 pools, 321 GB data, 110 kobjects
  961 GB used, 12431 GB / 13393 GB avail
  704 active+clean
  client io 170 kB/s wr, 0 op/s rd, 24 op/s wr
```

### Checking OSD Compute Node Services

Log on to each OSD Compute node and check the status of all services by executing the following command:

```
sudo systemctl list-units "openstack*" "neutron*" "openvswitch"
```

#### Example command output:

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
neutron-openvswitch-agent.service	loaded	active	running	OpenStack Neutron Open vSwitch Agent
neutron-ovs-cleanup.service	loaded	active	exited	OpenStack Neutron Open vSwitch Cleanup Utility
neutron-sriov-nic-agent.service	loaded	active	running	OpenStack Neutron SR-IOV NIC Agent
openstack-nova-compute.service	loaded	active	running	OpenStack Nova Compute Server
openvswitch.service	loaded	active	exited	Open vSwitch

LOAD = Reflects whether the unit definition was properly loaded.

ACTIVE = The high-level unit activation state, i.e. generalization of SUB.

SUB = The low-level unit activation state, values depend on unit type.

6 loaded units listed. Pass --all to see loaded but inactive units, too.

To show all installed unit files use 'systemctl list-unit-files'.

## Monitoring AutoDeploy Operations

This section identifies various commands that can be used to determine the status and health of AutoDeploy.

To use them, you must:

1. Log on to the AutoDeploy VM as *ubuntu*. Use the password that was created earlier for this user.
2. Become the root user.

```
sudo -i
```

## Viewing AutoDeploy Logs

AutoDeploy logs are available on the AutoDeploy VM in the following directory:

```
/var/log/upstart/autodeploy.log
```



### Important

To access the command used to view logs, you must be logged in to the Confd CLI as the *admin* user on the AutoDeploy VM:

```
confd_cli -u admin -C
```

When prompted, you must enter the *admin* user password.

## AutoDeploy Transaction Logs

Execute the following command to display AutoDeploy transaction logs:

```
show log $TX-ID | display xml
```

### Example VIM-ORCH and VIM Activation Log:

```
2018-01-23 22:01:56,266 - Send Deployment notification for: autoit-instance
2018-01-23 22:08:36,876 - Deployment activate-ns-deployment: autoit initiated
2018-01-23 22:08:36,919 - Send Deployment notification for: autoit-instance
2018-01-23 22:08:36,951 - Deployment activate-ns-deployment: autoit initiated
2018-01-23 22:08:37,004 - Send Deployment notification for: autoit-deploy
2018-01-23 22:08:37,029 - Image '/var/cisco/isos/rhel-server-7.3-x86_64-dvd.iso' exists
2018-01-23 22:08:37,134 - Send Deployment notification for: autoit-instance
2018-01-23 22:08:37,165 - Deployment activate-ns-deployment: autoit started
2018-01-23 22:08:37,181 - Adding NSR: autoit-instance
2018-01-23 22:08:37,215 - Start pipeline of 1 tasks
2018-01-23 22:08:37,257 - Scheduling Task: autoit
2018-01-23 22:08:37,269 - Waiting for all workers to finish the transactions
2018-01-23 22:08:37,364 - Send Deployment notification for: autoit-deploy
2018-01-23 22:08:37,387 - Deployment activate-ns-deployment: autoit started
2018-01-23 22:08:37,395 - Skipping VNF pre-deployment , since VNF is not defined
2018-01-23 22:08:37,424 - Skipping VNF-Package pre-deployment, since is not defined
2018-01-23 22:08:37,440 - Skipping VIM-Artifact pre-deployment, since VIM-Artifact is not
defined
2018-01-23 22:08:37,463 - VIM-Orchestrator deployment pre-check success, entry already
exists. Continuing...
2018-01-23 22:08:37,470 - VIM deployment pre-check success, entry already exists.
Continuing...
2018-01-23 22:08:37,501 - NS pre-check success
2018-01-23 22:08:37,513 - Copying '/var/cisco/isos/rhel-server-7.3-x86_64-dvd.iso' to
'/var/cisco/isos/underc_rhel-server-7.3-x86_64-dvd.iso'
/tmp/MEIulQrBS/Crypto/Cipher/blockalgo.py:141: FutureWarning: CTR mode needs counter
parameter, not IV
2018-01-23 22:09:00,685 - Connected to AutoIT[172.21.203.121]
2018-01-23 22:09:02,281 - Skipping VNFDs
2018-01-23 22:09:02,298 - Skipping VNF-PACKAGE
2018-01-23 22:09:02,314 - Skipping VIM-Artifact
2018-01-23 22:09:02,332 - XML:[<config>
  <nsd xmlns="http://www.cisco.com/usp/nfv/usp-nsds">
    <nsd-id>autoit</nsd-id>
    <vim-orch>underc</vim-orch>
```

```

    <vim>overc</vim>
  </nsd>
  <vim-orchd xmlns="http://www.cisco.com/usp/nfv/usp-vim-orch">
    <vim-orch-id>underc</vim-orch-id>
    <hostname>tb3-undercloud</hostname>
    <domain-name>cisco.com</domain-name>
  .
  .
  .
2018-01-23 22:38:53,531 - VIM-ORCH: in-progress:84/84
2018-01-23 22:38:53,781 - Received vim-orchestrator-deployment-event for
underc:1516745343-313472/1516745343-460684 with status:success
2018-01-23 22:38:53,811 - VIM-ORCH: success:None/None
2018-01-23 22:38:53,983 - Received vim-deployment-event for
overc:1516745343-313472/1516745343-581981 with status:in-progress
2018-01-23 22:38:54,426 - Received vim-deployment-event for
overc:1516745343-313472/1516745343-581981 with status:in-progress
2018-01-23 23:39:15,038 - Received vim-deployment-event for
overc:1516745343-313472/1516745343-581981 with status:success
2018-01-23 23:39:15,113 - Received ns-deployment-event for autoit:1516745343-313472 with
status:success
2018-01-23 23:39:15,167 - RPC NS[autoit:autoit-instance] success
2018-01-23 23:39:15,271 - Deployment activate-ns-deployment: autoit succeeded
2018-01-23 23:39:15,344 - Send Deployment notification for: autoit-deploy
No handlers could be found for logger "AutoVNF-Traces"
2018-01-23 23:39:15,518 - All workers finished the job
2018-01-23 23:39:15,532 - Deployment activate-ns-deployment: autoit succeeded
2018-01-23 23:39:15,571 - Send Deployment notification for: autoit-instance

```

### Example Tenant Creation Log:

```

2018-01-23 23:48:54,420 - Deployment activate-ns-deployment: autoit initiated
2018-01-23 23:48:54,449 - Send Deployment notification for: autoit-instance
2018-01-23 23:48:54,465 - Parsing role for tenant 'sjccore'
2018-01-23 23:48:54,473 - Parsing credentials for tenant 'sjccore'
2018-01-23 23:48:54,484 - Parsing attributes for tenant 'sjccore'
2018-01-23 23:48:54,540 - Deployment activate-ns-deployment: autoit initiated
2018-01-23 23:48:54,574 - Send Deployment notification for: autoit-deploy
2018-01-23 23:48:54,599 - Image '/var/cisco/isos/rhel-server-7.3-x86_64-dvd.iso' exists
2018-01-23 23:48:54,666 - Send Deployment notification for: autoit-instance
2018-01-23 23:48:54,689 - Deployment activate-ns-deployment: autoit started
2018-01-23 23:48:54,691 - Adding NSR: autoit-instance
2018-01-23 23:48:54,712 - Start pipeline of 1 tasks
2018-01-23 23:48:54,723 - Scheduling Task: autoit
2018-01-23 23:48:54,749 - Waiting for all workers to finish the transactions
2018-01-23 23:48:54,804 - Send Deployment notification for: autoit-deploy
2018-01-23 23:48:54,806 - Deployment activate-ns-deployment: autoit started
2018-01-23 23:48:54,822 - Skipping VNF pre-deployment , since VNFD is not defined
2018-01-23 23:48:54,829 - Skipping VNF-Package pre-deployment, since is not defined
2018-01-23 23:48:54,862 - VIM-Artifact deployment pre-check success
2018-01-23 23:48:54,866 - VIM-Orchestrator deployment pre-check success, entry already
exists. Continuing...
2018-01-23 23:48:54,879 - VIM deployment pre-check success, entry already exists.
Continuing...
2018-01-23 23:48:54,885 - NS pre-check success
2018-01-23 23:48:54,895 - Skipping copy, file
'/var/cisco/isos/underc_rhel-cisver-7.3-x86_64-dvd.iso' already exists
/tmp/_MEIulQrBS/Crypto/Cipher/blockalgo.py:141: FutureWarning: CTR mode needs counter
parameter, not IV
2018-01-23 23:48:55,244 - Connected to AutoIT[172.21.203.121]
2018-01-23 23:48:55,259 - Skipping VNFDs
2018-01-23 23:48:55,274 - Skipping VNF-PACKAGE
2018-01-23 23:48:55,279 - XML:[<config>
  <nsd xmlns="http://www.cisco.com/usp/nfv/usp-nsds">
    <nsd-id>autoit</nsd-id>

```

```

<vim-identity>vim1</vim-identity>
.
.
.
2018-01-23 23:48:56,419 - Received vim-orchestrator-deployment-event for
underc:1516751336-209342/1516751336-428695 with status:success
2018-01-23 23:48:56,441 - VIM-ORCH: success:None/None
2018-01-23 23:48:56,540 - Received vim-deployment-event for
overc:1516751336-209342/1516751336-532373 with status:in-progress
2018-01-23 23:48:56,671 - Received vim-deployment-event for
overc:1516751336-209342/1516751336-532373 with status:success
2018-01-23 23:48:56,802 - Received vim-deployment-event for
sjccore:1516751336-209342/1516751336-654858 with status:in-progress
2018-01-23 23:49:13,305 - Received vim-deployment-event for
sjccore:1516751336-209342/1516751336-654858 with status:success
2018-01-23 23:49:13,387 - Received ns-deployment-event for autoit:1516751336-209342 with
status:success
2018-01-23 23:49:13,414 - RPC NS[autoit:autoit-instance] success
2018-01-23 23:49:13,496 - Deployment activate-ns-deployment: autoit succeeded
2018-01-23 23:49:13,540 - Send Deployment notification for: autoit-deploy
No handlers could be found for logger "AutoVNF-Traces"
2018-01-23 23:49:13,670 - All workers finished the job
2018-01-23 23:49:13,689 - Deployment activate-ns-deployment: autoit succeeded
2018-01-23 23:49:13,723 - Send Deployment notification for: autoit-instance

```

### Example AutoVNF Creation Log:

```

<config xmlns="http://tail-f.com/ns/config/1.0">
  <log xmlns="http://www.cisco.com/usp/nfv/usp-transaction">
    <tx-id>1516900912-955117</tx-id>
    <log>
      2018-01-25 17:21:54,162 - Send Deployment notification for: autoit-instance
      2018-01-25 17:21:54,195 - Deployment activate-ns-deployment: autoit started
      2018-01-25 17:21:54,225 - Adding NSR: autoit-instance
      2018-01-25 17:21:54,288 - Start pipeline of 1 tasks
      2018-01-25 17:21:54,312 - Scheduling Task: autoit
      2018-01-25 17:21:54,342 - Waiting for all workers to finish the transactions
      2018-01-25 17:23:19,325 - All workers finished the job
      2018-01-25 17:23:19,365 - Deployment activate-ns-deployment: autoit succeeded
      2018-01-25 17:23:19,517 - Send Deployment notification for: autoit-instance
      2018-01-25 17:24:28,117 - Deployment activate-ns-deployment: tb3-autovnf_vpc initiated
      2018-01-25 17:24:28,209 - Send Deployment notification for: tb3-autovnf_vpc-instance
      2018-01-25 17:21:54,505 - Send Deployment notification for: autoit-deploy
      2018-01-25 17:21:54,550 - Deployment activate-vnf-deployment: autoit started
      2018-01-25 17:21:54,588 - Adding NSR: autoit-instance, VNFR: autoit-tb3-autovnf1, vlrs:
      None
      2018-01-25 17:21:54,661 - VNF deployment pre-check success(all-not-present)
      2018-01-25 17:21:55,001 - Connected to AutoIT[10.84.123.51]
      2018-01-25 17:21:55,039 - XML:[&lt;config>
        &lt;nsd xmlns="http://www.cisco.com/usp/nfv/usp-nsds">
          &lt;nsd-id>autoit&lt;/nsd-id>
          &lt;vim-identity>vim2&lt;/vim-identity>
        .
        .
        .
        2018-01-25 17:25:04,646 - &lt;?xml version="1.0" encoding="UTF-8"?>
        &lt;rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
        message-id="urn:uuid:1d0dd00b-a3a9-4e10-9a71-376680d05dca"
        xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">&lt;transaction-id
        xmlns='http://www.cisco.com/usp/nfv/usp-nsds'>1516901142-922838&lt;/transaction-id>
        &lt;/rpc-reply>
        2018-01-25 17:25:04,736 - Waiting for deployment notifications for tx-id '1516901142-922838'
        2018-01-25 17:25:04,816 - Received ns-deployment-event for tb3-autovnf_vpc:1516901142-922838

```

```

with status:requested
2018-01-25 17:25:04,851 - Received vim-deployment-event for
tb3-vnfl-rack:1516901142-922838/1516901143-301032 with status:requested
2018-01-25 17:25:04,908 - VIM: requested:None/None
2018-01-25 17:25:04,977 - Received vnf-package-deployment-event for
usp_6_0:1516901142-922838/1516901143-337769 with status:requested
2018-01-25 17:25:05,034 - VNF-PKG[usp_6_0]: requested, activate-vnf-package
2018-01-25 17:25:05,118 - Received vnf-deployment-event for
esc:1516901142-922838/1516901143-372586 with status:requested
2018-01-25 17:25:05,166 - Received vnf-deployment-event for
vpc:1516901142-922838/1516901143-418832 with status:requested
2018-01-25 17:25:05,201 - Received ns-deployment-event for tb3-autovnf_vpc:1516901142-922838
with status:in-progress
2018-01-25 17:25:05,235 - Received vim-deployment-event for
tb3-vnfl-rack:1516901142-922838/1516901143-301032 with status:in-progress
2018-01-25 17:25:05,269 - VIM: in-progress:None/None
2018-01-25 17:25:15,753 - Received vim-deployment-event for
tb3-vnfl-rack:1516901142-922838/1516901143-301032 with status:success
2018-01-25 17:25:15,786 - VIM: success:None/None
2018-01-25 17:25:15,889 - Received vnf-package-deployment-event for
usp_6_0:1516901142-922838/1516901143-337769 with status:in-progress
2018-01-25 17:25:15,927 - VNF-PKG[usp_6_0]: in-progress, activate-vnf-package
2018-01-25 17:27:44,479 - Received vnf-package-deployment-event for
usp_6_0:1516901142-922838/1516901143-337769 with status:success
2018-01-25 17:27:44,566 - VNF-PKG[usp_6_0]: success, activate-vnf-package
2018-01-25 17:27:44,624 - Received vnf-deployment-event for
esc:1516901142-922838/1516901143-372586 with status:in-progress
2018-01-25 17:31:13,916 - Received vnf-deployment-event for
esc:1516901142-922838/1516901143-372586 with status:success
2018-01-25 17:31:13,972 - Received vnf-deployment-event for
vpc:1516901142-922838/1516901143-418832 with status:in-progress
2018-01-25 17:45:29,291 - Received vnf-deployment-event for
vpc:1516901142-922838/1516901143-418832 with status:success
2018-01-25 17:45:29,318 - Received ns-deployment-event for tb3-autovnf_vpc:1516901142-922838
with status:success
2018-01-25 17:45:29,382 - RPC NS[tb3-autovnf_vpc:tb3-autovnf_vpc-instance] success
2018-01-25 17:45:30,000 - Deployment activate-ns-deployment: tb3-autovnf_vpc succeeded
2018-01-25 17:45:30,141 - Send Deployment notification for: tb3-autovnf_vpc-deploy</log>
</log>
</config>

```

## Checking AutoDeploy Processes

Check the status of AutoDeploy VM by entering the following commands:

```
service autodeploy status
```

```
service uas-confd status
```

## Determining the Running AutoDeploy Version

To display the version of the AutoDeploy software role that is currently operational:

```
show uas
```

**Example output:**

```

uas version                6.0.0
uas state                  active
uas external-connection-point 172.28.185.132
INSTANCE IP      STATE  ROLE
-----

```

```
172.28.185.133 alive CONF-D-MASTER
172.28.185.134 alive CONF-D-SLAVE
```

```
NAME                LAST HEARTBEAT
-----
AutoDeploy-MASTER  2018-01-24 21:29:54
USPCFMWorker       2018-01-24 21:29:45
USPCHBWorker       2018-01-24 21:29:45
USPCWorker         2018-01-24 21:29:45
```

## Monitoring AutoIT Operations

This section identifies various commands that can be used to determine the status and health of AutoIT.

To use them, you must:

1. Log on to the AutoIT VM as *ubuntu*. Use the password that was created earlier for this user.
2. Become the *root* user.

```
sudo -i
```

## Viewing AutoIT Logs

AutoIT maintains logs containing information pertaining to UAS deployment and termination transactions. The *autoit.log* file is located in the following directory on the Ultra M Manager Node:

```
/var/log/cisco/usp/auto-it/autoit.log
```

### Example Deployment Log:

```
tail -100f /var/log/cisco/usp/auto-it/autoit.log &^C
```

```
2017-05-25 22:04:57,527 - INFO: Received a request to list config folder names.
2017-05-25 22:04:57,527 - INFO: config contents are:
2017-05-25 22:04:57,536 - INFO: Received a request to list config folder names.
2017-05-25 22:04:57,536 - INFO: config contents are:
2017-05-25 22:04:57,545 - INFO: Received a request to create a configuration folder.
2017-05-25 22:04:57,551 - INFO: Received a request to create a configuration folder.
2017-05-25 22:04:57,553 - INFO: Received request to download package: system.cfg from ISO
2017-05-25 22:04:57,563 - INFO: Received request to download package: system.cfg from ISO
2017-05-25 22:04:57,565 - INFO: Received request to download package: system.cfg from ISO
2017-05-25 22:04:57,566 - INFO: Received request to upload config file system.cfg to config
named vnf-pkg1
2017-05-25 22:04:57,567 - INFO: Uploaded file system.cfg to config named vnf-pkg1

2017-05-25 22:05:54,268 - INFO: Received request to upload ISO usp-5_1_0.iso
2017-05-25 22:05:54,268 - INFO: Saving ISO to /tmp/tmpxu7Mu0/usp-5_1_0.iso
2017-05-25 22:06:30,678 - INFO: Mounting ISO to /tmp/tmpxu7Mu0/iso_mount
2017-05-25 22:06:30,736 - INFO: ISO version already installed, (5.1.0-662)
2017-05-25 22:06:31,355 - INFO: Received a request to list file names in config named
vnf-pkg1.
2017-05-25 22:06:31,355 - INFO: config contents are: system.cfg
2017-05-25 22:06:31,362 - INFO: Received a request to list file names in config named
vnf-pkg1-images.
2017-05-25 22:06:31,362 - INFO: config contents are:
2017-05-25 22:06:31,370 - INFO: Received request to get ISO details 5.1.0-662
2017-05-25 22:06:31,391 - INFO: Received a request to get an Host Aggregate details
2017-05-25 22:06:31,857 - INFO: Getting Host Aggregate failed: Aggregate
'auto-test-sjc-service1' not found on OpenStack setup
```

```

2017-05-25 22:06:31,872 - INFO: Received a request to deploy an Host Aggregate
2017-05-25 22:06:32,415 - INFO: Deploying Host Aggregate 'auto-test-sjc-service1' completed
2017-05-25 22:06:32,427 - INFO: Received a request to get an Host Aggregate details
2017-05-25 22:06:32,975 - INFO: Getting Host Aggregate failed: Aggregate
'auto-test-sjc-cf-esc-mgmt1' not found on OpenStack setup
2017-05-25 22:06:32,986 - INFO: Received a request to deploy an Host Aggregate
2017-05-25 22:06:33,513 - INFO: Deploying Host Aggregate 'auto-test-sjc-cf-esc-mgmt1'
completed
2017-05-25 22:06:33,524 - INFO: Received a request to get an Host Aggregate details
2017-05-25 22:06:33,881 - INFO: Getting Host Aggregate failed: Aggregate
'auto-test-sjc-em-autovnf-mgmt1' not found on OpenStack setup
2017-05-25 22:06:33,891 - INFO: Received a request to deploy an Host Aggregate
2017-05-25 22:06:34,535 - INFO: Deploying Host Aggregate 'auto-test-sjc-em-autovnf-mgmt1'
completed
2017-05-25 22:06:34,580 - INFO: Received a request to deploy AutoVnf
2017-05-25 22:06:40,340 - INFO: Creating AutoVnf deployment (3 instance(s)) on
'http://172.21.201.217:5000/v2.0' tenant 'core' user 'core', ISO '5.1.0-662'
2017-05-25 22:06:40,340 - INFO: Creating network 'auto-testautovnf1-uas-management'
2017-05-25 22:06:42,241 - INFO: Created network 'auto-testautovnf1-uas-management'
2017-05-25 22:06:42,241 - INFO: Creating network 'auto-testautovnf1-uas-orchestration'
2017-05-25 22:06:42,821 - INFO: Created network 'auto-testautovnf1-uas-orchestration'
2017-05-25 22:06:42,888 - INFO: Created flavor 'auto-testautovnf1-uas'
2017-05-25 22:06:42,888 - INFO: Loading image 'auto-testautovnf1-usp-uas-1.0.0-601.qcow2'
from '/opt/cisco/usp/bundles/5.1.0-662/uas-bundle/usp-uas-1.0.0-601.qcow2'
2017-05-25 22:06:53,927 - INFO: Loaded image 'auto-testautovnf1-usp-uas-1.0.0-601.qcow2'
2017-05-25 22:06:53,928 - INFO: Creating volume 'auto-testautovnf1-uas-vol-0' with command

[/opt/cisco/usp/apps/auto-it/vnf/./common/autoit/./autoit_os_utils/scripts/autoit_volume_staging.sh
OS_USERNAME core OS_TENANT_NAME core OS_PASSWORD **** OS_AUTH_URL
http://172.21.201.217:5000/v2.0 ARG_TENANT core ARG_DEPLOYMENT test-uas ARG_VM_NAME
auto-testautovnf1-uas-vol-0 ARG_VOLUME_TYPE LUKS FILE_1 /tmp/tmpHsTAj6/encrypted.cfg]
2017-05-25 22:07:06,104 - INFO: Created volume 'auto-testautovnf1-uas-vol-0'
2017-05-25 22:07:06,104 - INFO: Creating volume 'auto-testautovnf1-uas-vol-1' with command

[/opt/cisco/usp/apps/auto-it/vnf/./common/autoit/./autoit_os_utils/scripts/autoit_volume_staging.sh
OS_USERNAME core OS_TENANT_NAME core OS_PASSWORD **** OS_AUTH_URL
http://172.21.201.217:5000/v2.0 ARG_TENANT core ARG_DEPLOYMENT test-uas ARG_VM_NAME
auto-testautovnf1-uas-vol-1 ARG_VOLUME_TYPE LUKS FILE_1 /tmp/tmpHsTAj6/encrypted.cfg]
2017-05-25 22:07:17,598 - INFO: Created volume 'auto-testautovnf1-uas-vol-1'
2017-05-25 22:07:17,598 - INFO: Creating volume 'auto-testautovnf1-uas-vol-2' with command

[/opt/cisco/usp/apps/auto-it/vnf/./common/autoit/./autoit_os_utils/scripts/autoit_volume_staging.sh
OS_USERNAME core OS_TENANT_NAME core OS_PASSWORD **** OS_AUTH_URL
http://172.21.201.217:5000/v2.0 ARG_TENANT core ARG_DEPLOYMENT test-uas ARG_VM_NAME
auto-testautovnf1-uas-vol-2 ARG_VOLUME_TYPE LUKS FILE_1 /tmp/tmpHsTAj6/encrypted.cfg]
2017-05-25 22:07:29,242 - INFO: Created volume 'auto-testautovnf1-uas-vol-2'
2017-05-25 22:07:30,477 - INFO: Assigned floating IP '172.21.201.59' to IP '172.57.11.101'
2017-05-25 22:07:33,843 - INFO: Creating instance 'auto-testautovnf1-uas-0' and attaching
volume 'auto-testautovnf1-uas-vol-0'
2017-05-25 22:08:00,717 - INFO: Created instance 'auto-testautovnf1-uas-0'
2017-05-25 22:08:00,717 - INFO: Creating instance 'auto-testautovnf1-uas-1' and attaching
volume 'auto-testautovnf1-uas-vol-1'
2017-05-25 22:08:27,577 - INFO: Created instance 'auto-testautovnf1-uas-1'
2017-05-25 22:08:27,578 - INFO: Creating instance 'auto-testautovnf1-uas-2' and attaching
volume 'auto-testautovnf1-uas-vol-2'
2017-05-25 22:08:58,345 - INFO: Created instance 'auto-testautovnf1-uas-2'
2017-05-25 22:08:58,345 - INFO: Deploy request completed
2017-05-25 22:14:07,201 - INFO: Received request to download file system.cfg from config
named vnf-pkg1
2017-05-25 22:19:05,050 - INFO: Received a request to list config folder names.
2017-05-25 22:19:05,051 - INFO: config contents are: vnf-pkg1-images, vnf-pkg1
2017-05-25 22:19:05,059 - INFO: Received a request to list config folder names.
2017-05-25 22:19:05,059 - INFO: config contents are: vnf-pkg1-images, vnf-pkg1
2017-05-25 22:19:05,066 - INFO: Received a request to create a configuration folder.

```

```

2017-05-25 22:19:05,073 - INFO: Received a request to create a configuration folder.
2017-05-25 22:19:05,076 - INFO: Received request to download package: system.cfg from ISO
2017-05-25 22:19:05,083 - INFO: Received request to download package: system.cfg from ISO
2017-05-25 22:19:05,085 - INFO: Received request to download package: system.cfg from ISO
2017-05-25 22:19:05,086 - INFO: Received request to upload config file system.cfg to config
named vnf-pkg2
2017-05-25 22:19:05,087 - INFO: Uploaded file system.cfg to config named vnf-pkg2
2017-05-25 22:19:59,895 - INFO: Received request to upload ISO usp-5_1_0.iso
2017-05-25 22:19:59,895 - INFO: Saving ISO to /tmp/tmpWbdnmxm/usp-5_1_0.iso
2017-05-25 22:20:21,395 - INFO: Mounting ISO to /tmp/tmpWbdnmxm/iso_mount
2017-05-25 22:20:22,288 - INFO: ISO version already installed, (5.1.0-662)
2017-05-25 22:20:23,203 - INFO: Received a request to list file names in config named
vnf-pkg2.
2017-05-25 22:20:23,203 - INFO: config contents are: system.cfg
2017-05-25 22:20:23,211 - INFO: Received a request to list file names in config named
vnf-pkg2-images.
2017-05-25 22:20:23,211 - INFO: config contents are:
2017-05-25 22:20:23,220 - INFO: Received request to get ISO details 5.1.0-662
2017-05-25 22:20:23,251 - INFO: Received a request to get an Host Aggregate details
2017-05-25 22:20:23,621 - INFO: Getting Host Aggregate failed: Aggregate
'auto-test-sjc-em-autovnf-mgmt2' not found on OpenStack setup
2017-05-25 22:20:23,633 - INFO: Received a request to deploy an Host Aggregate
2017-05-25 22:20:24,301 - INFO: Deploying Host Aggregate 'auto-test-sjc-em-autovnf-mgmt2'
completed
2017-05-25 22:20:24,313 - INFO: Received a request to get an Host Aggregate details
2017-05-25 22:20:24,843 - INFO: Getting Host Aggregate failed: Aggregate
'auto-test-sjc-service2' not found on OpenStack setup
2017-05-25 22:20:24,853 - INFO: Received a request to deploy an Host Aggregate
2017-05-25 22:20:25,524 - INFO: Deploying Host Aggregate 'auto-test-sjc-service2' completed
2017-05-25 22:20:25,537 - INFO: Received a request to get an Host Aggregate details
2017-05-25 22:20:25,898 - INFO: Getting Host Aggregate failed: Aggregate
'auto-test-sjc-cf-esc-mgmt2' not found on OpenStack setup
2017-05-25 22:20:25,909 - INFO: Received a request to deploy an Host Aggregate
2017-05-25 22:20:26,540 - INFO: Deploying Host Aggregate 'auto-test-sjc-cf-esc-mgmt2'
completed
2017-05-25 22:20:26,584 - INFO: Received a request to deploy AutoVnf
2017-05-25 22:20:31,604 - INFO: Creating AutoVnf deployment (3 instance(s)) on
'http://172.21.201.217:5000/v2.0' tenant 'core' user 'core', ISO '5.1.0-662'
2017-05-25 22:20:31,605 - INFO: Creating network 'auto-testautovnf2-uas-management'
2017-05-25 22:20:33,720 - INFO: Created network 'auto-testautovnf2-uas-management'
2017-05-25 22:20:33,720 - INFO: Creating network 'auto-testautovnf2-uas-orchestration'
2017-05-25 22:20:34,324 - INFO: Created network 'auto-testautovnf2-uas-orchestration'
2017-05-25 22:20:34,402 - INFO: Created flavor 'auto-testautovnf2-uas'
2017-05-25 22:20:34,402 - INFO: Loading image 'auto-testautovnf2-usp-uas-1.0.0-601.qcow2'
from '/opt/cisco/usp/bundles/5.1.0-662/uas-bundle/usp-uas-1.0.0-601.qcow2'
2017-05-25 22:20:43,169 - INFO: Loaded image 'auto-testautovnf2-usp-uas-1.0.0-601.qcow2'
2017-05-25 22:20:43,169 - INFO: Creating volume 'auto-testautovnf2-uas-vol-0' with command

[/opt/cisco/usp/apps/auto-it/vnf/./common/autoit/./autoit_os_utils/scripts/autoit_volume_staging.sh
OS_USERNAME core OS_TENANT_NAME core OS_PASSWORD **** OS_AUTH_URL
http://172.21.201.217:5000/v2.0 ARG_TENANT core ARG_DEPLOYMENT test-uas ARG_VM_NAME
auto-testautovnf2-uas-vol-0 ARG_VOLUME_TYPE LUKS FILE_1 /tmp/tmpelmmIL/encrypted.cfg]
2017-05-25 22:20:54,713 - INFO: Created volume 'auto-testautovnf2-uas-vol-0'
2017-05-25 22:20:54,714 - INFO: Creating volume 'auto-testautovnf2-uas-vol-1' with command

[/opt/cisco/usp/apps/auto-it/vnf/./common/autoit/./autoit_os_utils/scripts/autoit_volume_staging.sh
OS_USERNAME core OS_TENANT_NAME core OS_PASSWORD **** OS_AUTH_URL
http://172.21.201.217:5000/v2.0 ARG_TENANT core ARG_DEPLOYMENT test-uas ARG_VM_NAME
auto-testautovnf2-uas-vol-1 ARG_VOLUME_TYPE LUKS FILE_1 /tmp/tmpelmmIL/encrypted.cfg]
2017-05-25 22:21:06,203 - INFO: Created volume 'auto-testautovnf2-uas-vol-1'
2017-05-25 22:21:06,204 - INFO: Creating volume 'auto-testautovnf2-uas-vol-2' with command

[/opt/cisco/usp/apps/auto-it/vnf/./common/autoit/./autoit_os_utils/scripts/autoit_volume_staging.sh
OS_USERNAME core OS_TENANT_NAME core OS_PASSWORD **** OS_AUTH_URL

```

```

http://172.21.201.217:5000/v2.0 ARG_TENANT core ARG_DEPLOYMENT test-uas ARG_VM_NAME
auto-testautovnf2-uas-vol-2 ARG_VOLUME_TYPE LUKS FILE_1 /tmp/tmpelMIL/encrypted.cfg]
2017-05-25 22:21:18,184 - INFO: Created volume 'auto-testautovnf2-uas-vol-2'
2017-05-25 22:21:19,626 - INFO: Assigned floating IP '172.21.201.64' to IP '172.67.11.101'
2017-05-25 22:21:22,762 - INFO: Creating instance 'auto-testautovnf2-uas-0' and attaching
volume 'auto-testautovnf2-uas-vol-0'
2017-05-25 22:21:49,741 - INFO: Created instance 'auto-testautovnf2-uas-0'
2017-05-25 22:21:49,742 - INFO: Creating instance 'auto-testautovnf2-uas-1' and attaching
volume 'auto-testautovnf2-uas-vol-1'
2017-05-25 22:22:16,881 - INFO: Created instance 'auto-testautovnf2-uas-1'
2017-05-25 22:22:16,881 - INFO: Creating instance 'auto-testautovnf2-uas-2' and attaching
volume 'auto-testautovnf2-uas-vol-2'
2017-05-25 22:22:43,304 - INFO: Created instance 'auto-testautovnf2-uas-2'
2017-05-25 22:22:43,304 - INFO: Deploy request completed
2017-05-25 22:28:08,865 - INFO: Received request to download file system.cfg from config
named vnf-pkg2
2017-05-25 22:40:03,550 - INFO: Received request to download file system.cfg from config
named vnf-pkg1

```

### Example Termination Log:

```

2017-05-25 22:53:30,970 - INFO: Received a request to destroy AutoVnf
2017-05-25 22:53:31,310 - INFO: Destroying AutoVnf deployment on
'http://172.21.201.217:5000/v2.0' tenant 'core' user 'core', ISO '5.1.0-662'
2017-05-25 22:53:32,698 - INFO: Removed floating IP '172.21.201.64'
2017-05-25 22:53:34,114 - INFO: 3 instance(s) found with name matching 'auto-testautovnf2'
2017-05-25 22:53:34,448 - INFO: Removing volume 'auto-testautovnf2-uas-vol-2'
2017-05-25 22:53:43,481 - INFO: Removed volume 'auto-testautovnf2-uas-vol-2'
2017-05-25 22:53:43,481 - INFO: Removing instance 'auto-testautovnf2-uas-2'
2017-05-25 22:53:47,080 - INFO: Removed instance 'auto-testautovnf2-uas-2'
2017-05-25 22:53:47,283 - INFO: Removing volume 'auto-testautovnf2-uas-vol-1'
2017-05-25 22:53:56,508 - INFO: Removed volume 'auto-testautovnf2-uas-vol-1'
2017-05-25 22:53:56,508 - INFO: Removing instance 'auto-testautovnf2-uas-1'
2017-05-25 22:54:00,290 - INFO: Removed instance 'auto-testautovnf2-uas-1'
2017-05-25 22:54:00,494 - INFO: Removing volume 'auto-testautovnf2-uas-vol-0'
2017-05-25 22:54:04,714 - INFO: Removed volume 'auto-testautovnf2-uas-vol-0'
2017-05-25 22:54:04,714 - INFO: Removing instance 'auto-testautovnf2-uas-0'
2017-05-25 22:54:11,647 - INFO: Removed instance 'auto-testautovnf2-uas-0'
2017-05-25 22:54:15,107 - INFO: 1 image(s) 'auto-testautovnf2-usp-uas-1.0.0-601.qcow2'
found, removing
2017-05-25 22:54:19,289 - INFO: Removed network 'auto-testautovnf2-uas-management'
2017-05-25 22:54:20,463 - INFO: Removed network 'auto-testautovnf2-uas-orchestration'
2017-05-25 22:54:20,541 - INFO: Removed flavor 'auto-testautovnf2-uas'
2017-05-25 22:54:20,541 - INFO: Destroy request completed
2017-05-25 22:54:20,562 - INFO: Received a request to get an Host Aggregate details
2017-05-25 22:54:20,925 - INFO: Getting Host Aggregate 'auto-test-sjc-em-autovnf-mgmt2'
completed
2017-05-25 22:54:20,940 - INFO: Received a request to destroy an Host Aggregate
2017-05-25 22:54:21,564 - INFO: Destroying Host Aggregate 'auto-test-sjc-em-autovnf-mgmt2'
completed
2017-05-25 22:54:21,575 - INFO: Received a request to get an Host Aggregate details
2017-05-25 22:54:21,930 - INFO: Getting Host Aggregate 'auto-test-sjc-service2' completed
2017-05-25 22:54:21,947 - INFO: Received a request to destroy an Host Aggregate
2017-05-25 22:54:22,456 - INFO: Destroying Host Aggregate 'auto-test-sjc-service2' completed
2017-05-25 22:54:22,468 - INFO: Received a request to get an Host Aggregate details
2017-05-25 22:54:22,826 - INFO: Getting Host Aggregate 'auto-test-sjc-cf-esc-mgmt2' completed
2017-05-25 22:54:22,840 - INFO: Received a request to destroy an Host Aggregate
2017-05-25 22:54:23,394 - INFO: Destroying Host Aggregate 'auto-test-sjc-cf-esc-mgmt2'
completed
2017-05-25 22:56:55,925 - INFO: Received a request to destroy AutoVnf
2017-05-25 22:56:56,391 - INFO: Destroying AutoVnf deployment on
'http://172.21.201.217:5000/v2.0' tenant 'core' user 'core', ISO '5.1.0-662'
2017-05-25 22:56:57,507 - INFO: Removed floating IP '172.21.201.59'
2017-05-25 22:56:58,614 - INFO: 3 instance(s) found with name matching 'auto-testautovnf1'
2017-05-25 22:56:58,949 - INFO: Removing volume 'auto-testautovnf1-uas-vol-2'

```

```

2017-05-25 22:57:08,166 - INFO: Removed volume 'auto-testautovnf1-uas-vol-2'
2017-05-25 22:57:08,166 - INFO: Removing instance 'auto-testautovnf1-uas-2'
2017-05-25 22:57:15,117 - INFO: Removed instance 'auto-testautovnf1-uas-2'
2017-05-25 22:57:15,323 - INFO: Removing volume 'auto-testautovnf1-uas-vol-1'
2017-05-25 22:57:24,501 - INFO: Removed volume 'auto-testautovnf1-uas-vol-1'
2017-05-25 22:57:24,502 - INFO: Removing instance 'auto-testautovnf1-uas-1'
2017-05-25 22:57:28,275 - INFO: Removed instance 'auto-testautovnf1-uas-1'
2017-05-25 22:57:28,722 - INFO: Removing volume 'auto-testautovnf1-uas-vol-0'
2017-05-25 22:57:37,702 - INFO: Removed volume 'auto-testautovnf1-uas-vol-0'
2017-05-25 22:57:37,703 - INFO: Removing instance 'auto-testautovnf1-uas-0'
2017-05-25 22:57:44,622 - INFO: Removed instance 'auto-testautovnf1-uas-0'
2017-05-25 22:57:47,921 - INFO: 1 image(s) 'auto-testautovnf1-usp-uas-1.0.0-601.qcow2'
found, removing
2017-05-25 22:57:52,453 - INFO: Removed network 'auto-testautovnf1-uas-management'
2017-05-25 22:57:53,677 - INFO: Removed network 'auto-testautovnf1-uas-orchestration'
2017-05-25 22:57:53,760 - INFO: Removed flavor 'auto-testautovnf1-uas'
2017-05-25 22:57:53,760 - INFO: Destroy request completed

```

## Viewing AutoIT Operational Data

View the AutoIT operational data by executing the following command:

```
show uas
```

### Example show uas Command Output

```

uas version          6.0.0
uas state            active
uas external-connection-point 172.28.185.132
INSTANCE IP        STATE   ROLE
-----
172.28.185.133    alive  CONFD-MASTER
172.28.185.134    alive  CONFD-SLAVE

NAME                LAST HEARTBEAT
-----
AutoIT-MASTER      2018-01-24 21:24:30
USPCFMWorker        2018-01-24 21:24:30
USPCHBWorker        2018-01-24 21:24:30
USPCWorker          2018-01-24 21:24:30

```



### Important

In case of standalone mode (non-HA) deployments, the *uas external-connection-point* information and *Instance IP* table are not applicable and are not displayed.

## Checking AutoIT Processes

Verify that key processes are running on the AutoIT VM:

With Ubuntu 14.04:

```
service autoit status
```

### Example output:

```
AutoIT is running.
```

Check ConfD.

```
service uas-confd status
```

With Ubuntu 16.04:

```
* autoit.service - Job that runs the autoit daemon
Loaded: loaded (/etc/systemd/system/autoit.service; static; vendor preset: enabled)
Active: active (running) since Fri 2018-09-21 22:11:54 UTC; 1 weeks 0 days ago
Main PID: 1320 (autoit.sh)
CGroup: /system.slice/autoit.service
|-1320 /bin/sh /etc/cisco/autoit.sh start
|-1337 /bin/sh /etc/cisco/autoit.sh start
|-1338 /opt/cisco/usp/uas/autoit/autoit
|-1339 tee -a /var/log/upstart/autoit.log
|-1341 /opt/cisco/usp/uas/autoit/autoit
|-1346 /opt/cisco/usp/uas/autoit/autoit
|-1347 /opt/cisco/usp/uas/autoit/autoit
|-1348 /opt/cisco/usp/uas/autoit/autoit
|-1349 /opt/cisco/usp/uas/autoit/autoit
|-1350 /opt/cisco/usp/uas/autoit/autoit
|-1352 /opt/cisco/usp/uas/autoit/autoit
|-1353 /opt/cisco/usp/uas/autoit/autoit
```

## Monitoring AutoVNF Operations

This section identifies various commands that can be used to determine the status and health of AutoVNF.

To use them, you must:

1. Log on to the AutoVNF VM as *ubuntu*. Use the password that was created earlier for this user.
2. Become the root user.

```
sudo -i
```

## Viewing AutoVNF Logs

### General AutoVNF Logs

AutoVNF logs are available on the AutoVNF VM in the following file:

```
/var/log/upstart/autovnf.log
```

To collect AutoVNF logs:

1. Navigate to the *scripts* directory.

```
cd /opt/cisco/usp/uas/scripts
```

2. Launch the *collect-uas-logs.sh* script to collect the logs.

```
sudo ./collect-uas-logs.sh
```

#### Example log output:

```
Creating log tarball uas-logs-2017-05-26_00.24.55_UTC.tar.bz2 ...
uas-logs/
uas-logs/autovnf/
uas-logs/autovnf/autovnf_server.log
uas-logs/autovnf/a15bf26c-41a1-11e7-b3ab-fa163eccaffc/
uas-logs/autovnf/a15bf26c-41a1-11e7-b3ab-fa163eccaffc/netconf_traces
uas-logs/autovnf/a15bf26c-41a1-11e7-b3ab-fa163eccaffc/vnfd
```

```

uas-logs/autovnf/audit.log
uas-logs/autovnf/579b4546-41a2-11e7-b3ab-fa163eccaffc/
uas-logs/autovnf/579b4546-41a2-11e7-b3ab-fa163eccaffc/netconf_traces
uas-logs/autovnf/579b4546-41a2-11e7-b3ab-fa163eccaffc/vnfd
uas-logs/ha/
uas-logs/ha/info.log
uas-logs/uas_manager/
uas-logs/uas_manager/info.log
uas-logs/zk/
uas-logs/zk/zookeeper.out
uas-logs/zk/zookeeper.log
uas-logs/upstart/
uas-logs/upstart/uas-confd.log
uas-logs/upstart/zk.log
uas-logs/upstart/autovnf.log
uas-logs/upstart/uws-ae.log
uas-logs/upstart/ensemble.log

===== Tarball available at: /tmp/uas-logs-2017-05-26_00.24.55.UTC.tar.bz2
=====

To extract the tarball, run: "tar jxf /tmp/uas-logs-2017-05-26_00.24.55.UTC.tar.bz2"

```

## AutoVNF Transaction Logs

AutoVNF server and transaction logs are available on the Ultra M Manager Node in the following directory on the UAS VM:

```
/var/log/cisco-uas/autovnf
```

Inside this directory are transaction sub-directories, VNFD information and NETCONF traces are provided for the given transaction.

### Example:

```

total 3568
drwxr-xr-x 4 root root    4096 May 25 23:31 ./
drwxr-xr-x 7 root root    4096 May 25 19:39 ../
drwxr-xr-x 2 root root    4096 May 25 23:31 579b4546-41a2-11e7-b3ab-fa163eccaffc/
drwxr-xr-x 2 root root    4096 May 25 23:29 a15bf26c-41a1-11e7-b3ab-fa163eccaffc/
-rw-r--r-- 1 root root 3632813 May 26 18:33 audit.log
-rw-r--r-- 1 root root      0 May 25 23:26 autovnf_server.log

cd a15bf26c-41a1-11e7-b3ab-fa163eccaffc
total 2568
drwxr-xr-x 2 root root    4096 May 25 23:29 ./
drwxr-xr-x 4 root root    4096 May 25 23:31 ../
-rw-r--r-- 1 root root 2614547 May 25 23:37 netconf_traces
-rw-r--r-- 1 root root      0 May 25 23:29 vnfd

```

## AutoVNF Event Logs

Event logs provide useful information on UAS task progress. These logs are located in the *autovnf.log* file within the following directory on the UAS VM:

```
/var/log/upstart
```

Event logs are filed by transaction ID. To view transaction IDs:

1. Log in to the ConfD CLI as the *admin* user.

```
confd_cli -u admin -C
```

2. Enter the *admin* user password when prompted.
3. List the transactions.

### show transactions

#### Example output:

TX ID	STATUS	TX TYPE	DEPLOYMENT ID	TIMESTAMP
562c18b0-4199-11e7-ad05-fa163ec6a7e4		vnf-deployment	vnfd2-deployment	
2017-05-25T22:27:28.962293-00:00		deployment-success		
abf51428-4198-11e7-ad05-fa163ec6a7e4		vnfm-deployment	ab-auto-test-vnfm2	
2017-05-25T22:22:43.389059-00:00		deployment-success		

To view the logs associated with a specific transaction:

### show log <transaction\_id> | display xml

#### Example log pertaining to VNF deployment:

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <logs xmlns="http://www.cisco.com/usp/nfv/usp-autovnf-oper">
    <tx-id>abf51428-4198-11e7-ad05-fa163ec6a7e4</tx-id>
    <log>2017-05-25 22:22:43,402 - VNF Deployment RPC triggered for deployment:
ab-auto-test-vnfm2, deactivate: 0
2017-05-25 22:22:43,446 - Notify deployment
2017-05-25 22:22:43,472 - VNF Transaction: abf51428-4198-11e7-ad05-fa163ec6a7e4 for
deployment: ab-auto-test-vnfm2 started
2017-05-25 22:22:43,497 - Downloading Image:
http://172.21.201.63:80/bundles/5.1.0-662/vnfm-bundle/ESC-2_3_2_143.qcow2
2017-05-25 22:22:49,146 - Image: //opt/cisco/vnf-staging/vnfm_image downloaded
successfully
2017-05-25 22:22:49,714 - Checking network 'public' existence
2017-05-25 22:22:49,879 - Checking flavor 'ab-auto-test-vnfm2-ESC-flavor' non existence
2017-05-25 22:22:50,124 - Checking image 'ab-auto-test-vnfm2-ESC-image' non existence
2017-05-25 22:22:50,598 - Checking network 'auto-testautovnf2-uas-management' existence
2017-05-25 22:22:50,752 - Checking network 'auto-testautovnf2-uas-orchestration' existence
2017-05-25 22:22:50,916 - Checking instance 'ab-auto-test-vnfm2-ESC-0' non existence
2017-05-25 22:22:51,357 - Checking instance 'ab-auto-test-vnfm2-ESC-1' non existence
2017-05-25 22:22:52,084 - Creating flavor 'ab-auto-test-vnfm2-ESC-flavor'
2017-05-25 22:22:52,184 - Loading image 'ab-auto-test-vnfm2-ESC-image' from
'//opt/cisco/vnf-staging/vnfm_image'...
2017-05-25 22:23:06,444 - ESC HA mode is ON
2017-05-25 22:23:07,118 - Allocated these IPs for ESC HA: ['172.67.11.3', '172.67.11.4',
'172.67.11.5']
2017-05-25 22:23:08,228 - Creating VNF 'ab-auto-test-vnfm2-ESC-0' with [python
//opt/cisco/vnf-staging/bootvm.py ab-auto-test-vnfm2-ESC-0 --flavor
ab-auto-test-vnfm2-ESC-flavor --image b29e7a72-9ad0-4178-aa35-35df0a2b23b7 --net
auto-testautovnf2-uas-management --gateway_ip 172.67.11.1 --net
auto-testautovnf2-uas-orchestration
--os_auth_url http://172.21.201.217:5000/v2.0 --os_tenant_name core --os_username *****
--os_password ***** --bs_os_auth_url http://172.21.201.217:5000/v2.0 --bs_os_tenant_name
core --bs_os_username ***** --bs_os_password ***** --esc_ui_startup false
--esc_params_file /tmp/esc_params.cfg --encrypt_key ***** --user_pass *****
--user_confid_pass ***** --kad_vif eth0 --kad_vip 172.67.11.5 --ipaddr 172.67.11.3 dhcp
--ha_node_list 172.67.11.3 172.67.11.4 --file
root:0755:/opt/cisco/esc/esc-scripts/esc_volume_em_staging.sh:
/opt/cisco/usp/uas/autovnf/vnfms/esc-scripts/esc_volume_em_staging.sh
--file
root:0755:/opt/cisco/esc/esc-scripts/esc_vpc_chassis_id.py:/opt/cisco/usp/uas/autovnf/vnfms/esc-scripts/esc_vpc_chassis_id.py
--file
root:0755:/opt/cisco/esc/esc-scripts/esc_vpc-di-internal-keys.sh:/opt/cisco/usp/uas/autovnf/vnfms/esc-scripts/esc_vpc-di-internal-keys.sh)...
2017-05-25 22:24:13,329 - ESC started!
```

```

2017-05-25 22:24:13,803 - Creating VNF 'ab-auto-test-vnfm2-ESC-1' with [python
//opt/cisco/vnf-staging/bootvm.py ab-auto-test-vnfm2-ESC-1 --flavor
ab-auto-test-vnfm2-ESC-flavor --image b29e7a72-9ad0-4178-aa35-35df0a2b23b7 --net
auto-testautovnf2-uas-management --gateway_ip 172.67.11.1 --net
auto-testautovnf2-uas-orchestration
--os_auth_url http://172.21.201.217:5000/v2.0 --os_tenant_name core --os_username *****
--os_password ***** --bs_os_auth_url http://172.21.201.217:5000/v2.0 --bs_os_tenant_name
core --bs_os_username ***** --bs_os_password ***** --esc_ui_startup false
--esc_params_file /tmp/esc_params.cfg --encrypt_key ***** --user_pass *****
--user_confd_pass ***** --kad_vif eth0 --kad_vip 172.67.11.5 --ipaddr 172.67.11.4 dhcp
--ha_node_list 172.67.11.3 172.67.11.4
--file root:0755:/opt/cisco/esc/esc-scripts/esc_volume_em_staging.sh:
/opt/cisco/usp/uas/autovnf/vnfms/esc-scripts/esc_volume_em_staging.sh --file
root:0755:/opt/cisco/esc/esc-scripts/esc_vpc_chassis_id.py:/opt/cisco/usp/uas/autovnf/vnfms/esc-scripts/esc_vpc_chassis_id.py
--file
root:0755:/opt/cisco/esc/esc-scripts/esc_vpc-di-internal-keys.sh:/opt/cisco/usp/uas/autovnf/vnfms/esc-scripts/esc_vpc-di-internal-keys.sh]...
2017-05-25 22:25:12,660 - ESC started!
2017-05-25 22:25:12,677 - Waiting for VIM to declare 2 instance(s) active
2017-05-25 22:25:18,254 - Instance(s) are active
2017-05-25 22:25:18,271 - Waiting for VNF to be ready...
2017-05-25 22:25:18,292 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:25:21,313 - Could not establish NETCONF session to 172.67.11.5
2017-05-25 22:25:31,341 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:25:31,362 - Could not establish NETCONF session to 172.67.11.5
2017-05-25 22:25:41,379 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:25:41,397 - Could not establish NETCONF session to 172.67.11.5
2017-05-25 22:25:51,424 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:25:51,495 - Could not establish NETCONF session to 172.67.11.5
2017-05-25 22:26:01,521 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:26:01,539 - Could not establish NETCONF session to 172.67.11.5
2017-05-25 22:26:11,563 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:26:11,591 - Could not establish NETCONF session to 172.67.11.5
2017-05-25 22:26:21,617 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:26:21,635 - Could not establish NETCONF session to 172.67.11.5
2017-05-25 22:26:31,662 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:26:31,680 - Could not establish NETCONF session to 172.67.11.5
2017-05-25 22:26:41,706 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:26:41,726 - Could not establish NETCONF session to 172.67.11.5
2017-05-25 22:26:51,748 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:26:51,765 - Could not establish NETCONF session to 172.67.11.5
2017-05-25 22:27:01,791 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:27:02,204 - NETCONF Sessions (Transaction/Notifications) established
2017-05-25 22:27:02,507 - Notify VNF Up
2017-05-25 22:27:02,525 - VNF Transaction: abf51428-4198-11e7-ad05-fa163ec6a7e4 for
deployment: ab-auto-test-vnfm2 completed successfully.
2017-05-25 22:27:02,545 - Notify deployment</log>
</logs>
</config>

```

### Example log pertaining to VNF deployment:

```

<config xmlns="http://tail-f.com/ns/config/1.0">
  <logs xmlns="http://www.cisco.com/usp/nfv/usp-autovnf-oper">
    <tx-id>562c18b0-4199-11e7-ad05-fa163ec6a7e4</tx-id>
    <log>2017-05-25 22:27:29,039 - Notify deployment
2017-05-25 22:27:29,062 - Connection to VNF (esc) at 172.67.11.5
2017-05-25 22:27:29,404 - NETCONF Sessions (Transaction/Notifications) established
2017-05-25 22:27:29,420 - Get Images
2017-05-25 22:27:29,435 - NETCONF get-config Request sent, waiting for reply
2017-05-25 22:27:29,560 - NETCONF Transaction success!
2017-05-25 22:27:29,570 - Get Flavors List
2017-05-25 22:27:29,582 - Adding images ...
2017-05-25 22:27:29,592 - Creating Images
2017-05-25 22:27:29,603 - image: ab-auto-test-vnfm2-element-manager
2017-05-25 22:27:29,620 - src:

```

```
http://172.21.201.63:80/bundles/5.1.0-662/em-bundle/em-1_0_0_532.qcow2
2017-05-25 22:27:29,630 - disk_format: qcow2
2017-05-25 22:27:29,641 - container_format: bare
2017-05-25 22:27:29,655 - serial_console: True
2017-05-25 22:27:29,665 - disk_bus: virtio
2017-05-25 22:27:29,674 - NETCONF edit-config Request sent, waiting for reply
2017-05-25 22:27:29,901 - NETCONF Transaction success!
2017-05-25 22:27:29,911 - Waiting for VNF to process CREATE_IMAGE transaction
2017-05-25 22:27:46,987 - | CREATE_IMAGE | ab-auto-test-vnfm2-element-manager | SUCCESS
| (1/1)
2017-05-25 22:27:47,004 - NETCONF transaction completed successfully!
2017-05-25 22:27:47,749 - Creating Images
2017-05-25 22:27:47,764 - image: ab-auto-test-vnfm2-control-function
2017-05-25 22:27:47,776 - src:
http://172.21.201.63:80/bundles/5.1.0-662/ugp-bundle/qvpc-di-cf.qcow2
2017-05-25 22:27:47,793 - disk_format: qcow2
2017-05-25 22:27:47,805 - container_format: bare
2017-05-25 22:27:47,819 - serial_console: True
2017-05-25 22:27:47,831 - disk_bus: virtio
2017-05-25 22:27:47,841 - NETCONF edit-config Request sent, waiting for reply
2017-05-25 22:27:48,317 - NETCONF Transaction success!
2017-05-25 22:27:48,331 - Waiting for VNF to process CREATE_IMAGE transaction
2017-05-25 22:27:56,403 - | CREATE_IMAGE | ab-auto-test-vnfm2-control-function | SUCCESS
| (1/1)
2017-05-25 22:27:56,434 - NETCONF transaction completed successfully!
2017-05-25 22:27:56,822 - Creating Images
2017-05-25 22:27:56,838 - image: ab-auto-test-vnfm2-session-function
2017-05-25 22:27:57,267 - src:
http://172.21.201.63:80/bundles/5.1.0-662/ugp-bundle/qvpc-di-sf.qcow2
2017-05-25 22:27:57,412 - disk_format: qcow2
2017-05-25 22:27:57,423 - container_format: bare
2017-05-25 22:27:57,523 - serial_console: True
2017-05-25 22:27:57,535 - disk_bus: virtio
2017-05-25 22:27:57,550 - NETCONF edit-config Request sent, waiting for reply
2017-05-25 22:27:58,378 - NETCONF Transaction success!
2017-05-25 22:27:58,391 - Waiting for VNF to process CREATE_IMAGE transaction
2017-05-25 22:28:06,339 - | CREATE_IMAGE | ab-auto-test-vnfm2-session-function | SUCCESS
| (1/1)
2017-05-25 22:28:06,355 - NETCONF transaction completed successfully!
2017-05-25 22:28:06,367 - Images added successfully
2017-05-25 22:28:06,378 - Creating flavors ...
2017-05-25 22:28:06,388 - Creating flavors
2017-05-25 22:28:06,432 - flavor: ab-auto-test-vnfm2-element-manager
2017-05-25 22:28:06,444 - vcpus: 2
2017-05-25 22:28:06,457 - memory_mb: 4096
2017-05-25 22:28:06,469 - root_disk_mb: 40960
2017-05-25 22:28:06,481 - ephemeral_disk_mb: 0
2017-05-25 22:28:06,491 - swap_disk_mb: 0
2017-05-25 22:28:06,505 - NETCONF edit-config Request sent, waiting for reply
2017-05-25 22:28:06,781 - NETCONF Transaction success!
2017-05-25 22:28:06,793 - Waiting for VNF to process CREATE_FLAVOR transaction
2017-05-25 22:28:07,286 - | CREATE_FLAVOR | ab-auto-test-vnfm2-element-manager | SUCCESS
| (1/1)
2017-05-25 22:28:07,298 - NETCONF transaction completed successfully!
2017-05-25 22:28:07,310 - Creating flavors
2017-05-25 22:28:07,328 - flavor: ab-auto-test-vnfm2-control-function
2017-05-25 22:28:07,341 - vcpus: 8
2017-05-25 22:28:07,358 - memory_mb: 16384
2017-05-25 22:28:07,374 - root_disk_mb: 6144
2017-05-25 22:28:07,386 - ephemeral_disk_mb: 0
2017-05-25 22:28:07,398 - swap_disk_mb: 0
2017-05-25 22:28:07,410 - NETCONF edit-config Request sent, waiting for reply
2017-05-25 22:28:07,586 - NETCONF Transaction success!
2017-05-25 22:28:07,603 - Waiting for VNF to process CREATE_FLAVOR transaction
```

```

2017-05-25 22:28:07,818 - | CREATE_FLAVOR | ab-auto-test-vnfm2-control-function | SUCCESS
| (1/1)
2017-05-25 22:28:07,830 - NETCONF transaction completed successfully!
2017-05-25 22:28:07,842 - Creating flavors
2017-05-25 22:28:07,853 -   flavor: ab-auto-test-vnfm2-session-function
2017-05-25 22:28:07,865 -     vcpus: 8
2017-05-25 22:28:07,877 -     memory_mb: 16384
2017-05-25 22:28:07,889 -     root_disk_mb: 6144
2017-05-25 22:28:07,901 -     ephemeral_disk_mb: 0
2017-05-25 22:28:07,917 -     swap_disk_mb: 0
2017-05-25 22:28:07,928 - NETCONF edit-config Request sent, waiting for reply
2017-05-25 22:28:08,204 - NETCONF Transaction success!
2017-05-25 22:28:08,216 - Waiting for VNFM to process CREATE_FLAVOR transaction
2017-05-25 22:28:08,455 - | CREATE_FLAVOR | ab-auto-test-vnfm2-session-function | SUCCESS
| (1/1)
2017-05-25 22:28:08,473 - NETCONF transaction completed successfully!
2017-05-25 22:28:08,489 - Flavors created successfully
2017-05-25 22:28:08,501 - Onboarding configuration file: ('control-function',
'staros_config.txt', 'http://172.21.201.63:5001/configs/vnf-pkg2/files/system.cfg')
2017-05-25 22:28:08,547 - NETCONF get-operational Request sent, waiting for reply
2017-05-25 22:28:08,724 - NETCONF Transaction success!
2017-05-25 22:28:08,855 - Notify VDU Create Catalog for : element-manager, status:
SUCCESS, txid: 562c18b0-4199-11e7-ad05-fa163ec6a7e4
2017-05-25 22:28:08,892 - Notify VDU Create Catalog for : control-function, status:
SUCCESS, txid: 562c18b0-4199-11e7-ad05-fa163ec6a7e4
2017-05-25 22:28:09,008 - Notify VDU Create Catalog for : session-function, status:
SUCCESS, txid: 562c18b0-4199-11e7-ad05-fa163ec6a7e4
2017-05-25 22:28:09,024 - NETCONF get-config Request sent, waiting for reply
2017-05-25 22:28:09,151 - NETCONF Transaction success!
2017-05-25 22:28:14,837 - Deployment: vnfd2-deployment started ...
2017-05-25 22:28:14,858 - Generating VNFD
2017-05-25 22:28:14,930 - VNFD generated successfully.
2017-05-25 22:28:14,966 - Generating configuration files for EM
2017-05-25 22:28:14,979 - Creating VIP Ports
2017-05-25 22:28:16,970 - VIP ports created successfully
2017-05-25 22:28:16,987 - Deploying EM
2017-05-25 22:28:17,000 - Added anti-affinity placement policy for ab-auto-test-vnfm2-em-1
2017-05-25 22:28:17,012 - Added anti-affinity placement policy for ab-auto-test-vnfm2-em-2
2017-05-25 22:28:17,025 - Added anti-affinity placement policy for ab-auto-test-vnfm2-em-3
2017-05-25 22:28:17,041 - Starting Service Deployment: ab-auto-test-vnfm2-em
2017-05-25 22:28:17,054 - Start VM: ab-auto-test-vnfm2-em-1
2017-05-25 22:28:17,066 - Start VM: ab-auto-test-vnfm2-em-2
2017-05-25 22:28:17,077 - Start VM: ab-auto-test-vnfm2-em-3
2017-05-25 22:28:17,089 - NETCONF edit-config Request sent, waiting for reply
2017-05-25 22:28:17,721 - NETCONF Transaction success!
2017-05-25 22:28:17,733 - Waiting for VNFM to process SERVICE_ALIVE transaction
2017-05-25 22:29:37,185 - | VM_DEPLOYED | ab-auto-test-vnfm2-em-1 | SUCCESS | Waiting
for: SERVICE_ALIVE|
2017-05-25 22:29:59,679 - | VM_ALIVE | ab-auto-test-vnfm2-em-1 | SUCCESS | Waiting for:
SERVICE_ALIVE|
2017-05-25 22:30:42,170 - | VM_DEPLOYED | ab-auto-test-vnfm2-em-2 | SUCCESS | Waiting
for: SERVICE_ALIVE|
2017-05-25 22:30:59,620 - | VM_ALIVE | ab-auto-test-vnfm2-em-2 | SUCCESS | Waiting for:
SERVICE_ALIVE|
2017-05-25 22:31:51,510 - | VM_DEPLOYED | ab-auto-test-vnfm2-em-3 | SUCCESS | Waiting
for: SERVICE_ALIVE|
2017-05-25 22:32:13,584 - | VM_DEPLOYED | c2 | SUCCESS | Waiting for: SERVICE_ALIVE|
2017-05-25 22:32:29,639 - | VM_ALIVE | ab-auto-test-vnfm2-em-3 | SUCCESS | Waiting for:
SERVICE_ALIVE|
2017-05-25 22:32:29,661 - | SERVICE_ALIVE | ab-auto-test-vnfm2-em | SUCCESS | (1/1)
2017-05-25 22:32:29,674 - NETCONF transaction completed successfully!
2017-05-25 22:32:29,687 - EM Online !
2017-05-25 22:32:29,699 - HA-VIP[element-manager] : 172.67.11.12
2017-05-25 22:32:29,716 - HA-VIP[control-function] : 172.67.11.13

```

```

2017-05-25 22:32:29,729 - Deployment: vnf2-deployment completed successfully.
2017-05-25 22:32:29,742 - NETCONF get-operational Request sent, waiting for reply
2017-05-25 22:32:30,221 - NETCONF Transaction success!
2017-05-25 22:32:30,261 - Notify EM Up
2017-05-25 22:32:30,274 - VNF Transaction completed successfully!
2017-05-25 22:32:30,292 - Notify deployment</log>
</logs>
</config>

```

## Viewing AutoVNF Operational Data

AutoVNF maintains history information for all transactions, associated events, and related error/information logs in persistent storage. These logs are useful for monitoring deployment progress and for troubleshooting issues.

These logs can be retrieved at time using the “task-id” returned as well as by running ConfD “show” commands.

To access these commands, you must be logged in to the ConfD CLI as the *admin* user on the AutoVNF VM:

```
confd_cli -u admin -C
```

When prompted, enter the *admin* user password.

[Table 18: ConfD Log Descriptions, on page 143](#) provides a list of the available commands and describes the information in the output.

**Table 18: ConfD Log Descriptions**

ConfD Command	Purpose
In releases prior to 6.0: <b>show autovnf-oper:errors</b> In 6.0 and later releases: <b>show uas</b>	Displays a list of any deployment errors that may have occurred.
In releases prior to 6.0: <b>show autovnf-oper:logs   display xml</b> In 6.0 and later releases: <b>show log   display xml</b>	Displays log messages for AutoVNF transactions.
In releases prior to 6.0: <b>show autovnf-oper:network-catalog</b> In 6.0 and later releases: <b>show vnf-packager</b>	Displays information for the networks deployed with USP.
In releases prior to 6.0: <b>show autovnf-oper:transactions</b> In 6.0 and later releases: <b>show transaction</b>	Displays a list of transaction IDs that correspond to the USP deployment along with their execution date, time, and status.

ConfD Command	Purpose
In releases prior to 6.0: <b>show autovnf-oper:vdu-catalog</b>  In 6.0 and later releases: <b>show vnfr</b>	Displays information pertaining to the virtual descriptor units (VDUs) used to deploy USP.
In releases prior to 6.0: <b>show autovnf-oper:vip-port</b>  In 6.0 and later releases: <b>show vnfr</b>	Displays information port, network, and virtual IP addresses information.
In releases prior to 6.0: <b>show autovnf-oper:vnfm</b>  In 6.0 and later releases: <b>show vnfr</b>	Displays information pertaining to the VNFM deployment and UEM VM deployment.
<b>show confd-state</b>	Displays information pertaining to confd-state on AutoVNF.
<b>show confd-state ha</b>	Displays information pertaining to HA specific confd-state on AutoVNF.
<b>show log &lt;transaction_id&gt;</b>	Displays detailed log information for a specific transaction ID.
<b>show running-config</b>	Displays the configuration running on the AutoVNF.
<b>show uas</b>	Displays information pertaining to the AutoVNF VM deployment.
In releases prior to 6.0: <b>show usp</b>  In 6.0 and later releases: <b>show vnfr</b>	Displays information pertaining to the overall USP VM deployment.

**NOTES:**

- Log information can be saved out of ConfD to a file for later retrieval using one of the following commands:

```
show log transaction_id | save url
```

OR

```
show autovnf-oper: command | save url
```

Where *transaction\_id* is a specific ID, *url* is a valid directory path, and *command* is one of the command operators identified in [Table 18: ConfD Log Descriptions, on page 143](#).

## Example show confd-state Command Output

### show confd-state

```
confd-state version 6.3.1
confd-state epoll false
confd-state daemon-status started
confd-state ha mode master
confd-state ha node-id confd-master
confd-state ha connected-slave [ a2dd5178-afae-4b3a-8b2b-910216583501 ]
```

NAME	PREFIX	EXPORTED		NAMESPACE
		TO ALL	EXPORTED TO	
iana-crypt-hash			2014-08-06	urn:ietf:params:xml:ns:yang:iana-crypt-hash
ianach		X	-	
ietf-inet-types			2013-07-15	urn:ietf:params:xml:ns:yang:ietf-inet-types
inet		X	-	
ietf-netconf-acm			2012-02-22	urn:ietf:params:xml:ns:yang:ietf-netconf-acm
nacm		X	-	
ietf-netconf-monitoring			2010-10-04	urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring
ncm		X	-	

<-- SNIP -->

## Example show confd-state ha Command Output

### show confd-state ha

```
confd-state ha mode master
confd-state ha node-id confd-master
confd-state ha connected-slave [ a2dd5178-afae-4b3a-8b2b-910216583501 ]
```

## Example show log Command Output

```
show log <transaction_id> | display xml
```

## Example show running-config Command Output

### show running-config

```
<-- SNIP -->
autovnf:secure-token autovnf-admin
  user      $8$YQiswhu0QLpA4N2kBo7t5eZN2uUW0L19m8WaaBzkVoc=
  password  $8$mSaszfxjZ8My8Y/FqLL3Sasn1b/DmRh3pdblatq49cM=
  !
autovnf:secure-token autovnf-oper
  user      $8$kTEQZ4YNdV6BcnH3ggRHJJPmhk61sh5KQFqhsQnh/KV8=
  password  $8$KdTBd7ZeYuHrpdKlK5m888ckE3ZGIM7RbEMJwMwCjfo=
  !
autovnf:secure-token em-login
  user      $8$jVDkSMi/WlXzkZj/qx07kEfHB9PlpPlnzCKUSjWiPXA=
  password  $8$52ELrKMilGT/nad5WcPgUh7cijHiizAt8A8Tly79Q/I=
  !
autovnf:secure-token confd-auth
  user      $8$bHYvP179/hlGW08qoTnJFmm8A1HqqlRESasX+G1SAPw=
  password  $8$S52APq1vb9WhLjbSPNSWiBmAmaG1tzTTmSkkkKs8reo=
  !
volume-catalog em-volume
  volume type LUKS
  volume size 1024
  volume bus ide
```

## Example show uas Command Output

```

    volume bootable false
    !
volume-catalog cf-boot
  volume type LUKS
  volume size 16
  volume bus ide
  volume bootable true
  !
volume-catalog cf-cdr
  volume type LUKS
  volume size 200
  volume bus ide
  volume bootable false
  !
autovnf:network-catalog di-internall
  pre-created di-internall
  type          sriov-flat
  physnet       phys_pcie1_0
  ip-prefix     192.168.1.0/24
  dhcp          true
  vlan-tag      true
  vlan          2110
<-- SNIP -->

<-- SNIP -->
autovnf:vdu-catalog control-function
  ha-type          one-to-one
  health-check-frequency 10
  health-probe-max-miss 6
  recovery-type    recovery-restart
  image location  http://172.21.201.63:80/bundles/5.1.0-662/ugp-bundle/qvpc-di-cf.qcow2
  neds netconf
  ned-id          cisco-staros-nc
  port-number     830
  authentication  confd-auth
  !
volumes cf-cdr
  !
volumes cf-boot
  !
flavor host-aggregate  auto-test-sjc-cf-esc-mgmt1
flavor vcpus           8
flavor ram             16384
flavor root-disk       6
flavor ephemeral-disk 0
flavor swap-disk       0
flavor anti-affinity-placement true
configuration staros_config.txt
  apply-at day-zero
  source-url http://172.21.201.63:5001/configs/vnf-pkg1/files/system.cfg
<-- SNIP -->

```

## Example show uas Command Output

```

show uas
uas version          6.0.0
uas state            active
uas external-connection-point 50.50.50.67 INSTANCE
IP      STATE  ROLE
-----
10.2.3.6  alive  CONFD-MASTER
10.2.3.11 alive  CONFD-SLAVE

```

NAME	LAST HEARTBEAT
AutoVNF-MASTER	2018-01-20 02:35:03
ESCHearBeatMonitor-fremont-autovnf-vpc	2018-01-20 02:35:00
USPCFMWorker	2018-01-20 02:34:51
USPCHBWorker	2018-01-20 02:35:00
USPCWorker	2018-01-20 02:35:00



**Important** In this example, 10.2.3.6 is the confd-master and the active UAS VM.



**Important** In case of standalone mode (non-HA) deployments, the *uas external-connection-point* information and *Instance IP* table are not applicable and are not displayed.

Example output that shows the floating IP for AutoVNF:

```
-SNIP-
nsd autoit
vim-identity vim1
vim-artifact vim_artifact_one
vnf-package [ usp_5_7 ]
vld mgmt
  vl-type          management
  network-instance bmarconi-management
!
vld orch
  vl-type orchestration
  network sjc-orch
!
vnfd f-autovnf
vnf-type          usp-uas
version           6.0
high-availability true
nsd               fremont-autovnf
configuration boot-time 1800
configuration set-vim-instance-name true
external-connection-point avf
  connection-point eth0
  floating-ip enabled
  floating-ip external-network public
!
vnfc avf
health-check disabled
health-check boot-time 300
vdu vdu-id autovnf
connection-point eth0
  virtual-link service-vl mgmt
!
  connection-point eth1
  virtual-link service-vl orch
!
!
!
-SNIP-
```

The current version of AutoVNF software can also be seen through the USP UWS – AutoVNF User Interface under –

## Example show vnfr Command Output

- the Site Overview screen (Service Deployment > Site) only if the AutoVNF configuration type is a record.
- the Auto-Vnf Configuration Overview screen only if the AutoVNF configuration type is a record.
- the UWS – AutoVNF dashboard.

## Example show vnfr Command Output

**show vnfr**

```

vnfr sj-autovnf-esc
vnfd     esc
vnf-type esc
state    deployed
external-connection-point esc
connection-point-instance-id sj-autovnf-esc-ha-vip
virtual-link-ref             uas-management
ip-address                   12.12.12.40
mac-address                  fa:16:3e:6a:db:9b
connection-point-type        virtual-port
port-id                      37a14e07-52f7-48c0-9dbb-471146a709a5
vdu esc
vnfc-instance sj-autovnf-esc-esc-1
state          deployed
vnfc           esc
flavor-key     sj-autovnf-esc
uuid           83f44e0f-380e-4320-a35a-34de82cf84dd
image name     /vnfm-bundle/ESC-4_2_0_74.qcow2
image version  "Version: 4.2.0.74, SHA1: de45b53, Date: Sat Sep 01 08:51:12 EDT 2018"
image package  usp_6_0
image uuid     c35c2a86-6d60-4259-85cc-d023803c7245
host           tb2-compute-15.localdomain
vdu-type       cisco-esc
connection-point-instance eth0
virtual-link-ref             uas-management
ip-address                   12.12.12.22
mac-address                  fa:16:3e:e8:d6:b1
connection-point-type        virtual-port
port-id                      f0f6b82f-336f-4f9f-aae5-d581be8cfa63
connection-point-instance eth1
virtual-link-ref             uas-orchestration
ip-address                   22.22.22.27
mac-address                  fa:16:3e:16:32:4c
connection-point-type        virtual-port
port-id                      f2b7aeae-83f1-4f83-b45e-f92b3a1f6600
vnfc-instance sj-autovnf-esc-esc-2
state          deployed
vnfc           esc
flavor-key     sj-autovnf-esc-esc
uuid           087a5b48-db45-4002-a157-51fa37236545
image name     /vnfm-bundle/ESC-4_2_0_74.qcow2
image version  "Version: 4.2.0.74, SHA1: de45b53, Date: Sat Sep 01 08:51:12 EDT 2018"
image package  usp_6_0
image uuid     c35c2a86-6d60-4259-85cc-d023803c7245
host           tb2-compute-12.localdomain
vdu-type       cisco-esc
connection-point-instance eth0
virtual-link-ref             uas-management
ip-address                   12.12.12.37
mac-address                  fa:16:3e:48:c4:6c
connection-point-type        virtual-port
port-id                      8cb138ab-c575-4eb2-a622-d2648042f48f
connection-point-instance eth1

```

```

    virtual-link-ref    uas-orchestration
    ip-address          22.22.22.28
    mac-address         fa:16:3e:98:78:07
    connection-point-type virtual-port
    port-id             7d73aeae-81e1-410b-ac3a-e34c1bd23c16
vnfr sj-autovnf-vpc
  vnfd      vpc
  vnf-type  ugp
  state     ha-error
external-connection-point cf
  connection-point-instance-id CF-sj-autovnf-vpc-vip
  virtual-link-ref            uas-management
  ip-address                  12.12.12.43
  mac-address                  fa:16:3e:04:80:b7
  connection-point-type      virtual-port
  port-id                     984a6e8b-107a-48f7-b0b4-398a308aff9a
external-connection-point em
  connection-point-instance-id em-sj-autovnf-vpc-vip
  virtual-link-ref            uas-management
  ip-address                  12.12.12.35
  mac-address                  fa:16:3e:b4:7e:b8
  connection-point-type      virtual-port
  port-id                     f47c2150-932c-455f-99c1-7b77fe47a9d7
vdu cf
vnfc-instance sj-autovnf-vpc-cf-0
  state      alive
  vnfc      cf
  flavor-key sj-autovnf-vpc-cf
  uuid      a46de643-b76d-4307-91e8-996b79da4c1e
  image name /ugp-bundle/qvpc-di-cf.qcow2
  image version "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
  image package usp_6_0
  image uuid 6d63f613-9b46-4bd9-853d-024dcf27f1a7
  host      tb2-compute-9.localdomain
  vdu-type  control-function
connection-point-instance eth0
  virtual-link-ref      di-internall
  ip-address            192.168.10.105
  mac-address           fa:16:3e:46:f8:79
  connection-point-type pnic-sriov
  port-id               b408eedd-8650-44e2-930c-95ee2c9ae380
connection-point-instance eth1
  virtual-link-ref      uas-management
  ip-address            12.12.12.44
  mac-address           fa:16:3e:5e:e0:bc
  connection-point-type virtual-port
  port-id               3e94bcd8-0e58-44e1-99a5-366f7453df02
connection-point-instance eth2
  virtual-link-ref      uas-orchestration
  ip-address            22.22.22.33
  mac-address           fa:16:3e:c5:58:c6
  connection-point-type virtual-port
  port-id               e0a51253-5740-4e34-b4a2-ba6cdaa504cf
vnfc-instance sj-autovnf-vpc-cf-1
  state      alive
  vnfc      cf
  flavor-key sj-autovnf-vpc-cf
  uuid      10ble4c2-d3e5-494c-bec9-26bd38e4c705
  image name /ugp-bundle/qvpc-di-cf.qcow2
  image version "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
  image package usp_6_0
  image uuid 6d63f613-9b46-4bd9-853d-024dcf27f1a7
  host      tb2-compute-12.localdomain
  vdu-type  control-function

```

## Example show vnfr Command Output

```

connection-point-instance eth0
  virtual-link-ref      di-internall1
  ip-address            192.168.10.99
  mac-address           fa:16:3e:94:3d:38
  connection-point-type pnic-sriov
  port-id               c1df9769-fcdc-4cb1-b7ea-f791ef80ff65
connection-point-instance eth1
  virtual-link-ref      uas-management
  ip-address            12.12.12.47
  mac-address           fa:16:3e:66:27:71
  connection-point-type virtual-port
  port-id               7d77aac2-6409-499a-a4b0-afc4c70e6904
connection-point-instance eth2
  virtual-link-ref      uas-orchestration
  ip-address            22.22.22.45
  mac-address           fa:16:3e:c3:c1:a4
  connection-point-type virtual-port
  port-id               75d7b8c7-1801-4cce-b665-64a060414abd
vdu em
vnfc-instance sj-autovnf-vpc-em-1
  state                 ha-error
  vnfc                  em
  flavor-key            sj-autovnf-vpc-em
  uuid                  119edc4c-9ba0-48f8-a928-63e0c3c88f22
  image name            /em-bundle/em-6_3_0_4148.qcow2
  image version         "Version: 6.3.0, SHA1: 40d8f29, Date: Thu Aug 30 22:15:22 EDT 2018"
  image package         usp_6_0
  image uuid            d21b6d92-9964-4db8-8376-4a645fecfbf2
  host                  tb2-compute-14.localdomain
  vdu-type              element-manager
connection-point-instance eth0
  virtual-link-ref      uas-orchestration
  ip-address            22.22.22.40
  mac-address           fa:16:3e:33:57:a6
  connection-point-type virtual-port
  port-id               050d8843-f309-45b3-889a-a1516a338c9f
connection-point-instance eth1
  virtual-link-ref      uas-management
  ip-address            12.12.12.26
  mac-address           fa:16:3e:02:b8:4a
  connection-point-type virtual-port
  port-id               ae8036c5-1a91-488d-98f2-65a8fe57a033
vnfc-instance sj-autovnf-vpc-em-2
  state                 ha-error
  vnfc                  em
  flavor-key            sj-autovnf-vpc-em
  uuid                  dd2c9327-c954-49bf-803c-ca38d718da2c
  image name            /em-bundle/em-6_3_0_4148.qcow2
  image version         "Version: 6.3.0, SHA1: 40d8f29, Date: Thu Aug 30 22:15:22 EDT 2018"
  image package         usp_6_0
  image uuid            d21b6d92-9964-4db8-8376-4a645fecfbf2
  host                  tb2-compute-15.localdomain
  vdu-type              element-manager
connection-point-instance eth0
  virtual-link-ref      uas-orchestration
  ip-address            22.22.22.46
  mac-address           fa:16:3e:e5:f7:18
  connection-point-type virtual-port
  port-id               30816589-9a12-4c1d-840c-c84100f714f4
connection-point-instance eth1
  virtual-link-ref      uas-management
  ip-address            12.12.12.45
  mac-address           fa:16:3e:f3:ff:4e
  connection-point-type virtual-port

```

```

    port-id                cf9d991f-e45b-41ed-9ac1-7e6f0bee620b
vdu sf
vnfc-instance sj-autovnf-vpc-sf-0
state          alive
vnfc           sf
flavor-key    sj-autovnf-vpc-sf
uuid          d9b13253-a67e-4078-a75c-04d834577cc2
image name    /ugp-bundle/qvpc-di-xf.qcow2
image version "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
image package usp_6_0
image uuid    c65df544-0230-4e86-88bf-4aa93e0e268d
host          tb2-compute-14.localdomain
vdu-type      session-function
connection-point-instance eth0
virtual-link-ref          di-internal1
ip-address                192.168.10.95
mac-address               fa:16:3e:87:49:22
connection-point-type     pnic-sriov
port-id                   5d9a9a89-5857-48cb-8081-7273c4b9354c
connection-point-instance eth1
virtual-link-ref          uas-orchestration
ip-address                22.22.22.18
mac-address               fa:16:3e:8f:47:ce
connection-point-type     virtual-port
port-id                   d7dd7006-0134-4767-af02-1922d351d1d5
connection-point-instance eth2
virtual-link-ref          vpc-svc
ip-address                22.11.11.8
mac-address               fa:16:3e:a6:fa:9e
connection-point-type     virtual-port
port-id                   1c5dda23-65f0-4541-ace5-0d6e5e1564ea
vnfc-instance sj-autovnf-vpc-sf-1
state          alive
vnfc           sf
flavor-key    sj-autovnf-vpc-sf
uuid          868158de-e202-4af4-9f3e-c5c7722c5a7f
image name    /ugp-bundle/qvpc-di-xf.qcow2
image version "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
image package usp_6_0
image uuid    c65df544-0230-4e86-88bf-4aa93e0e268d
host          tb2-compute-15.localdomain
vdu-type      session-function
connection-point-instance eth0
virtual-link-ref          di-internal1
ip-address                192.168.10.97
mac-address               fa:16:3e:bb:ee:38
connection-point-type     pnic-sriov
port-id                   c166c76d-3ef9-4f52-a243-25b49ae0886f
connection-point-instance eth1
virtual-link-ref          uas-orchestration
ip-address                22.22.22.47
mac-address               fa:16:3e:b0:8e:75
connection-point-type     virtual-port
port-id                   9dd61ba8-9455-4f0a-a6ce-13ef28ce6c39
connection-point-instance eth2
virtual-link-ref          vpc-svc
ip-address                22.11.11.13
mac-address               fa:16:3e:25:5a:56
connection-point-type     virtual-port
port-id                   3f8b60aa-4155-4192-b537-afb812d784da

```

## Example show vnf-packager Command Output

### show vnf-packager

```

version      "Version: 6.4.M0, SHA1: cdd46bcm, Build-Number: 0"
image application-function
  image-uri  /ugp-bundle/qvpc-di-xf.qcow2
  vim-id     c65df544-0230-4e86-88bf-4aa93e0e268d
  version    "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
  disk-format qcow2
image automation-service
  image-uri  /uas-bundle/usp-uas-6.3.0-0.qcow2
  vim-id     b32d2aeb-9dbe-42f0-99bf-982db8ae7ae8
  version    "Version: 6.3.0, SHA1: 175ea8em, Date: Thu Sep 06 16:17:26 PDT 2018"
  disk-format qcow2
image cisco-esc
  image-uri  /vnfm-bundle/ESC-4_2_0_74.qcow2
  vim-id     c35c2a86-6d60-4259-85cc-d023803c7245
  version    "Version: 4.2.0.74, SHA1: de45b53, Date: Sat Sep 01 08:51:12 EDT 2018"
  disk-format qcow2
image control-function
  image-uri  /ugp-bundle/qvpc-di-cf.qcow2
  vim-id     6d63f613-9b46-4bd9-853d-024dcf27f1a7
  version    "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
  disk-format qcow2
image element-manager
  image-uri  /em-bundle/em-6_3_0_4148.qcow2
  vim-id     d21b6d92-9964-4db8-8376-4a645fecfbf2
  version    "Version: 6.3.0, SHA1: 40d8f29, Date: Thu Aug 30 22:15:22 EDT 2018"
  disk-format qcow2
image network-function
  image-uri  /ugp-bundle/qvpc-di-xf.qcow2
  vim-id     c65df544-0230-4e86-88bf-4aa93e0e268d
  version    "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
  disk-format qcow2
image session-function
  image-uri  /ugp-bundle/qvpc-di-xf.qcow2
  vim-id     c65df544-0230-4e86-88bf-4aa93e0e268d
  version    "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
  disk-format qcow2
image user-plane-function
  image-uri  /ugp-bundle/qvpc-si-21.10.M0.70226.qcow2
  vim-id     078bc882-d29c-4974-a21d-dbf2bc59149b
  version    "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
  disk-format qcow2
configuration bootvm
  data-id 1538437650-071830
configuration staros
  data-id 1538437650-060109
vnf-packager 6.4.M0-6133
vnf-package usp_6_t
  version    "Version: 6.4.M0, SHA1: cdd46bcm, Build-Number: 6133"
image application-function
  image-uri  /ugp-bundle/qvpc-di-xf.qcow2
  vim-id     d3b3dd85-464d-4b49-90f1-5dc59c9a111b
  version    "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
  disk-format qcow2
image automation-service
  image-uri  /uas-bundle/usp-uas-6.3.0-4206.qcow2
  vim-id     294e5f52-453a-4bd8-8192-b8144607759f
  version    "Version: 6.3.0, SHA1: 175ea8e, Date: Wed Sep 05 06:15:40 EDT 2018"
  disk-format qcow2
image cisco-esc

```

```

image-uri /vnfm-bundle/ESC-4_2_0_74.qcow2
vim-id 87a322cc-3736-407d-855f-f2a566fadd22
version "Version: 4.2.0.74, SHA1: de45b53, Date: Sat Sep 01 08:51:12 EDT 2018"
disk-format qcow2
image control-function
image-uri /ugp-bundle/qvpc-di-cf.qcow2
vim-id 22b34ebf-060c-4e99-8083-e702cef96aca
version "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
disk-format qcow2
image element-manager
image-uri /em-bundle/em-6_3_0_4148.qcow2
vim-id c4424476-a9b6-4308-98b3-4aa0f441d5c1
version "Version: 6.3.0, SHA1: 40d8f29, Date: Thu Aug 30 22:15:22 EDT 2018"
disk-format qcow2
image network-function
image-uri /ugp-bundle/qvpc-di-xf.qcow2
vim-id d3b3dd85-464d-4b49-90f1-5dc59c9a111b
version "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
disk-format qcow2
image session-function
image-uri /ugp-bundle/qvpc-di-xf.qcow2
vim-id d3b3dd85-464d-4b49-90f1-5dc59c9a111b
version "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
disk-format qcow2
image user-plane-function
image-uri /ugp-bundle/qvpc-si-21.10.M0.70226.qcow2
vim-id 8c78ef58-4556-4e8c-bef6-8f98a33bf6c1
version "Version: 21.10.M0.70226, SHA1: NA, Date: Thu Sep 06 10:07:27 EDT 2018"
disk-format qcow2
configuration bootvm
data-id 1538437651-235460
configuration staros
data-id 1538437651-221341

```

## UAS Log Collection

The UAS generates and consolidates a comprehensive set of UAS logs and VNF diagnostic information from StarOS device for troubleshooting purposes. The log collection includes logs from all components in a deployed UAS cluster, i.e. from AutoIT, AutoDeploy, AutoVNF, UEM, ESC and StarOS.

This section describes the following topics:

- [Feature Description, on page 154](#)
- [Limitations, on page 154](#)
- [Collecting the UAS Logs, on page 154](#)
  - [Via the ConfD CLI Command, on page 154](#)
  - [Via the Standalone Script, on page 155](#)
- [Collecting VNF Diagnostic Information, on page 158](#)
- [Sample Logs, on page 158](#)

## Feature Description

Automation of UAS log collection is facilitated through the use of ConfD CLI command **collect-logs**. When this command is executed, the logs from all or required components can be collected and copied to a common location.

## Limitations

The following limitations exist with the UAS log collection feature.

- With the use of ConfD CLI approach, it is not possible to collect logs for AutoDeploy and AutoIT. To aggregate logs for these two components, use the standalone script.
- Direct collection of logs from CF, SF, UP instances is not supported. If VNFDs corresponding to these instances are invoked directly, appropriate error message will be recorded in the log file of AutoDeploy and AutoVNF.

## Collecting the UAS Logs

The UAS logs can be collected using one of the following approaches:

- [Via the ConfD CLI Command, on page 154](#)
- [Via the Standalone Script, on page 155](#)

### Via the ConfD CLI Command

UAS logs collection is automated through a remote procedure call (RPC) executed from the ConfD command line interface (CLI). The RPC “collect-logs” has been introduced to collect logs from the AutoVNF, ESC and UEM.

This command, on execution, fetches the logs from components under given NSD and VNFD levels in a deployed setup and creates a final consolidated tar ball comprising all logs.




---

#### Important

When the ConfD CLI command is used, the log collection for AutoDeploy and AutoIT components is not supported.

---

This command can be invoked from ConfD CLI of AutoDeploy and AutoVNF.




---

#### Important

The IP and credentials of instances are fetched from oper data of AutoDeploy/AutoVNF. So, the logs can be collected for instances whose VNFR is present in oper db.

---

To collect the UAS logs via ConfD CLI:

1. Login to the ConfD CLI as the *admin* user on the AutoDeploy VM or AutoVNF VM.

```
confd_cli -u admin -C
```

2. Enter the *admin* user password when prompted.

- Execute the following command:

```
collect-logs nsd-id <nsd id>
```

This command collects logs from all components present under given NSD and also from deploy-nsd if present.

To collect logs by VNFD ID, use the following command:

```
collect-logs nsd-id <nsd id> vnfd vnf id
```

This command collects logs from components specific to the given VNFD. Additionally, under given vnf, corresponding AutoVNF logs of the given NSD will also be collected in case of RPC invoked from AutoDeploy (outside AutoVNF).

For example, if VNFD of ESC, or UEM is invoked from AutoDeploy, the corresponding AutoVNF logs of given VNFD will also be collected, since log collection happens through respective AutoVNF in case of multiple AutoVNFs deployed through a single AutoDeploy.

The vnf-diags from StarOS instances will be collected along with the UEM logs.

- View the status of log collection using the AutoDeploy or AutoVNF logs under `/var/log/upstart/` based on where it is invoked.

If invoked from AutoDeploy, the RPC internally connects with AutoVNF and collects logs from UAS instances. The respective progress can be viewed from AutoVNF log.

- Untar the `autocollect_logs.tgz` file to extract the collected logs.

```
tar -zxvf autocollect_logs.tgz
```

Note that the output consolidated tar ball `autocollect_logs.tgz` is created under `/var/log/autocollect/` directory of the instance (AutoDeploy/AutoVNF) from where the log collection RPC was invoked.



### Important

Every time the log collection is triggered, the `/var/log/autocollect` directory will be automatically cleaned and then new logs will be copied.

Example output:

```
vnf-logs/
vnf-logs/em/
vnf-logs/em/abc-vnf-vnf1-em-rmuruga-em1-1-em-logs-2018-08-09_06.41.25.UTC.tar.bz2
vnf-logs/em/abc-vnf-vnf1-em-rmuruga-em1-2-em-logs-2018-08-09_06.40.35.UTC.tar.bz2
vnf-logs/autovnf/
vnf-logs/autovnf/autoit-f-autovnf1-rmuruga-avf-2-uas-logs-2018-08-09_06.40.06.UTC.tar.bz2
vnf-logs/autovnf/autoit-f-autovnf1-rmuruga-avf-1-uas-logs-2018-08-09_06.40.49.UTC.tar.bz2
vnf-logs/esc/
vnf-logs/esc/esc_log_abc-vnf-vnf1-esc-rmuruga-esc-2_2018-08-09_06.40.15.UTC.tar.bz2
vnf-logs/esc/esc_log_abc-vnf-vnf1-esc-rmuruga-esc-1_2018-08-09_06.39.42.UTC.tar.bz2
```

## Via the Standalone Script

The UAS uses a standalone script "`collect_all_uas_logs.py`" to generate and consolidate a comprehensive set of UAS logs and VNF diagnostic information from StarOS device for troubleshooting purposes.

Before using this script, you should be aware of the following:

- AutoVNF IP and SSH credentials (username and password)

- AutoVNF NETCONF login credentials
- Login credentials for UEM and ESC
- Login credentials for AutoIT and AutoDeploy

This script is available in the AutoIT, AutoDeploy and AutoVNF VMs in the `/opt/cisco/usp/uas/scripts/` directory.

A sample yaml file (`sample_config.yaml`) is present in the same directory along with the script. The yaml file should be updated with proper IP and credential details.

Example configuration of `sample_config.yaml` file:

```
uas-cluster:
  autovnf:
    172.21.201.237:
      autovnf:
        login:
          user: ubuntu
          password: Cisco@123
      netconf:
        user: admin
        password: Cisco@123
  .
  .
  .
```




---

**Important**

You can exclude AutoDeploy or AutoIT for log collection by commenting out the phrase ‘autodeploy’ or ‘autoit’ using ‘#’ in the yaml file.

For example:

```
#autodeploy:
```

---

To collect the UAS logs via the standalone script:

1. Log on to the AutoDeploy, AutoIT or AutoVNF VM using the user credentials specified during deployment.
2. Navigate to the `scripts` directory.

```
cd /opt/cisco/usp/uas/scripts
```

3. Edit the `sample_config.yaml` file using a standard text editor to update the appropriate credential details and save it.
4. Launch the `collect_all_uas_logs.py` script to collect the logs.

```
sudo ./collect_all_uas_logs.py - -cfgfile sample_config.yaml
```

The script starts collecting the logs and the progress is displayed on the console. Upon completion, a final tar file `autocollect_logs.tgz` will be copied to `/var/log/autocollect/` directory.

Detailed logs are stored in a log file named `autocollect.log` under the `/var/log/autocollect/` directory.




---

**Note**

The logs directory will be cleared and re-created for every execution of the script.

---

Example output:

```
sudo ./collect_all_uas_logs.py --cfgfile sample_config.yaml
2018-07-26 08:52:57,273 - Uas-cluster present in config
2018-07-26 08:52:57,273 - .....Executing step 1 of 4.....
2018-07-26 08:52:57,273 - Collecting logs from uas instances
2018-07-26 08:53:06,530 - Logs collected successfully from esc 48.48.48.24
2018-07-26 08:53:09,677 - Logs collected successfully from autovnf 38.38.38.23
2018-07-26 08:53:09,784 - Logs collected successfully from esc 48.48.48.16
2018-07-26 08:53:15,404 - Logs collected successfully from autovnf 38.38.38.26
2018-07-26 08:54:21,160 - Logs collected successfully from vnf-em 38.38.38.11
2018-07-26 08:54:24,829 - Logs collected successfully from vnf-em 38.38.38.27
2018-07-26 08:54:24,831 - Generating tar file for uas instance 10.225.202.93
2018-07-26 08:54:25,260 - Log collection completed for uas instance 10.225.202.93.
2018-07-26 08:54:25,261 - Auto-it present in config
2018-07-26 08:54:25,261 - .....Executing step 2 of 4.....
2018-07-26 08:54:25,261 - Collecting logs from Autoit 10.225.202.77
2018-07-26 08:54:25,847 - Logs collected successfully for instance 10.225.202.77
2018-07-26 08:54:25,847 - Auto-deploy present in config
2018-07-26 08:54:25,848 - .....Executing step 3 of 4.....
2018-07-26 08:54:25,848 - Collecting logs from Autodeploy 10.225.202.64
2018-07-26 08:54:26,454 - Logs collected successfully for instance 10.225.202.64
2018-07-26 08:54:26,454 - .....Executing step 4 of 4.....
2018-07-26 08:54:26,454 - Generating consolidated final tar file
2018-07-26 08:54:26,499 - Log collection script completed. Please collect the tar file
'autocollect_logs.tgz' under /var/log/autocollect/
```

5. Extract the contents of *autocollect\_logs.tgz* file using the following command:

```
tar -zxvf autocollect_logs.tgz
```

The following are the contents of the *autocollect\_logs.tgz* tar file:

- .bz2 files for each AutoDeploy and AutoIT logs
- Two log files — *autocollect.log* for AutoDeploy logs and *autocollect-vnf\_<autovnf ip>.log* for logs collected from AutoVNF and its instances.




---

**Important** In case of failures of any UAS cluster nodes, check the *autocollect-vnf\_<autovnf ip>.log* for the detailed error message.

---

- *uas\_logs\_<autovnf ip>.tgz* — This file comprises tar bundles for the UAS cluster.

When the *uas\_logs\_<autovnf ip>.tgz* tar file is extracted, a *vnf-logs* directory is created with individual sub-directories for *em*, *esc*, and *autovnf* for storing the corresponding collected tar file contents.

If the VNF diagnostic log files are unavailable in the *diags* directory under *em*, you can use **vnf-collect-diags** RPC command from UEM VM to collect the logs. For information on collecting logs through the RPC, see the [Collecting VNF Diagnostic Information, on page 158](#) section.

The logs collected through the RPC are also stored in the *diags* directory under *em* directory.

6. Verify that files have been extracted.

```
ls -lt
```

Example output:

```
total 2488
```

```
-rw-r--r-- 1 root root 1174824 Jul 26 08:54 uas_logs_10.225.202.93.tgz
-rw-r--r-- 1 root root 43113 Jul 26 08:54
auto-it-5571-cups-1-uas-logs-2018-07-26_08.54.25.UTC.tar.bz2
-rw-r--r-- 1 root root 3197 Jul 26 08:54 autocollect-vnf_10.225.202.93.log
-rw-r--r-- 1 root root 47104 Jul 26 08:54
auto-deploy-5571-cups-1-uas-logs-2018-07-26_08.54.26.UTC.tar.bz2
-rw-r--r-- 1 root root 3488 Jul 26 08:54 autocollect.log
-rw-r--r-- 1 root root 1266857 Jul 26 08:54 autocollect_logs.tgz
```

## Collecting VNF Diagnostic Information

The VNF diagnostic information can also be collected through the use of **vnf-collect-diags** RPC command in UEM VM.

To collect the diagnostic information from StarOS device:

1. Log on to the master UEM VM.
2. Access the NCS CLI.

```
sudo -i
ncs_cli -u admin -C
```

3. Execute the following command to collect VNF diagnostic logs from StarOS device.

```
vnf-collect-diags [ correlator <correlator id> vnfd <vnfd-id> ]
```

This command collects the VNF diagnostic log files and makes it available in the `/var/log/em/diags-<correlator id>` directory.

4. Check the per-VNF diagnostic collection status using the following command:

```
show vnf-collect-diags-status
```

Example output:

```
NAME                                STATUS
-----
xyz_autovnf-cups-c-abc  in-progress
```

## Sample Logs

This section provides a few sample log files.

### AutoDeploy Log:

```
2018-08-09 06:40:10,385 - Deployment uas-log-collection: autoit initiated
2018-08-09 06:40:10,385 - Send Deployment notification for: autoit-instance
2018-08-09 06:40:10,385 - Deploy nsd present under given nsd: abc-vnf
2018-08-09 06:40:10,415 - avf nsd, vnfd is autoit, f-autovnf1
2018-08-09 06:40:10,430 - Direct log collection not supported for vdu-type: control-function
2018-08-09 06:40:10,442 - Uas-cluster present in config
2018-08-09 06:40:10,448 - Collecting logs from uas instances
2018-08-09 06:40:10,453 - Fetching Autovnf, Esc and EM details for 10.225.202.64
2018-08-09 06:40:10,459 - Initiating nc session to 10.225.202.64
/usr/lib/python2.7/dist-packages/Crypto/Cipher/blockalgo.py:141: FutureWarning: CTR mode
needs counter parameter, not IV
  self._cipher = factory.new(key, *args, **kwargs)
2018-08-09 06:40:11,091 - VNFR list fetched successfully
```

```

2018-08-09 06:40:11,097 - Instances details retrieved from Autovnf
2018-08-09 06:40:11,333 - Uas ip details retrieved for 10.225.202.64 : {None: ['45.45.45.53',
'45.45.45.30'], 'esc': ['44.44.44.14', '44.44.44.23'], 'autovnf': ['45.45.45.6',
'45.45.45.28'], 'vnf-em': ['45.45.45.11', '45.45.45.13']}
2018-08-09 06:40:11,509 - Removing staged files from Autovnf
2018-08-09 06:40:11,586 - Files removed successfully
2018-08-09 06:40:11,870 - Connected to Autovnf[10.225.202.64]
2018-08-09 06:40:12,117 - Collecting logs from esc 44.44.44.14
2018-08-09 06:40:17,122 - Collecting logs from esc 44.44.44.23
2018-08-09 06:40:22,125 - Collecting logs from autovnf 45.45.45.6
2018-08-09 06:40:23,364 - Logs collected successfully from autovnf 45.45.45.6
2018-08-09 06:40:23,625 - Logs collected successfully from esc 44.44.44.23
2018-08-09 06:40:27,131 - Collecting logs from autovnf 45.45.45.28
2018-08-09 06:40:32,139 - Collecting logs from vnf-em 45.45.45.11
2018-08-09 06:40:32,230 - Logs collected successfully from autovnf 45.45.45.28
2018-08-09 06:40:32,248 - Logs collected successfully from esc 44.44.44.14
2018-08-09 06:40:37,144 - Collecting logs from vnf-em 45.45.45.13
2018-08-09 06:40:38,332 - Logs collected successfully from vnf-em 45.45.45.13
2018-08-09 06:41:10,963 - Logs collected successfully from vnf-em 45.45.45.11
2018-08-09 06:41:11,084 - All threads finished working for uas instance 10.225.202.64
2018-08-09 06:41:11,091 - Generating tar file for uas instance 10.225.202.64
2018-08-09 06:41:11,315 - Copying the tar file to Autodeploy...
2018-08-09 06:41:11,469 - Tar file successfully copied to Autodeploy under
/var/log/autocollect/ directory
2018-08-09 06:41:11,477 - Log collection completed for uas instance 10.225.202.64.
2018-08-09 06:41:11,491 - Log collection script completed. Please collect the tar file
'autocollect_logs.tgz' under /var/log/autocollect/
2018-08-09 06:41:11,498 - Deployment uas-log-collection: autoit succeeded
2018-08-09 06:41:11,521 - Send Deployment notification for: autoit-instance

```

### AutoVNF Log:

```

2018-08-09 06:40:34,474 - Directory /var/log/autocollect/vnf-logs/esc created successfully
to stage files
2018-08-09 06:40:34,474 - Attempting ssh to 44.44.44.14
2018-08-09 06:40:34,641 - Executing script collect_esc_log.sh in 44.44.44.14
2018-08-09 06:40:39,475 - Attempting ssh to 44.44.44.23
2018-08-09 06:40:39,641 - Executing script collect_esc_log.sh in 44.44.44.23
2018-08-09 06:40:44,477 - Directory /var/log/autocollect/vnf-logs/autovnf created successfully
to stage files
2018-08-09 06:40:44,477 - Attempting ssh to 45.45.45.6
2018-08-09 06:40:44,642 - Executing script collect-uas-logs.sh in 45.45.45.6
2018-08-09 06:40:45,199 - Output file to collect in 45.45.45.6 is
autoit-f-autovnf1-rmuruga2-avf-2-uas-logs-2018-08-09_06.40.06.UTC.tar.bz2
2018-08-09 06:40:45,340 - Collecting bz2 file from instance 45.45.45.6
2018-08-09 06:40:45,346 - Logs collected successfully for instance 45.45.45.6
2018-08-09 06:40:45,415 - Output file to collect in 44.44.44.23 is
esc_log_abc-vnf-vnf1-esc-rmuruga2-esc-2_2018-08-09_06.40.15.UTC.tar.bz2
2018-08-09 06:40:45,623 - Collecting bz2 file from instance 44.44.44.23
2018-08-09 06:40:45,629 - Logs collected successfully for instance 44.44.44.23
2018-08-09 06:40:49,483 - Attempting ssh to 45.45.45.28
2018-08-09 06:40:49,652 - Executing script collect-uas-logs.sh in 45.45.45.28
2018-08-09 06:40:53,979 - Output file to collect in 45.45.45.28 is
autoit-f-autovnf1-rmuruga2-avf-1-uas-logs-2018-08-09_06.40.49.UTC.tar.bz2
2018-08-09 06:40:54,090 - Output file to collect in 44.44.44.14 is
esc_log_abc-vnf-vnf1-esc-rmuruga2-esc-1_2018-08-09_06.39.42.UTC.tar.bz2
2018-08-09 06:40:54,118 - Collecting bz2 file from instance 45.45.45.28
2018-08-09 06:40:54,216 - Logs collected successfully for instance 45.45.45.28
2018-08-09 06:40:54,296 - Collecting bz2 file from instance 44.44.44.14
2018-08-09 06:40:54,357 - Logs collected successfully for instance 44.44.44.14
2018-08-09 06:40:54,491 - Attempting to collect vnf diags through Em 45.45.45.11
2018-08-09 06:40:59,490 - Attempting to collect vnf diags through Em 45.45.45.13
2018-08-09 06:40:59,741 - Vnf diags not collected through 45.45.45.13 as it can be collected
only from Master Em instance
2018-08-09 06:40:59,741 - Directory /var/log/autocollect/vnf-logs/em created successfully

```

```

to stage files
2018-08-09 06:40:59,741 - Attempting ssh to 45.45.45.13
2018-08-09 06:40:59,913 - Executing script collect-em-logs.sh in 45.45.45.13
2018-08-09 06:41:00,178 - Output file to collect in 45.45.45.13 is
abc-vnf-vnf1-em-rmuruga2-em1-2-em-logs-2018-08-09_06.40.35.UTC.tar.bz2
2018-08-09 06:41:00,317 - Collecting bz2 file from instance 45.45.45.13
2018-08-09 06:41:00,322 - Logs collected successfully for instance 45.45.45.13
2018-08-09 06:41:25,592 - Vnf-diags logs collected through EM: 45.45.45.11
2018-08-09 06:41:25,592 - Attempting ssh to 45.45.45.11
2018-08-09 06:41:25,724 - Executing script collect-em-logs.sh in 45.45.45.11
2018-08-09 06:41:32,846 - Output file to collect in 45.45.45.11 is
abc-vnf-vnf1-em-rmuruga2-em1-1-em-logs-2018-08-09_06.41.25.UTC.tar.bz2
2018-08-09 06:41:32,993 - Collecting bz2 file from instance 45.45.45.11
2018-08-09 06:41:33,044 - Logs collected successfully for instance 45.45.45.11

```

### Logs for unsupported vnfd

```

2018-08-09 07:00:11,466 - Deployment uas-log-collection: abc-vnf initiated
2018-08-09 07:00:11,481 - Send Deployment notification for: abc-vnf-instance
2018-08-09 07:00:11,496 - Unsupported Log collection. No oper data retrieved for given vnfd:
vpc-up
2018-08-09 07:00:11,502 - Deployment uas-log-collection: abc-vnf failed
2018-08-09 07:00:11,520 - Send Deployment notification for: abc-vnf-instance

```

## Monitoring VNFM Operations



**Note** The Cisco Elastic Services Controller (ESC) is the only VNFM supported in this release.

## Viewing ESC Status

ESC status can be viewed from the ESC command line or by executing a REST API from AutoVNF.

### Monitoring Status Through the ESC Command Line

Log on to the primary ESC VM and execute the following command from the command line:

```
escadm status
```

#### Example command output:

```
0 ESC status=0 ESC Master Healthy
```

### Monitoring Status Through an AutoVNF API

Log on to the master AutoVNF VM and execute the following command:

```
curl -u admin:<password> -k https://<master_vnfm_address>:60000/esc/health
```

#### Example command output:

```
{"message": "ESC services are running.", "status_code": "2000"}
```

Status code and message display information about ESC health conditions as identified in [Table 19: ESC Status Code Messages, on page 161](#). Status codes in the 2000s imply ESC is operational, 5000 status codes imply at least one of the ESC components is not in service.

Table 19: ESC Status Code Messages

Code	Message
2000	ESC services are running
2010	ESC services are running. ESC High-Availability node not reachable.
2020	ESC services are running. One or more VIM services (keystone, nova) not reachable.*
2030	ESC services are running. VIM credentials not provided.
2040	ESC services running. VIM is configured, ESC initializing connection to VIM.
2100	ESC services are running. ESC High-Availability node not reachable. One or more VIM services ( nova ) not reachable
5010	ESC service ESC_MANAGER not running.
5020	ESC service CONFD not running.
5030	ESC service MONA not running.
5040	ESC service VIM_MANAGER not running.
5090	More than one ESC service (confd, mona) not running.**

## Viewing ESC Health

ESC health can be viewed by logging on to the primary ESC VM and executing the following command from the command line:

```
health.sh
```

### Example command output:

```
esc ui is disabled -- skipping status check
esc_monitor start/running, process 840
esc_mona is up and running ...
vimmanager start/running, process 2807

vimmanager start/running, process 2807
esc_confd is started
tomcat6 (pid 2973) is running... [ OK ]
postgresql-9.4 (pid 2726) is running...
ESC service is running...
Active VIM = OPENSTACK
ESC Operation Mode=OPERATION

/opt/cisco/esc/esc_database is a mountpoint
===== ESC HA (MASTER) with DRBD =====
DRBD_ROLE_CHECK=0
MNT_ESC_DATABASE_CHECK=0
VIMMANAGER_RET=0
ESC_CHECK=0
STORAGE_CHECK=0
```

```
ESC_SERVICE_RET=0
MONA_RET=0
ESC_MONITOR_RET=0
=====
ESC HEALTH PASSED
```

## Viewing ESC Logs

ESC logs are available on the VNF VM in the following directory:

```
/var/log/esc/
```

Two levels of logs are available for ESC:

- [ESC Logs, on page 162](#)
- [ESC YANG Logs, on page 163](#)

Refer also to the ESC user documentation for additional information on monitoring and maintaining the software.

## ESC Logs

To collect ESC logs:

1. Log on to the primary VNF VM.
2. Navigate to the scripts directory.  
**cd /opt/cisco/esc/esc-scripts**
3. Launch the *collect-esc-logs.sh* script to collect the logs.  
**sudo ./collect-esc-logs.sh**

### Example log output:

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

```
#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.
```

```
[sudo] password for admin: Creating log tarball:
/var/tmp/esc_log-2017-05-25_18.09.31.UTC.tar.bz2
Creating temporary working directory: /var/tmp/esc_log-2017-05-25_18.09.31.UTC
```

```
Dumping thread status of ESCManager from tomcat pid 2973 to catalina.out
escadm-output.txt
vm_info.txt
esc_version.txt
esc/
esc/vimmanager/
esc/vimmanager/operations_vimmanager.log
esc/vimmanager/vimmanager.log
esc/esc_gc.log.2.current
esc/esc_gc.log.0
esc/escmanager.log
esc/event_escmanager.log
esc/escmanager_tagged.log
```

```

esc/esc_gc.log.1
esc/custom_script/
esc/pgstartup.log
esc/mona/
esc/mona/actions_mona.log
esc/mona/mona_gc.log.0.current
esc/mona/rules_mona.log
esc/mona/mona.log
tar: esc/mona/mona.log: file changed as we read it
esc/confd/
esc/confd/global.data
esc/confd/devel.log
esc/confd/confd.log
esc/confd/browser.log
esc/confd/audit.log
esc/confd/netconf.trace
esc/confd/netconf.log
esc/spy.log
esc/error_escmanager.log
esc/esc_monitor.log
esc/esc_haagent.log
esc/yangesc.log
esc/debug_yangesc.log
esc/esc_confd.log
boot.log
secure
messages
dmesg
tomcat6/
tomcat6/localhost.2017-05-24.log
tomcat6/host-manager.2017-05-24.log
tomcat6/manager.2017-05-24.log
tomcat6/catalina.out
tomcat6/catalina.2017-05-24.log
audit/
audit/audit.log
postgresql/data/pg_log/
postgresql/data/pg_log/postgresql-Thu.log
postgresql/data/pg_log/postgresql-Wed.log
esc-config/esc-config.xml
Warning: tar completed with status: 1

Tarball file: /var/tmp/esc_log-2017-05-25_18.09.31.UTC.tar.bz2
Symbolic link: /tmp/esc_log-2017-05-25_18.09.31.UTC.tar.bz2

```

Suggestions:

1. Transfer the tarball file from the esc vm
2. Remove the tarball and symbolic link (to save ESC disk space):
 

```

sudo rm /var/tmp/esc_log-2017-05-25_18.09.31.UTC.tar.bz2
sudo rm /tmp/esc_log-2017-05-25_18.09.31.UTC.tar.bz2

```
3. Command to list contents of tarball:
 

```

tar jtvf esc_log-2017-05-25_18.09.31.UTC.tar.bz2

```
4. Command to extract from the tarball:
 

```

tar jxf esc_log-2017-05-25_18.09.31.UTC.tar.bz2

```

## ESC YANG Logs

ESC YANG logs are stored in the following file:

**`/var/log/esc/yangesc.log`**

# Monitoring VNF Operations

## Viewing UEM Service Status

1. Log on to the master UEM VM as the user *ubuntu*.

2. Access the NCS CLI.

```
/opt/cisco/usp/packages/nso/ncs-4.1.1/bin/ncs_cli -C -u admin
```

3. Check the NCS state.

```
show ncs-state ha
```

**Example command output:**

```
ncs-state ha mode master
ncs-state ha node-id 3-1501714180
ncs-state ha connected-slave [ 4-1501714262 ]
```

4. Display the health of cluster.

```
show ems
```

**Example command output:**

EM	VNFM			
ID	SLA	SCM	PROXY	VERSION
3	UP	UP	UP	5.7.0
6	UP	UP	UP	5.7.0



### Important

The UEM services will no longer run on the slave UEM to simplify troubleshooting, maintenance, and synchronization related issues.

## Viewing UEM Logs

To collect UEM logs:

1. Navigate to the *scripts* directory.

```
cd /opt/cisco/em-scripts
```

2. Launch the *collect-em-logs.sh* script to collect the logs.

```
sudo ./collect-em-logs.sh
```

**Example log output:**

```
Collecting Zookeeper nodes...
Traceback (most recent call last):
  File "/opt/cisco/em-scripts/zk_dump.py", line 2, in <module>
    from kazoo.client import KazooClient
ImportError: No module named kazoo.client
```

```

Creating log tarball em-logs-2017-05-26_00.37.28.UTC.tar.bz2 ...
em-logs/
em-logs/upstart/
em-logs/upstart/proxy.log
em-logs/upstart/zk.log
em-logs/upstart/ncs.log
em-logs/scm/
em-logs/scm/audit.log.1.gz
em-logs/scm/ncserr.log.1
em-logs/scm/ncs-java-vm.log.2.gz
em-logs/scm/xpath.trace.1.gz
em-logs/scm/ncs-java-vm.log.1.gz
em-logs/scm/xpath.trace.2.gz
em-logs/scm/ncs-java-vm.log
em-logs/scm/ncserr.log.siz
em-logs/scm/xpath.trace
em-logs/scm/audit.log
em-logs/scm/audit.log.2.gz
em-logs/scm/ncserr.log.idx
em-logs/sla/
em-logs/sla/sla-mgr.log
em-logs/sla/sla-system.log
em-logs/zookeeper/
em-logs/zookeeper/zookeeper.out
em-logs/zookeeper/zookeeper.log
em-logs/vnfm-proxy/
em-logs/vnfm-proxy/vnfm-proxy.log

===== Tarball available at: /tmp/em-logs-2017-05-26_00.37.28.UTC.tar.bz2
=====

To extract the tarball, run: "tar jxf /tmp/em-logs-2017-05-26_00.37.28.UTC.tar.bz2"

```

## Viewing UEM Zookeeper Logs

The UEM maintains logs on the Zookeeper process. The logs are located in the following directories:

```

/var/log/em/zookeeper/zookeeper.log
/var/log/em/zookeeper/zookeeper.out

```

For logs from 3rd Zookeeper instance, check the following directories:

```

/var/log/em/zookeeper/arbitrator/zookeeper.log
/var/log/em/zookeeper/arbitrator/zookeeper.out

```

To collect the contents of UEM Zookeeper database, use the following command:

```
sudo ./collect-em-logs.sh -add-zookeeper
```

If "new -add-zookeeper" flag is specified, then by default, zookeeper data is collected in a single file named *zk\_data*.

For a larger deployment, Zookeeper content can be collected in sub-folders. To collect the Zookeeper content in sub-folders, specify "-tree-output" flag as shown in the following command:

```
sudo ./collect-em-logs.sh -add-zookeeper -tree-output
```

When this command is executed, Zookeeper contents are collected under *zk\_data* directory.

## Viewing VNF Information through the Control Function

Information on the VNF deployment can be obtained by executing commands on the Control Function (CF) VNFC. To access the CF CLI:

1. Open an SSH connection to the IP address of the management interface associated with CF1.
2. Press **Enter** to bring up the log in prompt.
3. Enter the username and password.
4. At the Exec mode prompt, enter each of the following commands and observe the results to ensure that the VNF components have been properly deployed according to the desired configuration:

Command	Purpose
<b>show card table</b>	Displays all VM types (e.g. CF, SF, NF, and AF) that have been deployed.
<b>show crash list</b>	Displays software crash events records and associated dump files (minicore, NPU or kernel) for all crashes or a specified crash event. Verify that there are no new or unexpected crashes listed.
<b>show emctrl vdu list</b>	Displays card to VM mappings for the VNF. Each card should have a valid universally unique identifier (UUID).
<b>show ret stats</b>	Displays statistics associated with Recovery Control Task (RCT) events, including migrations, switchovers and shutdowns. RCT statistics are associated with card-to-card session recovery activities.
<b>show session progress</b>	Displays session progress information for the current context filtered by the options specified. Check for any active or new calls before proceeding with a deactivation.
<b>show version verbose</b>	Displays the software version that has been deployed.
<b>show vdu summary</b>	Displays general information pertaining to the virtual descriptor units (VDUs) that have been deployed.
In releases prior to 6.0: <b>show usf vdu all</b>  In 6.0 and later releases: <b>show vnfr vdu all</b>	Displays detailed information for the VDUs that have been deployed for the USF VDU.
In releases prior to 6.0: <b>show usf vdu-group all</b>  In 6.0 and later releases: <b>show vnfr vdu-group all</b>	Displays information for VDU groups pertaining to the USF VNF use case (if deployed).

Command	Purpose
In releases prior to 6.0: <b>show usf network-path all</b>  In 6.0 and later releases: <b>show vnfr network-path all</b>	Displays network path information for USF VNF components (if deployed).
In releases prior to 6.0: <b>show usf service-function-chain all</b>  In 6.0 and later releases: <b>show vnfr service-function-chain all</b>	Displays SFC information for the USF VNF (if deployed).

## Monitoring and Recovering AutoVNF Through AutoIT

AutoIT provides the ability to monitor and auto-recover AutoVNF instances.

This functionality is enabled through configuration of the AutoVNF VNFC(s) at the time of deployment. Once enabled, AutoIT automatically monitors for faults/failures of the AutoVNF VNFC(s) for which the functionality is enabled. If a fault/failure is detected, AutoIT automatically attempts to auto-heal/recover (redeploy) the VNFC(s).



### Important

The Provisioning Network (floating) IP address is required to leverage the health monitoring functionality.

The following parameters must be configured at the VNFC-level:

**Table 20: Health Check Descriptor Parameters**

Parameter	Required	Type	Description
enabled	O	bool	Enable/Disable health monitoring.
probe-frequency	O	uint16	Health Check Frequency in seconds. UAS uses this as health probe time, meaning every polling interval UAS will invoke health check. Default value is 10 seconds.
probe-max-miss	O	uint16	Maximum number of health probe misses before VNFC instance is declared dead. Default value is 6.

Parameter	Required	Type	Description
recovery-type	O	choice string	Recovery type. It can be one of the following: <ul style="list-style-type: none"> <li>• restart: Recovery only by restarting, move the VNFC instance to error after max retries</li> <li>• external: Recovery performed by external entity. No auto-recovery</li> <li>• restart-then-redeploy: Restart the VM on failure. After maximum retries, redeploy the failed VNFC instances.</li> </ul> Default value is restart-then-redeploy.
retry-count	O	uint16	Number of retries to recover the VNFC Instance. Default value is set to restart-then-redeploy.
boot-time	O	uint16	Initial Bootup time for the VNFC. Default value is 300 seconds.
script	O	string	Script to check VNFC health, by default UAS ICMP script will be used.

The above parameters are configured at the VNFC-level within the VNF descriptor information that is part of the deployment network service descriptor (NSD) as shown in the following example configuration:

```
nsd <nsd_name>
...
vnfd <autovnf_vnfd_name>
...
vnfc <autovnf_vnfc_name>
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 6
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
...
```

Refer to the *Cisco Ultra Services Platform NETCONF API Guide* for more information on the use of these and other parameters related to VNF configuration and deployment.

In the event that automatic recovery is not possible, an API is available to manually recover the VNFC(s).

VNFC status can be viewed by executing the **show vnfr** command from AutoIT. Additional details can be found in the transaction logs for the deployment.

To manually recover a failed AutoVNF VNFC, execute the following command:

```
recover nsd-id <nsd_name> vnfd <vnfd_name>
```

## Monitoring and Recovering VNFC Through AutoVNF

The UEM, CF, and SF VNFCs were autorecovered through the VNF (ESC). In these situations, AutoVNF was not informed of these events. With this release, the AutoVNF monitors these VNFC VMs and can

auto-recover them if required. Additionally, the AutoVNF can also monitor the VNFM (ESC) VMs and provide auto-recovery as needed.

This functionality is enabled through configuration of the VNFC(s) at the time of deployment. Once enabled, AutoVNF automatically monitors for faults/failures of the VNFCs for which the functionality is enabled. If a fault/failure is detected, AutoVNF automatically attempts to auto-heal/recover (redeploy) the VNFC(s).



**Important** The Provisioning Network (floating) IP address is required to leverage the health monitoring functionality.

This functionality is currently only supported for the following VNFCs:

- VNFM (ESC)
- UEM
- CF
- SF



**Important** Ultra M Manager sends fault notification when VMs are down and/or recovered.

The following parameters must be configured at the VNFC-level:

**Table 21: Health Check Descriptor Parameters**

Parameter	Required	Type	Description
enabled	O	bool	Enable/disable health monitoring.
probe-frequency	O	uint16	Health Check Frequency in seconds. UAS uses this as health probe time, meaning every polling interval UAS will invoke health check. Default value is 10 seconds.
probe-max-miss	O	uint16	Maximum number of health probe misses before VNFC instance is declared dead. Default value is 6.
recovery-type	O	choice string	Recovery type. It can be one of the following: <ul style="list-style-type: none"> <li>• restart: Recovery only by restarting, move the VNFC instance to error after max retries</li> <li>• external: Recovery performed by external entity. No auto-recovery</li> <li>• restart-then-redeploy: Restart the VM on failure. After maximum retries, redeploy the failed VNFC instances.</li> </ul> Default value is restart-then-redeploy.

Parameter	Required	Type	Description
retry-count	O	uint16	Number of retries to recover the VNFC Instance. Default value is set to restart-then-redeploy.
boot-time	O	uint16	Initial bootup time for the VNFC. Default value is 300 seconds.
script	O	string	Script to check VNFC health, by default UAS ICMP script will be used.

The above parameters are configured at the VNFC-level within the VNF descriptor information that is part of the deployment network service descriptor (NSD) as shown in the following example configuration:

```

nsd <nsd_name>
...
  vnfd <autovnf_vnfd_name>
...
    vnfc <vnfm_vnfc_name>
      health-check enabled
      health-check probe-frequency 10
      health-check probe-max-miss 6
      health-check retry-count 6
      health-check recovery-type restart-then-redeploy
      health-check boot-time 300
...
    vnfc <uem_vnfc_name>
      health-check enabled
      health-check probe-frequency 10
      health-check probe-max-miss 6
      health-check retry-count 6
      health-check recovery-type restart-then-redeploy
      health-check boot-time 300
...
    vnfc <cf_vnfc_name>
      health-check enabled
      health-check probe-frequency 10
      health-check probe-max-miss 6
      health-check retry-count 6
      health-check recovery-type restart-then-redeploy
      health-check boot-time 300
...
    vnfc <sf_vnfc_name>
      health-check enabled
      health-check probe-frequency 10
      health-check probe-max-miss 6
      health-check retry-count 6
      health-check recovery-type restart-then-redeploy
      health-check boot-time 300
...

```

Refer to the *Cisco Ultra Services Platform NETCONF API Guide* for more information on the use of these and other parameters related to VNF configuration and deployment.

In the event that automatic recovery is not possible, an API is available to manually recover the VNFC(s).

VNFC status can be viewed by executing the **show vnfr** command from AutoIT. Additional details can be found in the transaction logs for the deployment.

To manually recover a failed VNFC, execute the following command:

```
recover nsd-id <nsd_name> vnfd <vnfd_name>
```

# Troubleshooting Deactivation Process and Issues

## NOTES:

- The deactivate process is idempotent and can be multiple times and without error. The system will retry to remove any resources that remain.
- If a deactivation fails (a transaction failure occurs), look at the logs on various UAS software components (AutoDeploy, AutoIT, and AutoVNF), VNFM (ESC), and UEM.
- If deactivation has failed, you must ensure that a clean up is performed either using automation tools or manually if necessary.
- Activation must not be reattempted until all of the previous artifacts have been removed.

## Deactivation Fails Due to Communication Errors with AutoVNF

### Problem Description

During the AutoVNF deactivation process, AutoDeploy indicates that it is unable to deactivate the AutoVNF. This is observed through:

- AutoDeploy transaction log
- AutoDeploy upstart log

### Possible Cause(s)

- AutoDeploy is not able to communicate with AutoVNF.

### Action(s) to Take

- Check network connectivity between the AutoDeploy VM and the AutoVNF VIP.
- Check the management and orchestration network.
- Address any connectivity issues.

### Next Steps

- Once connectivity issues are addressed, perform the deactivate procedure again.

## Deactivation Fails Because AutoDeploy Generates an Exception

### Problem Description

AutoDeploy generates an exception error during the deactivation process.

### Possible Cause(s)

- Connectivity issues
- Configuration issues
- OpenStack/VIM specific issues

- Hardware issues

### Action(s) to Take

1. Capture logs from `/var/log/upstart/autodeploy.log` along with exception error message.
2. Log on to AutoIT and collect the logs from `/var/log/upstart/autoit.log` along with the exception message, if any.
3. Log on to VIP of the active (master) AutoVNF VM and perform a cleanup by running the **deactivate** command from there.
  - a. Log on to the AutoVNF VM as the default user, *ubuntu*.
  - b. Switch to the root user.
 

```
sudo su
```
  - c. Enter the ConfD CLI.
 

```
confd_cli -C -u admin
```
  - d. Enter the *admin* user password when prompted.
  - e. Deactivate the deployment.
 

```
deactivate nsd-id <nsd_name>
```
4. Check the last transaction log to verify that the deactivation was successful. (Transactions are auto-sorted by timestamp, so it should be the last one in the list.)

### Example commands and outputs:

#### show transactions

TX ID	TX TYPE	ID	TIMESTAMP	STATUS
-----				
1500605583-055162	vnf-deployment	dep-5-5	2017-07-21T02:53:03.055205-00:00	
deployment-failed	-			
1500606090-581863	vnf-deployment	dep-5-5	2017-07-21T03:01:30.581892-00:00	
deployment-success	-			
1500606127-221084	vnf-deployment	dep-5-5	2017-07-21T03:02:07.221114-00:00	
deployment-success	-			

```
show log 1500606127-221084 | display xml
<config xmlns="http://tail-f.com/ns/config/1.0">
  <log xmlns="http://www.cisco.com/usp/nfv/usp-autovnf-oper">
    <tx-id>1500606127-221084</tx-id>
    <log>2017-07-21 03:02:07,276 - Notify deployment
2017-07-21 03:02:07,297 - Connection to VNFM (esc) at 172.16.181.107
2017-07-21 03:02:07,418 - NETConf Sessions (Transaction/Notifications) established
...

```

5. Manually delete the AutoDeploy VM using the information in [Terminating the AutoDeploy VM, on page 96](#).

### Next Steps

- Open a support case providing all of the log information that was collected.

## Deactivation Fails Because of AutoVNF-VNFM Communication Issues

### Problem Description

During the AutoVNF deactivation process, AutoVNF indicates that it is unable to deactivate the VNFM. This is observed through:

- AutoVNF transaction log
- AutoVNF upstart log

### Possible Cause(s)

- AutoVNF is not able to communicate with the VNFM.

### Action(s) to Take

- Check network connectivity between the master AutoVNF VM and the VNFM VIP.
- Check the management and orchestration network.
- Address any connectivity issues.

### Next Steps

- Once connectivity issues are addressed, perform the deactivate procedure again.

## Deactivation Fails Because of Issue at VNFM

### Problem Description

During the AutoVNF deactivation process, the VNFM returns an error. This is observed through:

- AutoVNF transaction log
- AutoVNF upstart log
- ESC logs

### Possible Cause(s)

- ESC health is not good due to an issue or network connectivity.
- ESC is not able to communicate with the VIM.
- ESC has an internal error.
- AutoVNF is unable to create/delete OpenStack artifacts.

### Action(s) to Take

1. Check `/var/log/esc/yangesc.log` for any issues or error messages.
2. Run `health.sh` to determine the health of ESC.
3. Check network connectivity and address an issues. Retry the deactivation.
4. Check network connectivity with the VIM and address any issues. Retry the deactivation.

- Determine if ESC has a deployment configuration. From the active ESC VM:

```
/opt/cisco/esc/confd/bin/confd_cli -C
show running-config
```

If a configuration is present, most likely ESC is still retrying the deactivation, allow more time for the process to continue.

If no configuration exists, check if there are deployment artifacts still on the VIM. Retry the deactivation.

- Collect logs by running *collect\_esc\_log.sh* from both the active and standby ESC VMs.
- Perform a manual cleanup.




---

**Note** Only artifacts which UAS created need to be removed. Any pre-created artifacts must remain in place.

---

- Login on to the VIM as tenant.
- Remove all VMs.
- Remove all VIP Ports.
- Remove all networks.
- Remove all flavors.
- Remove all volumes.
- Remove all images.
- Remove host-aggregate created as part of automation.

#### Next Steps

- Open a support case providing all of the log information that was collected.

## Deactivation Fails Because AutoVNF Generates an Exception

### Problem Description

AutoVNF generates an exception error during the deactivation process.

### Possible Cause(s)

- Connectivity issues
- Configuration issues
- OpenStack/VIM specific issues
- Hardware issues

### Action(s) to Take

- Collect all logs from */var/log/cisco-uas*.
- Perform a manual cleanup.



---

**Note** Only artifacts which UAS created need to be removed. Any pre-created artifacts can remain in place.

---

- a. Login on to the VIM as tenant.
- b. Remove all VMs.
- c. Remove all VIP Ports.
- d. Remove all networks.
- e. Remove all flavors.
- f. Remove all volumes.
- g. Remove all images.
- h. Remove host-aggregate created as part of automation.

#### Next Steps

- Open a support case providing all of the log information that was collected.

## Troubleshooting UEM Issues

This section contains information on troubleshooting UEM issues.

### UEM VM Stuck in a Boot Loop

#### Problem Description

Processes that normally run on the UEM VM are unable to start and the VM is stuck in a boot-loop.

#### Possible Cause(s)

There is an error with the Zookeeper database keeping the Zookeeper process and other UEM processes from starting. (No other UEM process can be started unless the Zookeeper process has started.)



---

**Important** Release 6.3 onwards, no UEM services will run on the slave UEM.

---

#### Action(s) to Take

1. Check the UEM Zookeeper logs. Refer to [Viewing UEM Zookeeper Logs, on page 165](#).
2. Look for error messages similar to the following:

```
[myid:4] - INFO [main:FileSnap@83] - Reading snapshot
/var/lib/zookeeper/data/version-2/snapshot.5000004ba
[myid:4] - ERROR [main:QuorumPeer@557] - Unable to load database on disk
java.io.EOFException
```

If the above errors exist, proceed to the next step. If not, further debugging is required. Please contact your local support representative.

### 3. Rebuild the Zookeeper database.

- a. Check the health of Master and Slave EM instances. Execute the following commands on each instance.

#### Master UEM VM:

```
sudo -i
ncs_cli -u admin -C
admin connected from 127.0.0.1 using console on deploymentem-1

show ems
EM              VNFM
ID  SLA  SCM  PROXY  VERSION
-----
3   UP   UP   UP     5.7.0
6   UP   UP   UP     5.7.0

exit
```



#### Important

Only the master UEM status may be displayed in the above command because the slave UEM is in the boot loop.

```
show ncs-state ha
ncs-state ha mode master
ncs-state ha node-id 6-1506059686
ncs-state ha connected-slave [ 3-1506059622 ]
```

#### Slave UEM VM:



#### Important

The slave UEM may not be accessible if it is experiencing the boot loop issue.

```
sudo -i
ncs_cli -u admin -C
admin connected from 127.0.0.1 using console on deploymentem-1

show ems
EM              VNFM
ID  SLA  SCM  PROXY  VERSION
-----
3   UP   UP   UP     5.7.0
6   UP   UP   UP     5.7.0

exit

show ncs-state ha
ncs-state ha mode slave
ncs-state ha node-id 3-1506059622
ncs-state ha master-node-id 6-1506059686
```

- b. Log on to the node on which Zookeeper data is corrupted.
- c. Enable the debug mode.

```
/opt/cisco/em-scripts/enable_debug_mode.sh
Disable EM reboot. Enable debug mode
```

- d. Reboot the VM in order to enter the debug mode.
- e. Remove the corrupted data.

```
cd /var/lib/zookeeper/data/
ls
myid version-2 zookeeper_server.pid
```

```
mv version-2 version-2_old
```

Check for corrupted data in the arbiter directory as well.

```
cd /var/lib/zookeeper/arbiter/data
ls
myid version-2 zookeeper_server.pid
```

```
mv version-2 version-2_old
```



### Important

This process removes the Zookeeper database by renaming it for additional debugging/recovery.

- f. Reboot the node instance for it to reconcile and rebuild the Zookeeper database from a healthy UEM instance.

```
reboot
```

- g. Log on to the UEM VM upon reboot.

- h. Validate that the database has been successfully rebuilt on the previously failing UEM node.

```
sudo -i
ncs_cli -u admin -C
admin connected from 127.0.0.1 using console on vnfddeploymentem-0
```

```
show ems
```

```
EM          VNFM
ID  SLA  SCM  PROXY  VERSION
-----
3   UP   UP   UP     5.7.0
6   UP   UP   UP     5.7.0
```

```
show ncs-state ha
```

```
ncs-state ha mode slave
ncs-state ha node-id 3-1506093933
ncs-state ha master-node-id 6-1506093930
```

```
exit
```

```
cd /var/lib/zookeeper/data/
ls
myid version-2 version-2_old zookeeper_server.pid
```

```
cat /var/log/em/zookeeper/zookeeper.log
```

```
<---SNIP--->
2017-09-22 15:25:35,192 [myid:3] - INFO
[QuorumPeer[myid=3]/0:0:0:0:0:0:0:0:2181:Follower@61] - FOLLOWING - LEADER ELECTION
```

```

TOOK - 236
2017-09-22 15:25:35,194 [myid:3] - INFO
[QuorumPeer[myid=3]/0:0:0:0:0:0:0:2181:QuorumPeer$QuorumServer@149] - Resolved
hostname: 30.30.62.6 to address: /30.30.62.6
2017-09-22 15:25:35,211 [myid:3] - INFO
[QuorumPeer[myid=3]/0:0:0:0:0:0:0:2181:Learner@329] - Getting a snapshot from leader
2017-09-22 15:25:35,224 [myid:3] - INFO
[QuorumPeer[myid=3]/0:0:0:0:0:0:0:2181:FileTxnSnapLog@240] - Snapshotting:
0x200000050 to /var/lib/zookeeper/data/version-2/snapshot.200000050
2017-09-22 15:25:37,561 [myid:3] - INFO
[NIOServerCxn.Factory:0.0.0.0/0.0.0.0:2181:NIOServerCnxnFactory@192] - Accepted socket
connection from /30.30.62.15:58011
2017-09-22 15:25:37,650 [myid:3] - WARN
[NIOServerCxn.Factory:0.0.0.0/0.0.0.0:2181:ZooKeeperServer@882] - Connection request
from old client /30.30.62.15:58011; will be dropped if server is in r-o mode
2017-09-22 15:25:37,652 [myid:3] - INFO
[NIOServerCxn.Factory:0.0.0.0/0.0.0.0:2181:ZooKeeperServer@928] - Client attempting
to establish new session at /30.30.62.15:58011
<---SNIP--->

```

Also, check the logs in the `/var/log/em/zookeeper/arbiter` directory.

- i. Disable the UEM debug mode on the VM on which the Zookeeper database was rebuilt.

```

/opt/cisco/em-scripts/disable_debug_mode.sh
Disable debug mode

```

### Next Steps

Open a support case providing all the log information that was collected.



# APPENDIX A

## boot\_uas.py Help

```
./boot_uas.py --help
usage: boot_uas.py [-h] [--version] [--hostname HOSTNAME]
                  [--delete DELETE [DELETE ...]] [--recover RECOVER]
                  [--delete-uas] [--upgrade-uas] [--upgrade UPGRADE] [--json]
                  [--openstack] [--kvm] [--autodeploy] [--autoit] [--autovnf]
                  [--os_auth_url OS_AUTH_URL]
                  [--os_tenant_name OS_TENANT_NAME]
                  [--os_tenant_id OS_TENANT_ID]
                  [--os_project_name OS_PROJECT_NAME]
                  [--os_project_id OS_PROJECT_ID]
                  [--os_project_domain_name OS_PROJECT_DOMAIN_NAME]
                  [--os_project_domain_id OS_PROJECT_DOMAIN_ID]
                  [--os_username OS_USERNAME] [--os_user_id OS_USER_ID]
                  [--os_password OS_PASSWORD]
                  [--os_user_domain_name OS_USER_DOMAIN_NAME]
                  [--os_user_domain_id OS_USER_DOMAIN_ID]
                  [--os_identity_api_version OS_IDENTITY_API_VERSION]
                  [--net NET [NET ...]] [--ip [IPADDR [IPADDR ...]]]
                  [--orch-interface ORCH_INTF] [--gateway DEFAULT_GW]
                  [--gateway_if DEFAULT_GW_IDX] [--ha] [--ha-vip HA_VIP]
                  [--ha-net HA_VIP_NET] [--ha-secret HA_SECRET]
                  [--floating-ip [FLOATING_IP]]
                  [--external-network PUBLICNET] [--flavor FLAVOR]
                  [--avail_zone AVAIL_ZONE] [--image IMAGE]
                  [--ssh_key_file SSH_KEY_FILE] [--password PASSWORD]
                  [--admin ADMIN] [--oper OPER] [--security SECURITY]
                  [--syslog-ip SYSLOG_SERVER_IP] [--port SYSLOG_SERVER_PORT]
                  [--severity SYSLOG_SEVERITY]
```

optional arguments:

-h, --help	show this help message and exit
--version	show program's version number and exit
--hostname HOSTNAME	Hostname prefix
--delete DELETE [DELETE ...]	Delete UAS deployment. Applicable only for OpenStack
--recover RECOVER	Recover UAS deployment.
--delete-uas	Deletes the UAS deployment. Applicable only for KVM
--upgrade-uas	Rolling Upgrade for uas deployment. Applicable only for KVM
--upgrade UPGRADE	Rolling Upgrade for UAS deployment. Applicable only for OpenStack.
--json	Output Data in JSON

Specify the infrastructure to be used for the UAS VM:

--openstack	Use Openstack Infrastructure
--kvm	Use KVM Infrastructure

Specify the type of UAS VM to be instantiated:

```
--autodeploy      Boot AutoDeploy UAS type
--autoit          Boot AutoIT UAS type
--autovnf         Boot AutoVNF UAS type
```

OpenStack configuration to instantiate AutoVNF cluster.

You can either source RC file or provide them on command line:

```
--os_auth_url OS_AUTH_URL
                OS Auth-URL, defaults to env[OS_AUTH_URL].
--os_tenant_name OS_TENANT_NAME
                OS Tenant Name, defaults to env[OS_TENANT_NAME].
--os_tenant_id OS_TENANT_ID
                OS Tenant ID, defaults to env[OS_TENANT_ID].
--os_project_name OS_PROJECT_NAME
                OS Project Name, defaults to env[OS_PROJECT_NAME].
--os_project_id OS_PROJECT_ID
                OS Project ID, defaults to env[OS_PROJECT_ID].
--os_project_domain_name OS_PROJECT_DOMAIN_NAME
                OS Project Domain Name, defaults to
                env[OS_PROJECT_DOMAIN_NAME].
--os_project_domain_id OS_PROJECT_DOMAIN_ID
                OS Project Domain ID, defaults to
                env[OS_PROJECT_DOMAIN_ID].
--os_username OS_USERNAME
                OS Username, defaults to env[OS_USERNAME].
--os_user_id OS_USER_ID
                OS User ID, defaults to env[OS_USER_ID].
--os_password OS_PASSWORD
                OS Password, defaults to env[OS_PASSWORD].
--os_user_domain_name OS_USER_DOMAIN_NAME
                OS User Domain Name, defaults to
                env[OS_USER_DOMAIN_NAME].
--os_user_domain_id OS_USER_DOMAIN_ID
                OS User Domain ID, defaults to env[OS_USER_DOMAIN_ID].
--os_identity_api_version OS_IDENTITY_API_VERSION
                OS Identity API Version, defaults to
                env[OS_IDENTITY_API_VERSION].
```

Networks to be used, first network is used as orchestration:

```
--net NET [NET ...] Ordered list of networks (name or uuid) to attach to
                    AutoVNF Cluster.
--ip [IPADDR [IPADDR ...]]
                    Static IP, default is DHCP
--orch-interface ORCH_INTF
                    Orchestration Interface
```

Default gateway parameters.:

```
--gateway DEFAULT_GW Default Gateway IP Address, needed only in case of
                    static IP
--gateway_if DEFAULT_GW_IDX
                    Interface index to associate default route, default is
                    first interface.
```

High-Availability parameters:

```
--ha              Enable High-Availability
--ha-vip HA_VIP   Virtual IP Address (VIP) for cluster
--ha-net HA_VIP_NET
                    Network used to assign the VIP address
--ha-secret HA_SECRET
                    HA Secret for the cluster
```

Floating IP Parameters:

```
--floating-ip [FLOATING_IP]
                    Enable floating IP association to VIP port
--external-network PUBLICNET
```

External Network to allocate floating IP.

VM specific parameters:

```
--flavor FLAVOR      VM Flavor (name or uuid), default is 'm1.medium'
--avail_zone AVAIL_ZONE
                    The availability zone for AutoVNF placement.
--image IMAGE        Image name or UUID from VIM or location of the qcow2
                    image file
```

AutoVNF VM Login Parameters, if not provided, user will be prompted:

```
--ssh_key_file SSH_KEY_FILE
                    Path to SSH key file to be used as authorised key for
                    login as 'ubuntu'
--password PASSWORD Password for login as 'ubuntu', this is required if
                    SSH key is not provided
```

AutoVNF API Access parameters, if not provided, user will be prompted:

```
--admin ADMIN       Password for AutoVNF admin user.
--oper OPER          Password for AutoVNF oper user
--security SECURITY  Password for AutoVNF security user
```

Syslog Configuration Parameters:

```
--syslog-ip SYSLOG_SERVER_IP
                    Remote syslog server IP address
--port SYSLOG_SERVER_PORT
                    Remote syslog server port
--severity SYSLOG_SEVERITY
                    Severity of syslog
```





## APPENDIX **B**

# Sample VIM Orchestrator and VIM Configuration File

This configuration file dictates the deployment of the VIM Orchestrator (Undercloud) and the VIM (Overcloud). For information on the parameters, see the *Cisco Ultra Services Platform NETCONF API Guide*.

The file below is an example of a combined VIM Orchestrator and VIM configuration file. The configuration in this file brings up a single OSP-D VM on the Ultra M Manager Node within Ultra M deployments based on OSP 10 and that leverage the Hyper-Converged architecture. It also deploys the VIM and related parameters on the Controller, OSP Compute, and Compute nodes.



**Caution** This is only a sample configuration file provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment such as:

- Secure-tokens
- NFVI PoP details
- VIM orchestrator details
- Networking details
- Server flavors and information
- Satellite or CDN server information
- VIM role and node information

```
uas-mode generic
uas-instance vim-vimorch
  external-connection-point 172.21.203.118
  scm                        scm
!
nsd vim-vimorch
  vim-orch  underc
  vim       overc
!
secure-token cimc
  user      admin
  password *****
!
secure-token stack
```

```

user      stack
password *****
!
secure-token ssh-baremetal
user      nfvi
password *****
!
secure-token os-login
user      admin
password *****
!
vimd overc
vim-orch-id underc
nfvi-pop-id sjc-pop
nfvi-nodes node_1
  role vim-compute
!
nfvi-nodes node_2
  role vim-controller
!
nfvi-nodes node_3
  role vim-controller
!
nfvi-nodes node_4
  role vim-osd-compute
!
nfvi-nodes node_5
  role vim-osd-compute
!
nfvi-nodes node_6
  role vim-osd-compute
!
networking dns [ 171.70.168.183 ]
networking ntp 172.24.167.109
networking vlan-pool start 1001
networking vlan-pool end 2000
networking network-types internal-api
ip-prefix 11.120.0.0/24
vlan-id 20
allocation-pool start 11.120.0.10
allocation-pool end 11.120.0.200
!
networking network-types tenant
ip-prefix 11.117.0.0/24
vlan-id 17
allocation-pool start 11.117.0.10
allocation-pool end 11.117.0.200
!
networking network-types storage
ip-prefix 11.118.0.0/24
vlan-id 18
allocation-pool start 11.118.0.10
allocation-pool end 11.118.0.200
!
networking network-types storage-mgmt
ip-prefix 11.119.0.0/24
vlan-id 19
allocation-pool start 11.119.0.10
allocation-pool end 11.119.0.200
!
networking network-types external
ip-prefix 172.21.203.0/24
vlan-id 101
allocation-pool start 172.21.203.125

```

```

allocation-pool end 172.21.203.150
default-route 172.21.203.1
!
!
nfvi-popd sjc-pop
deployment-flavor ucs-1-vnf
nfvi-node node_0
  physical-server-manager ip-address 192.100.1.1
  physical-server-manager login-credential cimc
!
nfvi-node node_1
  physical-server-manager ip-address 192.100.1.2
  physical-server-manager login-credential cimc
!
nfvi-node node_2
  physical-server-manager ip-address 192.100.1.3
  physical-server-manager login-credential cimc
!
nfvi-node node_3
  physical-server-manager ip-address 192.100.1.4
  physical-server-manager login-credential cimc
!
nfvi-node node_4
  physical-server-manager ip-address 192.100.1.5
  physical-server-manager login-credential cimc
!
nfvi-node node_5
  physical-server-manager ip-address 192.100.1.6
  physical-server-manager login-credential cimc
!
nfvi-node node_6
  physical-server-manager ip-address 192.100.1.7
  physical-server-manager login-credential cimc
!
!
vim-orchd underc
hostname          tb3-undercloud
domain-name       cisco.com
dns               [ 171.70.168.183 ]
login-credential  stack
satellite-server ip-address 10.23.252.119
satellite-server hostname rh-satellite
satellite-server domain-name cisco.com
satellite-server organization ultram
satellite-server activation-key "openstack 10"
satellite-server pool-id 8a977cf75c0ca9df015d2ce1cb4d06ab

external-network ip-address 172.21.203.119
external-network netmask 255.255.255.0
provisioning-network ip-address 192.200.0.1
provisioning-network netmask 255.0.0.0
provisioning-network public-vip 192.200.0.2
provisioning-network admin-vip 192.200.0.3
provisioning-network dhcp-ip-range start 192.200.0.101
provisioning-network dhcp-ip-range end 192.200.0.150
provisioning-network inspection-ip-range start 192.200.0.201
provisioning-network inspection-ip-range end 192.200.0.250
network-cidr      192.0.0.0/8
masquerade-network 192.0.0.0/8
nfvi-node pop-id sjc-pop
nfvi-node id      node_0
nfvi-node ip-address 172.21.203.116
nfvi-node login-credential ssh-baremetal
image           /var/cisco/isos/rhel-server-7.3-x86_64-dvd.iso

```

```
flavor vcpus 4
flavor ram 16384
flavor root-disk 100
flavor ephemeral-disk 0
flavor swap-disk 0
!
```



## APPENDIX C

# Sample Tenant Configuration File

### Sample Configuration File for Single Tenant

As described in [Configure VIM Tenants, on page 64](#), tenant configuration parameters are contained in the VIM Artifact Descriptor (vim-artifact).



#### Caution

This is only a sample configuration file provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment.

In 6.3, the multi-tenants are supported per VNF but this feature was not fully qualified. It was made available only for testing purposes. You can define multiple tenants according to the specific needs of your deployment. If the configuration file includes more than one tenant, then multiple VNF instances are deployed in different tenants in a single “site” leveraging a single VIM. See the following section for an example configuration for multi-tenants.

```
nsd nsdl
vim-identity vim_one
vim-artifact vim_one_artifact
!
secure-token cimc
user $8$6FMSIVcieTXcXeCV4V37zQlwnoHIA/xNgQFKLmygLOY=
password $8$y1KOa2J9Qt487Ew8iUxkKbpzzUL3/0Wr0wqxe/KQXdA=
!
secure-token stack
user $8$oeUvS7KN3eThSmvj4S7e9/NT+cxSlfwm2BCRsQKoIcs=
password $8$Ry5Grth2Bc5Vaojx+HgMUVcBPEX69khyC5V8CnkX4LY=
!
secure-token ssh-baremetal
user $8$jwA0KuwOA6a7z9iFoBFGw6Pd+Np6sKhtw/IKX0iFXzc=
password $8$TO5mcj4yPSIpCoQCSG1tcgwg9rZ14WGBavsUnuJ3/5Q=
!
secure-token vim-admin-creds
user $8$WU2Q6jMIKA/C5tozjt/+M3L29heHstM0x68E80RQ6ME=
password $8$prN66xSTFb+5CU4fM2CeJ5BcZnOhxJUibj8/gMxuBMM=
!
secure-token sjc-core
user $8$FwHBMHTaIBE3TZ1cZHb2bizvT0Plej6awh0A2i7yHA=
password $8$OoEwoe7/hJjhjox1SDdJnf81eIO4i+9ZeGIeHlMYNpE=
!
vim vim_one
api-version v2
auth-url http://172.21.203.31:5000/v2.0
user vim-admin-creds
tenant admin
```

```

!
tenantd sjccore
login-credential sjc-core
tenant-role admin
tenant-quota no-of-instances-allowed 10000
tenant-quota no-of-cores-allowed 200000
tenant-quota no-of-injected-files 300000
tenant-quota no-of-injected-files-content-in-bytes 400000
tenant-quota megabytes-of-ram-allowed 500000000
tenant-quota no-of-floating-ips 600
tenant-quota max-no-of-subnets-allowed 700
tenant-quota max-no-of-ports-allowed 8000
tenant-quota no-of-volumes-allowed 900
tenant-quota volumes-allowed-gb 10000
tenant-quota volumes-backup-size-allowed 20000
!
vim-artifactd vim_one_artifact
tenant sjccore
!

```

### Sample Multi-tenant Configuration File



#### Important

In 6.3, the multi-tenant support for VNF was not fully qualified, and was made available only for testing purposes. For more information, contact your Cisco Accounts representative.

You can define the tenants according to the specific needs of your deployment. If the configuration file includes more than one tenant, then multiple VNF instances are deployed in different tenants in a single “site” leveraging a single VIM.



#### Caution

This is only a sample configuration file provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment.

```

uas-instance autoit-sun
external-connection-point 10.105.248.35
scm scm
!
nsd autovnf
vim-identity vim1
vim-artifact vim_artifact_one
vnf-package [ usp_6_3 ]
require [ autoit-sun ]
vld mgmt
vl-type management
network-instance management
!
vld orch
vl-type orchestration
network-instance orchestration
!
vld service
vl-type service
network abc-service
!
vld service1
vl-type service
network abc-service-1
!

```

```
vnfd vnfd-em
  vnf-type          uem
  version           6.3
  high-availability true
  vnfm vnfd vnfd-esc
  cloud-info vnfm vnfd vnfd-esc
  configuration boot-time 1800
  configuration domain-name cisco.com
  configuration set-vim-instance-name true
  configuration dns-server 1.1.1.1
  !
  external-connection-point vnf-em
  connection-point eth0
  floating-ip disabled
  !
vnfc vnf-em
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 6
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id vdu-em
  number-of-instances 2
  connection-point eth0
  virtual-link service-vl mgmt
  !
  connection-point eth1
  virtual-link service-vl orch
  !
  !
  !
vnfd vnfd-esc
  vnf-type          esc
  version           6.3
  high-availability true
  configuration openstack.endpoint publicURL
  configuration secure-login false
  configuration boot-time 1800
  configuration set-vim-instance-name true
  external-connection-point vnf-esc
  connection-point eth0
  floating-ip disabled
  !
vnfc vnf-esc
  health-check disabled
  health-check boot-time 300
  vdu vdu-id vdu-esc
  connection-point eth0
  virtual-link service-vl mgmt
  !
  connection-point eth1
  virtual-link service-vl orch
  !
  !
  !
vnfd vnfd-vpcl
  vnf-type          ugp-standalone
  version           6.3
  high-availability true
  cloud-info vnfm vnfd vnfd-esc
  cloud-info vim vim3
  cloud-info vnf-em vnfd vnfd-em
  configuration internal-network-mtu 1500
```

```

configuration boot-time 1800
configuration domain-name cisco.com
configuration set-vim-instance-name true
configuration dns-server 1.1.1.1
!
vld di_internal1
  network-instance di-internal1
!
vld di_internal2
  network-instance di-internal2
!
external-connection-point vnfc-cf1
  connection-point eth1
  floating-ip disabled
!
vnfc vnfc-cf1
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 6
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id vdu-cf1
  number-of-instances 1
  aggregate-connection-points DI_INTERFACE
    aggregate-connection-point eth0
    !
    aggregate-connection-point eth3
    !
  !
  connection-point eth0
    virtual-link internal-vl di_internal1
  !
  connection-point eth1
    virtual-link service-vl mgmt
  !
  connection-point eth2
    virtual-link service-vl orch
  !
  connection-point eth3
    virtual-link internal-vl di_internal2
  !
!
vnfc vnfc-sf1
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 6
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id vdu-sf1
  number-of-instances 1
  aggregate-connection-points DI_INT
    aggregate-connection-point eth0
    !
    aggregate-connection-point eth3
    !
  !
  connection-point eth0
    virtual-link internal-vl di_internal1
  !
  connection-point eth1
    virtual-link service-vl orch
  !

```

```
connection-point eth2
  virtual-link service-vl service1
!
connection-point eth3
  virtual-link internal-vl di_internal2
!
!
vnfd vnfd-vpc3
  vnf-type      ugp-standalone
  version       6.3
  high-availability true
  cloud-info vnm vnfd vnfd-esc
  cloud-info vim vim2
  cloud-info vnf-em vnfd vnfd-em
  configuration internal-network-mtu 1500
  configuration boot-time 1800
  configuration domain-name cisco.com
  configuration set-vim-instance-name true
  configuration dns-server 1.1.1.1
!
vld di_internal1
  network-instance di-internal1
!
vld di_internal2
  network-instance di-internal2
!
external-connection-point vnfc-cf3
  connection-point eth1
  floating-ip disabled
!
vnfc vnfc-cf3
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 6
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id vdu-cf3
  number-of-instances 1
  aggregate-connection-points DI_INTERFACE
  aggregate-connection-point eth0
  !
  aggregate-connection-point eth3
  !
!
connection-point eth0
  virtual-link internal-vl di_internal1
!
connection-point eth1
  virtual-link service-vl mgmt
!
connection-point eth2
  virtual-link service-vl orch
!
connection-point eth3
  virtual-link internal-vl di_internal2
!
!
vnfc vnfc-sf3
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 6
```

```

health-check recovery-type restart-then-redeploy
health-check boot-time 300
vdu vdu-id vdu-sf3
number-of-instances 1
aggregate-connection-points DI_INT
  aggregate-connection-point eth0
  !
  aggregate-connection-point eth3
  !
  !
connection-point eth0
  virtual-link internal-vl di_internal1
  !
connection-point eth1
  virtual-link service-vl orch
  !
connection-point eth2
  virtual-link service-vl service
  !
connection-point eth3
  virtual-link internal-vl di_internal2
  !
!
!
!
nsd autoit-sun
vim-identity vim1
vim-artifact vim_artifact_one
vnf-package [ usp_6_3 ]
vld mgmt
  vl-type          management
  network-instance management
  !
vld orch
  vl-type          orchestration
  network-instance orchestration
  !
vnfd sun-autovnf
vnf-type          usp-uas
version           6.3
high-availability true
deploy-nsd        autovnf
configuration boot-time 1800
configuration set-vim-instance-name true
external-connection-point avf
  connection-point eth0
  floating-ip enabled
  floating-ip external-network public
  !
vnfc avf
health-check disabled
health-check boot-time 300
vdu vdu-id autovnf
connection-point eth0
  virtual-link service-vl mgmt
  !
connection-point eth1
  virtual-link service-vl orch
  !
!
!
!
secure-token login
user          $8$HjP/ha1KqDij6DCUZEN0RmyOquG5ZGv6wzfgS8Gc9yw=

```

```

password $8$Y0kJt7G9L81QFGbqzTBo424a1wq0Q9tqckytZQI5YDo=
!
secure-token scm-admin
user $8$H4e9LO2wH2+W/x4Dhh7Sm8nWxT2lxi8031e2wN/2fMA=
password $8$KnJiA3at2nfQJk0BEMFDeG+AamCzrE58Fy6ES6lVUfc=
!
secure-token scm-oper
user $8$p9gUigD91WLAAEiVXjZLyHK8Isfg2nJGDwH4YZt7MQ0=
password $8$zDMX8pKBbbXKrc+w67QGt+RnWcz+kG889ojcwwWsKNI=
!
secure-token scm-security
user $8$zye2UX+dvdgMY7oq7Jt5EEgH+XLdWhIfzeZXB4Mp6m8=
password $8$EvcKiLx5SQTvUQKHmW0Innf60EZSuOSgakvXTIsFpKA=
!
secure-token em_login
user $8$OE7NrSx9PMY7GEB2Q4duuIMHC+OykguGTubnHevTeM4=
password $8$dUWpjhV5zmf8aDmFu/M+QR/erkZS21VKQ540sDVLhnw=
!
secure-token staros
user $8$Dlu3t/25edeXfBoB3n+VbJ+laDC+adLHiV6e5CuVLaE=
password $8$TMMWXGFdN14/fUM7MmVMxgwc5tStk/IeIqDQ2fpby6w=
!
secure-token esc_netconf
user $8$qpXG2yQn59+fPOJg3UwDvNSAzmsWyleipVxLg60OjQQ=
password $8$Vb1DSZTpGgx/BNCrFtCLktXBnfKsItJFYAjah6CW7zI=
!
secure-token esc_login
user $8$UyWzOYED01BSrdev9abCnUTtv066fE05m/zDWX16dDg=
password $8$c1HxnO39EkIYeJ4Xsf/bxhxtjgD6cMfRu7QpOWw3fpo=
!
secure-token cf_login
user $8$Jvv7tzTAgYX+xR6R513j+v1ugw4qqWud50wj5CsJYek=
password $8$nG1g/JwIoTWXOKXi/6INeswp9V6PavzopVP2SL626T8=
!
secure-token cimc
user $8$g+55KovPpwPbEiYpt+4U91g9DWCWyxElSeQYy50SFg8=
password $8$iMMdmt9irY6Bk0f5LzuNao3qbgUQMsgVEhazz9PDWzc=
!
secure-token stack
user $8$f64QjkNw+ilAHAvGxp4456jFF/aFJevtqgShUTPd2NY=
password $8$nVn9rkDbPtOGbkxCG5DcK//6yM1iYqjv0DNq0iiiXxm=
!
secure-token ssh-baremetal
user $8$61+cdWMEPftZoNRLS5bMTQbGZ/8MD+wgebQBTWgrzwk=
password $8$VQB8LSuTyE5DzMe3LKZiHSIH/H7AXKldZMYFP1YAS+w=
!
secure-token vim-tenant1-creds
user $8$55owGww93Gu2ChBFUqSnJbU5/5P+/4BKe3Bh7jjWWQQ=
password $8$5481U+r4vjNuIFTwniSKjkdzXxchl1p1gkSJCASFG2F4=
!
secure-token vim-tenant2-creds
user $8$tO57wgc4Mq4IVLkKofwtK+hgo5zz879/141609BJ9+o=
password $8$5iFf3mr3njU9EeNXVRohN5ntrUe8luUapW5gNpMQg5M=
!
secure-token vim-tenant3-creds
user user1
password Cisco@123
!
secure-token os_admin
user $8$io2sCUXjMkW4IuMNw+JUAs81Qlm9E63Hwu9qrmX81Q=
password $8$QgG0xeJsqCF0FJ1Ps7h7tnY3+KJJ1iY7Jy/nFLFaqfI=
!
scm scm
admin scm-admin

```

```

oper      scm-oper
security  scm-security
!
vnf-packaged usp_6_3
location  /home/ubuntu/usp-6_3_M0-1234-MK.iso
validate-signature false
configuration staros
external-url /home/ubuntu/system.cfg
!
!
vdu vdu-esc
vdu-type      cisco-esc
login-credential esc_login
netconf-credential esc_netconf
image vnf-package
vnf-rack      vnf-rack1
vnf-package primary usp_6_3
flavor vcpus  2
flavor ram    4096
flavor root-disk 40
flavor ephemeral-disk 0
flavor swap-disk 0
!
!
vdu vdu-em
vdu-type      element-manager
login-credential em_login
scm           scm
image vnf-package
vnf-rack      vnf-rack1
vnf-package primary usp_6_3
flavor vcpus  2
flavor ram    4096
flavor root-disk 40
flavor ephemeral-disk 0
flavor swap-disk 0
!
!
vdu vdu-cfl
vdu-type      control-function
login-credential cf_login
image vnf-package
vnf-rack      vnf-rack1
vnf-package primary usp_6_3
flavor vcpus  8
flavor ram    16384
flavor root-disk 40
flavor ephemeral-disk 0
flavor swap-disk 0
upp param DI_INTERFACE_VLANID
value 2116
!
!
ned netconf
ned-id      cisco-staros-nc
port-number 830
authentication staros
!
configuration staros_config.txt
apply-at day-zero
package staros
!
volume boot cf-boot
volume storage cf-cdr
!

```

```
!
vdu vdu-cf2
  vdu-type          control-function
  login-credential  cf_login
  image vnf-package
  vnf-rack          vnf-rack1
  vnf-package primary usp_6_3
  flavor vcpus      8
  flavor ram        16384
  flavor root-disk  40
  flavor ephemeral-disk 0
  flavor swap-disk  0
  upp param DI_INTERFACE_VLANID
    value 2116

!

ned netconf
  ned-id            cisco-staros-nc
  port-number       830
  authentication    staros
!
configuration staros_config.txt
  apply-at day-zero
  package staros
!
volume boot cf-boot
volume storage cf-cdr
!
!
vdu vdu-sf1
  vdu-type session-function
  image vnf-package
  vnf-rack vnf-rack1
  vnf-package primary usp_6_3
  flavor vcpus      12
  flavor ram        16384
  flavor root-disk  6144
  flavor ephemeral-disk 0
  flavor swap-disk  0
  upp cores         30
  upp crypto-cores  0
  upp service-mode  vpc
  upp disable-mcdma false
  upp disable-numa  false
  upp param DI_INTERFACE_VLANID
    value 2116
!
  upp param IFTASK_CORES
    value 30
!

!
vdu vdu-sf2
  vdu-type session-function
  image vnf-package
  vnf-rack vnf-rack1
  vnf-package primary usp_6_3
  flavor vcpus      12
  flavor ram        16384
  flavor root-disk  6144
  flavor ephemeral-disk 0
  flavor swap-disk  0
  upp cores         30
```

```

upp crypto-cores 0
upp service-mode vpc
upp disable-mcdma false
upp disable-numa false
upp param DI_INTERFACE_VLANID
  value 2116
!
upp param IFTASK_CORES
  value 30
!
!
vdu autovnf
  vdu-type          automation-service
  login-credential  login
  scm               scm
  image vnf-package
  vnf-rack          vnf-rack1
  vnf-package primary usp_6_3
  flavor vcpus      2
  flavor ram        4096
  flavor root-disk  40
  flavor ephemeral-disk 0
  flavor swap-disk  0
!
vdu vdu-cf3
  vdu-type          control-function
  login-credential  cf_login
  image vnf-package
  vnf-rack          vnf-rack1
  vnf-package primary usp_6_3
  flavor vcpus      8
  flavor ram        16384
  flavor root-disk  40
  flavor ephemeral-disk 0
  flavor swap-disk  0
  upp param DI_INTERFACE_VLANID
    value 2116
!
!
ned netconf
  ned-id          cisco-staros-nc
  port-number     830
  authentication  staros
!
configuration staros_config.txt
  apply-at day-zero
  package staros
!
volume boot cf-boot
volume storage cf-cdr
!
!
vdu vdu-sf3
  vdu-type session-function
  image vnf-package
  vnf-rack vnf-rack1
  vnf-package primary usp_6_3
  flavor vcpus      12
  flavor ram        16384
  flavor root-disk  6144
  flavor ephemeral-disk 0
  flavor swap-disk  0

```

```
upp cores      30
upp crypto-cores 0
upp service-mode vpc
upp disable-mcdma false
upp disable-numa false
upp param DI_INTERFACE_VLANID
  value 2116
!
upp param IFTASK_CORES
  value 30
!

!
volume cf-boot
  type          LUKS
  size          4
  bus           ide
  bootable      true
  preserve-on-upgrade false
!
volume cf-cdr
  type          LUKS
  size          16
  bus           ide
  bootable      false
  preserve-on-upgrade false
!
network abc-service
  ip-prefix 192.168.5.0/24
  type      vlan
  dhcp      true
  gateway   192.168.5.1
!
network abc-service-1
  ip-prefix 192.168.9.0/24
  type      vlan
  dhcp      true
  gateway   192.168.9.1
!
vnf-rackd vnf-rack1
host-aggregate abc-host
  host tblano-compute-0.localdomain
  !
  host tblano-compute-1.localdomain
  !
  host tblano-compute-10.localdomain
  !
  host tblano-compute-11.localdomain
  !
  host tblano-compute-2.localdomain
  !
  host tblano-compute-3.localdomain
  !
  host tblano-compute-4.localdomain
  !
  host tblano-compute-5.localdomain
  !
  host tblano-compute-6.localdomain
  !
  host tblano-compute-7.localdomain
  !
  host tblano-compute-8.localdomain
  !
  host tblano-compute-9.localdomain
```

```
!
!
!
vim vim1
  api-version v2
  auth-url    http://172.21.201.218:5000/v2.0
  user       vim-tenant1-creds
  tenant     tenant1
!
vim vim2
  api-version v2
  auth-url    http://172.21.201.218:5000/v2.0
  user       vim-tenant2-creds
  tenant     tenant2
!
vim vim3
  api-version v2
  auth-url    http://172.21.201.218:5000/v2.0
  user       vim-tenant3-creds
  tenant     tenant3
!
network-instance di-internal1
  ip-prefix 192.168.1.0/24
  type      sriov-flat
  dhcp      true
  vlan-tag  true
  vlan      2116
!
network-instance di-internal2
  ip-prefix 192.168.2.0/24
  type      sriov-flat
  dhcp      true
  vlan-tag  false
!
network-instance management
  ip-prefix 10.177.7.0/24
  type      vlan
  dhcp      true
  gateway   10.177.7.1
!
network-instance orchestration
  ip-prefix 172.177.7.0/24
  type      vlan
  dhcp      true
  gateway   172.177.7.1
!
vim-artifactd vim_artifact_one
  vnf-rack [ vnf-rack1 ]
!
```



## APPENDIX **D**

# Sample VNF Rack and VNF Descriptor Configuration File

As described in [Configure the VNF Rack and the VNF Descriptors, on page 68](#), VNF rack configuration parameters are contained in the VIM Artifact Descriptor (VIM-ArtifactD) while VNF configuration parameters are contained in the VNF descriptor (VNFD).



### Caution

This is only a sample configuration file provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment.

```
uas-mode generic
uas-instance autoit
  external-connection-point 172.21.201.68
  scm scm
!
nsd fremont-autovnf
vim-identity viml
vim-artifact vim_artifact_one
vnf-package [ usp_5_7 ]
require [ autoit ]
vld mgmt
  vl-type management
  network-instance abc-mgmt1
!
vld orch
  vl-type orchestration
  network-instance autoit-abc-orch1
!
vld service
  vl-type service
  network servicel
!
vnfd esc
  vnf-type esc
  version 6.0
  high-availability true
  configuration openstack.endpoint publicURL
  configuration secure-login false
  configuration boot-time 1800
  configuration set-vim-instance-name true
  external-connection-point esc
  aggregate-connection-point eth0
  floating-ip disabled
```

```

!
vnfc esc
  health-check disabled
  health-check boot-time 300
  vdu vdu-id esc
  aggregate-connection-point eth0
    virtual-link service-vl mgmt
  !
  aggregate-connection-point eth1
    virtual-link service-vl orch
  !
!
!
vnfd vpc
  vnf-type          ugp
  version           6.0
  high-availability true
  vnfm vnfd esc
  configuration internal-network-mtu 1500
  configuration boot-time 1800
  configuration domain-name cisco.com
  configuration set-vim-instance-name true
  configuration dns-server 1.1.1.1
  !
  vld di_internal
    network-instance di-internall
  !
  external-connection-point cf
    aggregate-connection-point eth1
    floating-ip disabled
  !
  external-connection-point em
    aggregate-connection-point eth0
    floating-ip disabled
  !
vnfc em
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 6
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id em
  number-of-instances 2
  aggregate-connection-point eth0
    virtual-link service-vl mgmt
  !
  aggregate-connection-point eth1
    virtual-link service-vl orch
  !
!
!
vnfc cf
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 6
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id cf
  number-of-instances 1
  aggregate-connection-points ab
    aggregate-connection-point eth0
  !
  aggregate-connection-point eth1

```

```
!
!
aggregate-connection-point eth0
  virtual-link internal-vl di_internal
!
aggregate-connection-point eth1
  virtual-link service-vl mgmt
!
aggregate-connection-point eth2
  virtual-link service-vl orch
!
!
vnfc sf
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 6
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id sf
  number-of-instances 2
  aggregate-connection-point eth0
    virtual-link internal-vl di_internal
  !
  aggregate-connection-point eth1
    virtual-link service-vl orch
  !
  aggregate-connection-point eth2
    virtual-link service-vl service
  !
!
!
nsd autoit
  vim-identity vim1
  vim-artifact vim_artifact_one
  vnf-package [ usp_5_7 ]
  vld mgmt
    vl-type          management
    network-instance abc-mgmt2
  !
  vld mgmt1
    vl-type          management
    network-instance abc-mgmt1
  !
  vld orch
    vl-type          orchestration
    network-instance abc-orch
  !
  vld orch1
    vl-type orchestration
    network abc-orch1
  !
vnfd autovnf2
  vnf-type          usp-uas
  version           6.0
  high-availability true
  deploy-nsd        fremont-autovnf
  configuration boot-time 1800
  configuration set-vim-instance-name true
  external-connection-point avf1
    aggregate-connection-point eth0
  floating-ip enabled
  floating-ip ip-address 172.21.201.65
```

```

floating-ip external-network public
!
vnfc avf1
health-check disabled
health-check boot-time 300
vdu vdu-id autovnf
aggregate-connection-point eth0
  virtual-link service-vl mgmt1
!
aggregate-connection-point eth1
  virtual-link service-vl orch1
!
!
!
vnfd f-autovnf
vnf-type          usp-uas
version           6.0
high-availability aggregate-bility true
deploy-nsd        bgl26-autovnf
configuration boot-time 1800
configuration set-vim-instance-name true
external-connection-point avf
aggregate-connection-point eth0
floating-ip enabled
floating-ip ip-address 172.21.201.69
floating-ip external-network public
!
vnfc avf
health-check disabled
health-check boot-time 300
vdu vdu-id autovnf
aggregate-connection-point eth0
  virtual-link service-vl mgmt
!
aggregate-connection-point eth1
  virtual-link service-vl orch
!
!
!
!
nsd bgl26-autovnf
vim-identity vim1
vim-artifact vim_artifact_one
vnf-package [ usp_5_7 ]
require [ autoit ]
vld mgmt
vl-type          management
network-instance abc-mgmt2
!
vld orch
vl-type          orchestration
network-instance abc-orch
!
vld service
vl-type service
network servicel
!
vnfd esc
vnf-type          esc
version           6.0
high-availability true
configuration openstack.endpoint publicURL
configuration secure-login false
configuration boot-time 1800

```

```
configuration set-vim-instance-name true
external-connection-point esc
aggregate-connection-point eth0
floating-ip disabled
!
vnfc esc
health-check disabled
health-check boot-time 300
vdu vdu-id esc
aggregate-connection-point eth0
virtual-link service-vl mgmt
!
aggregate-connection-point eth1
virtual-link service-vl orch
!
!
!
vnfd vpc
vnf-type          ugp
version           6.0
high-availability true
vnfm vnfd esc
configuration internal-network-mtu 1500
configuration boot-time 1800
configuration domain-name cisco.com
configuration set-vim-instance-name true
configuration dns-server 1.1.1.1
!
vld di_internal
network-instance di-internal2
!
external-connection-point cf
aggregate-connection-point eth1
floating-ip disabled
!
external-connection-point em
aggregate-connection-point eth0
floating-ip disabled
!
vnfc em
health-check enabled
health-check probe-frequency 10
health-check probe-max-miss 6
health-check retry-count 6
health-check recovery-type restart-then-redeploy
health-check boot-time 300
vdu vdu-id em
number-of-instances 2
aggregate-connection-point eth0
virtual-link service-vl mgmt
!
aggregate-connection-point eth1
virtual-link service-vl orch
!
!
vnfc cf
health-check enabled
health-check probe-frequency 10
health-check probe-max-miss 6
health-check retry-count 6
health-check recovery-type restart-then-redeploy
health-check boot-time 300
vdu vdu-id cf
number-of-instances 1
```

```

aggregate-connection-point eth0
  virtual-link internal-vl di_internal
!
aggregate-connection-point eth1
  virtual-link service-vl mgmt
!
aggregate-connection-point eth2
  virtual-link service-vl orch
!
!
vnfc sf
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 6
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id sf
  number-of-instances 2
  aggregate-connection-point eth0
    virtual-link internal-vl di_internal
  !
  aggregate-connection-point eth1
    virtual-link service-vl orch
  !
  aggregate-connection-point eth2
    virtual-link service-vl service
  !
!
!
!
secure-token login
  user      $8$tD/aXNp7XB9mZPRz9AwU0QOBqEx6UViBsXr7iApIKo=
  password  $8$K7sEXpYcu2s9JkKE+LzaDOWSavKXbMRGY1lWwYzXwUM=
!
secure-token scm-admin
  user      $8$2T5dBLixl7IBriq6t08owbp5e8Qa4WRzpyjdUYxtLA8=
  password  $8$FoFj/y6kTROB45idsPQbTUKh3U1SDLJZFFVsofsUrCog=
!
secure-token scm-oper
  user      $8$Z1zYrUHSdYqEB1qYlgBC67RBVCOpdRQFc5BHCac7raA=
  password  $8$PUngBUVSlGA95Bf3e9hg4o53y8RdmzU6zZYQI+2+Lag=
!
secure-token scm-security
  user      $8$NECfIqKrocobgh4aq5kzbW2MDLy+Ljehf6akGW4w7oU=
  password  $8$YzBR/inUnTxLq9Y95w2KYB4RnwWqx1lqEx1GpunV0ao=
!
secure-token em_login
  user      $8$N9RGEfjhuP15AL5E7okfG/ixBJW59U1FNxoAztDURpI=
  password  $8$bV0tB3TkvQE+wQPr69kb87kDi/2e8N8EupyETgFj3ls=
!
secure-token staros
  user      $8$J+fJxpD6GLwBJvbr26SHQC1K5KBY9QoWPoSeyEGTCJ8=
  password  $8$SUr9lr5br+pkkl6soDBZDiO5Fq6kBr12UNDHbkO8Ek8=
!
secure-token esc_netconf
  user      $8$ei4o1Fuh+SLHXBAyfZYiUB4T4ks3Py4XVCfAGE5XQ38=
  password  $8$Gyk22igyv/RtLI3qNE6n75pl2VGEwNIrhLoeJufMMe0=
!
secure-token esc_login
  user      $8$Vay8ILaZXOvB3/eI2k07jAy1cbaFkzDKdknU4UOYzdY=
  password  $8$Xv4tnGZw2IBsYoy+dOKMANqWubhGr5CZ8Raa0tnRJ9M=
!
secure-token cf_login

```

```

user      $8$Z11UTKyTfsMQSgLwUcsJMUmR4d3u92iMJHpbjHc+F9E=
password $8$lTOGXihOulMnIr5dYaZxkakaZDR3eHlId/2lQrz7NUw=
!
secure-token cimc
user      $8$34TyySpewxUS/bmY2HuIPYDaUm3STAtKb5bq6Xek0BI=
password $8$m5mmAmKbxZ+93H/iqqsMV5M+jLCGmPeUX8XQivk2ncE=
!
secure-token stack
user      $8$62QNFQ807mU5t3yr3BVhbUtQNF/pOyEwbhp3zws0xqE=
password $8$rPNYsenkANegQt0uLzlxjkIiD3SVvQyGcMTOWGnPx88=
!
secure-token ssh-baremetal
user      $8$8D27WujADEbJ8NkxF6V5c72BWSpqbCZR46sQm2irio8=
password $8$mTiooRQcT9ATWX93tYSwp4De7NwQMLBIgad4TnpRNLo=
!
secure-token vim-admin-creds
user      $8$QcFLMEKX5vr+B6fltJwXwk4eIOE4pdhzwoViU0ktWMM=
password $8$9GhEGCCGLRKMNYSlq//NvgCD29BHfW5XLvxsczvQ4ig=
!
secure-token sjc-core
user      $8$f/kleaYoXE80eji30o10EYb+sAmxYuxG+6P9/tFG9eE=
password $8$/h/BNL5lnN2Hro5cH3o1AZSUYlmRjD/ey5FwUJHjTlg=
!
secure-token os_admin
user      $8$LQ3QLGRu1ERpxBlifIfvEj6Y6dio8Zlokq2NXZpHWRU=
password $8$cCd78NeeuGcLnPXLJgV1LH4HaRDPVa/9ZBFK7smIUZA=
!
secure-token oc_admin
user      $8$d5cKBUSsyQGWEiG3mN10Yl0u7b2+ivTaxjRIHnymU=
password $8$s00NJU8Bz7nLfn80zkTZKhx3JVJQqnhG74pOnKiJ2b4=
!
secure-token uc_admin
user      $8$TRLlU1rbO7Y5lgx/i5UksEMhkb9fxKQLy8scQKNjci0=
password "$8$Pp4gMWUjzsmvpZCPLWTjWhP0xyhM/9P3dnUZrwJCoheUuybXvtiUbh9dj17ef9305YKgE28w\nZIolojut23wffa=="
!
scm scm
admin    scm-admin
oper     scm-oper
security scm-security
!
vnf-packaged usp_5_7
location /home/ubuntu/usp-6_0_M0-2896.iso
validate-signature true
configuration staros
  external-url /home/ubuntu/system.cfg
!
!
vnf-packaged usp_5_8
checksum 7e04ea370775ecedf85debe4f43cd576
location /root/usp-5_8_M0-2720.iso
validate-signature true
configuration staros
  external-url /root/system.cfg
!
!
vdu esc
vdu-type cisco-esc
login-credential esc_login
netconf-credential esc_netconf
image vnf-package usp_5_7
vnf-rack vnf-rack1
vnf-package usp_5_7
flavor vcpus 2

```

```

flavor ram      4096
flavor root-disk 40
flavor ephemeral-disk 0
flavor swap-disk 0
flavor host-aggregate rack1
!
vdu em
vdu-type      element-manager
login-credential em_login
scm           scm
image vnf-package usp_5_7
vnf-rack      vnf-rack1
vnf-package   usp_5_7
flavor vcpus  2
flavor ram    4096
flavor root-disk 40
flavor ephemeral-disk 0
flavor swap-disk 0
!
vdu cf
vdu-type      control-function
login-credential cf_login
image vnf-package usp_5_7
vnf-rack      vnf-rack1
vnf-package   usp_5_7
flavor vcpus  8
flavor ram    16384
flavor root-disk 40
flavor ephemeral-disk 0
flavor swap-disk 0
ned netconf
  ned-id      cisco-staros-nc
  port-number 830
  authentication staros
!
configuration staros_config.txt
  apply-at day-zero
  package staros
!
volume boot cf-boot
volume storage cf-cdr
!
!
vdu sf
vdu-type      session-function
image vnf-package usp_5_7
vnf-rack      vnf-rack1
vnf-package   usp_5_7
flavor vcpus  4
flavor ram    16384
flavor root-disk 6
flavor ephemeral-disk 0
flavor swap-disk 0
upp cores     10
upp crypto-cores 0
upp service-mode vpc
upp disable-mcdma false
upp disable-numa false
!
vdu autovnf
vdu-type      automation-service
login-credential login
scm           scm
image vnf-package usp_5_7

```

```
vnf-rack      vnf-rack1
vnf-package   usp_5_7
flavor vcpus  2
flavor ram    4096
flavor root-disk 40
flavor ephemeral-disk 0
flavor swap-disk 0
flavor host-aggregate rack1
!
volume cf-boot
  type          LUKS
  size          4
  bus           ide
  bootable      true
  preserve-on-upgrade false
!
volume cf-cdr
  type          LUKS
  size          16
  bus           ide
  bootable      false
  preserve-on-upgrade false
!
network orchestration
  ip-prefix 13.11.22.0/24
  type      vlan
  dhcp      true
  gateway   13.11.22.1
  router    main
!
network xyz-fmnt
  ip-prefix 13.22.10.0/24
  type      vlan
  dhcp      true
  gateway   13.22.10.1
!
network xyz-vpc-svc
  ip-prefix 22.12.11.0/24
  type      vlan
  dhcp      true
!
network sjc-orch
  ip-prefix 10.2.3.0/24
  type      vlan
  dhcp      true
  gateway   10.2.3.1
!
network servicel
  ip-prefix 15.15.15.0/24
  type      vlan
  dhcp      true
!
network abc-orch1
  ip-prefix 33.33.33.0/24
  type      vlan
  dhcp      true
  gateway   33.33.33.1
!
vnf-rackd vnf-rack1
  host-aggregate rack1
  host tblano-compute-0.localdomain
  !
  host tblano-compute-1.localdomain
  !
```

```

host tblano-compute-10.localdomain
!
host tblano-compute-11.localdomain
!
host tblano-compute-12.localdomain
!
host tblano-compute-13.localdomain
!
host tblano-compute-14.localdomain
!
host tblano-compute-15.localdomain
!
host tblano-compute-2.localdomain
!
host tblano-compute-3.localdomain
!
host tblano-compute-4.localdomain
!
host tblano-compute-5.localdomain
!
host tblano-compute-6.localdomain
!
host tblano-compute-7.localdomain
!
host tblano-compute-8.localdomain
!
host tblano-compute-9.localdomain
!
host tblano-osd-compute-0.localdomain
!
host tblano-osd-compute-1.localdomain
!
host tblano-osd-compute-2.localdomain
!
!
vim vim1
api-version v2
auth-url http://172.21.201.218:5000/v2.0
user vim-admin-creds
tenant test
!
network-instance xyz-autovnf-d-ua-management
ip-prefix 30.30.61.0/24
type vlan
dhcp true
gateway 30.30.61.1
!
network-instance autoit-abc-orch1
ip-prefix 33.33.33.0/24
type vlan
dhcp true
gateway 33.33.33.1
!
network-instance abc-mgmt
ip-prefix 32.32.32.0/24
type vlan
dhcp true
gateway 32.32.32.1
!
network-instance abc-mgmt1
ip-prefix 34.34.34.0/24
type vlan
dhcp true

```

```
    gateway 34.34.34.1
  !
network-instance abc-mgmt2
  ip-prefix 35.35.35.0/24
  type      vlan
  dhcp      true
  gateway   35.35.35.1
  !
network-instance abc-orch
  ip-prefix 31.31.31.0/24
  type      vlan
  dhcp      true
  gateway   31.31.31.1
  !
network-instance di-internal1
  ip-prefix 192.168.1.0/24
  type      sriov-flat
  dhcp      true
  vlan-tag  true
  vlan      2116
  !
network-instance di-internal2
  ip-prefix 192.168.2.0/24
  type      sriov-flat
  dhcp      true
  gateway   192.168.2.1
  vlan-tag  true
  vlan      2111
  !
vim-artifactd vim_artifact_one
  vnf-rack [ vnf-rack1 ]
  !
```





## APPENDIX E

# Sample system.cfg File

```
config
system hostname ugp-saegw
ssh key-gen wait-time 0
cli hidden
tech-support test-commands encrypted password ***
logging filter runtime facility confdmgr level debug critical-info
logging filter runtime facility vnfma level debug critical-info
context local
    administrator $CF_LOGIN_USER password $CF_LOGIN_PASSWORD ftp
    interface LOCAL1
        ip address $CF_VIP_ADDR 255.255.255.0
    #exit
    ip route 0.0.0.0 0.0.0.0 $NICID_1_GATEWAY LOCAL1
    ssh generate key
    server sshd
        subsystem sftp
    #exit
    server confd
        confd-user admin
    #exit
#exit
port ethernet 1/1
    bind interface LOCAL1 local
    no shutdown
#exit
snmp community public read-only
end
```





## APPENDIX F

# Sample ESC VIM Connector Configuration

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <!--represents a vim-->
    <vim_connector>
      <!--unique id for each vim-->
      <id>vim1</id>
      <!--vim type [OPENSTACK|VMWARE_VSPHERE|LIBVIRT|AWS|CSP]-->
      <type>OPENSTACK</type>
      <properties>
        <property>
          <name>os_auth_url</name>
          <value>http://10.84.16.185:5000/v3</value>
        </property>
        <!-- The project name for openstack authentication and authorization -->
        <property>
          <name>os_project_name</name>
          <value>saegw</value>
        </property>
        <!-- The project domain name is only needed for openstack v3 identity api -->
        <property>
          <name>os_project_domain_name</name>
          <value>default</value>
        </property>
        <property>
          <name>os_identity_api_version</name>
          <value>3</value>
        </property>
        <property>
          <name>os_identity_overwrite_endpoint</name>
          <value>http://10.84.16.185:5000/v3</value>
        </property>
      </properties>
    <users>
      <user>
        <id>saegw</id>
        <credentials>
          <properties>
            <property>
              <name>os_password</name>
              <value>*****</value>
            </property>
            <!-- The user domain name is only needed for openstack v3 identity api -->
            <property>
              <name>os_user_domain_name</name>
              <value>default</value>
            </property>
          </properties>
        </credentials>
      </user>
    </users>
  </vim_connectors>
</esc_system_config>
```

```
        </credentials>  
    </user>  
</users>  
</vim_connector>  
</vim_connectors>  
</esc_system_config>
```



## APPENDIX **G**

# Sample AutoVNF VNFM Configuration File

This configuration file provides AutoVNF with the necessary information for communicating with a pre-existing VNFM installation. This is used when deploying VNFs through a stand-alone AutoVNF instance. Refer to [Deploying VNFs Using AutoVNF, on page 73](#).



### Caution

This is only a sample configuration file provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment.

```
nsd vpc
  vnfd pgw
    vnf vnf-instance esc
    vnf vim      default_openstack_vim
  !
!
!
  vnf-instance esc
  vnf-type                esc
  external-connection-point 31.31.31.105
  netconf-credential      esc_nc
!
!
secure-token esc_nc
  user      $8$rGbuPzNgqZwdibSmAZ2pWDkIZT3WxL9sRy7ux4A9BSM=
  password  $8$4xqtIS1MMn+pptMDQhBdUHdxxZwaLtzJOCubIdPbnbQ=
!
vim default_openstack_vim
  api-version v2
  auth-url    http://172.21.201.218:5000/v2.0
  user       openstack
  tenant     tenant1
!
```





## APPENDIX H

# Sample AutoVNF VNF Configuration File

The AutoVNF file includes all the configuration information required to deploy all the VNF components. The AutnuoVNF configuration file dictates the deployment of the AutoVNF. For information on the parameters, see the *Cisco Ultra Services Platform NETCONF API Guide*.

The file below is an example of AutoVNF configuration file to bring up the AutoVNF within Ultra M deployments based on OSP 10 and that leverage the Hyper-Converged architecture.



### Caution

This is only a sample configuration file provided solely for your reference. You must create and modify your own configuration file according to the specific needs of your deployment.

```
-snip-
uas-mode standalone
nsd vpc
  vld mgmt
    vl-type          management
    network-instance mgmt
  !
  vld orch
    vl-type          orchestration
    network-instance orch
  !
  vld svc
    vl-type          service
    network-instance servicel
  !
vnfd pgw
  vnf-type          ugp
  version           6.0
  high-availability true
  configuration internal-network-mtu 1500
  configuration boot-time 1800
  configuration domain-name cisco.com
  configuration set-vim-instance-name true
  configuration dns-server 1.1.1.1
  !
  vld di-internal
    network-instance di-internal2
  !
  external-connection-point cf
    connection-point eth1
    ip-address       32.32.32.201
  !
  external-connection-point em
```

```

    connection-point eth0
    ip-address      31.31.31.110
  !
vnfc em
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 3
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id em
  vdu image em_image
  vdu flavor em_flvor
  number-of-instances 3
  connection-point eth0
    virtual-link service-vl orch
    virtual-link fixed-ip 31.31.31.111
  !
  virtual-link fixed-ip 31.31.31.112
  !
  virtual-link fixed-ip 31.31.31.113
  !
  !
!
vnfc cf
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 3
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id cf
  vdu image cf_image
  vdu flavor cf_flavor
  number-of-instances 1
  volume boot cf-boot-volumes
  volume storage cf-cdr-volumes
  !
  connection-point eth0
    virtual-link internal-vl di-internal
  !
  connection-point eth1
    virtual-link service-vl mgmt
  !
  connection-point eth2
    virtual-link service-vl orch
  !
  !
vnfc sf
  health-check enabled
  health-check probe-frequency 10
  health-check probe-max-miss 6
  health-check retry-count 3
  health-check recovery-type restart-then-redeploy
  health-check boot-time 300
  vdu vdu-id sf
  vdu image sf_image
  vdu flavor sf_flavor
  number-of-instances 2
  connection-point eth0
    virtual-link internal-vl di-internal
  !
  connection-point eth1
    virtual-link service-vl orch

```

```

!
connection-point eth2
virtual-link service-vl svc
!
!
secure-token em_login
user      $8$h2p6wBGvyLyG6PJ+1sLcCblNdYZ0G2Ak/PELpAliuNA=
password  $8$DQpFSWR1PGXCsyY6z23JETYo2eCvpoP4I3htPfcI14Q=
!
secure-token cf_login
user      $8$cZSBwQoEP/iFTjZ8npYtUXiWIrRLoWbnPtDq/2lzFvY=
password  $8$WOnJ0Ug0tdfxsV6lG59QBax2YTmqE7Nzi/ATR757QgU=
!
secure-token scm-admin
user      $8$P35TDpFRer+aQZGoq11B7d9CCy7taqenoveYdEoaOjc=
password  $8$rD300S75m+G3AYds9o86xoV921Y8/OFSIM8aQ06w9+Q=
!
secure-token scm-oper
user      $8$V2KoPEOcwgECEkv8P/UeXtG6P8r4LVWScJZlclsbQVs=
password  $8$osruKvpKho28t+InZL8fKXVgq13eMKmufLF14MOcdC8=
!
secure-token scm-security
user      $8$ab6FCW9tewTeRTAuobg62MHLMySV4+WJ+acPOYhkIOI=
password  $8$g8QqV0KkdLH09K6A7rLDfF16g6Drcc7FFXe8LKzJo8c=
!
secure-token openstack
user      $8$IV7e6WN+92ByeHjUsNUzavctpqYwfJi8w7iphYUkhT4=
password  $8$9WRej/z1HeBDmJbsoUQEhQnP6K2f0w9q/TVz5RAvoD4=
!
scm scm
admin    scm-admin
oper     scm-oper
security scm-security
!
vdu esc
vdu-type      cisco-esc
login-credential  esc_nc
netconf-credential esc_nc
image url none
flavor vcpus  2
flavor ram    4096
flavor root-disk 40
flavor ephemeral-disk 0
flavor swap-disk 0
configuration bootvm
  apply-at    day-zero
  source-url  file:///opt/cisco/usp/bundles/vnfm-bundle/bootvm-3_1_0_116.py
!
!
vdu em
vdu-type      element-manager
login-credential  em_login
scm           scm
image url none
flavor vcpus  2
flavor ram    4096
flavor root-disk 40
flavor ephemeral-disk 0
flavor swap-disk 0
!
vdu cf
vdu-type      control-function
login-credential  cf_login

```

```

image url none
flavor vcpus 8
flavor ram 16384
flavor root-disk 40
flavor ephemeral-disk 0
flavor swap-disk 0
ned netconf
  ned-id cisco-staros-nc
  port-number 830
  authentication cf_login
!
configuration staros_config.txt
  apply-at day-zero
  source-url file:///home/ubuntu/system.cfg
!
!
vdu sf
vdu-type session-function
image url none
flavor vcpus 8
flavor ram 16384
flavor root-disk 16
flavor ephemeral-disk 0
flavor swap-disk 0
upp cores 30
upp crypto-cores 0
upp service-mode vpc
upp disable-mcdma false
upp disable-numa false
!
network orch
ip-prefix 11.11.11.0/24
type vlan
dhcp true
!
volume-instance-group cf-boot-volumes
volume CF1-VOLUME-BOOT
  type LUKS
  bootable true
!
volume CF2-VOLUME-BOOT
  type LUKS
  bootable true
!
!
volume-instance-group cf-cdr-volumes
volume CF1-VOLUME-CDR
  type LUKS
  bootable false
!
volume CF2-VOLUME-CDR
  type LUKS
  bootable false
!
!
network-instance mgmt
ip-prefix 32.32.32.0/24
type vlan
dhcp false
ip-allocation-pool 32.32.32.200 32.32.32.210
!
!
network-instance orch
ip-prefix 31.31.31.0/24

```

```
type      vlan
dhcp      false
ip-allocation-pool 31.31.31.200 31.31.31.210
!
!
network-instance di-internal2
ip-prefix 192.168.2.0/24
type      sriov-flat
dhcp      true
gateway   192.168.2.1
vlan-tag  true
vlan      2111
!
network-instance di_internal_bh
ip-prefix 36.36.36.0/24
type      vlan
dhcp      true
!
network-instance service1
ip-prefix 192.168.3.0/24
type      sriov-flat
dhcp      true
vlan-tag  true
vlan      2111
!
network-instance service_bh
ip-prefix 37.37.37.0/24
type      vlan
dhcp      true
!
-snip-
```





## APPENDIX

# USP KPI Descriptions

- [USP KPI Descriptions, on page 223](#)

## USP KPI Descriptions

[Table 22: Supported KPIs, on page 223](#) lists and describes the key performance indicators supported in this release of the USP.

**Table 22: Supported KPIs**

KPI name	Description	Data type
leaf tx-throughput-bps	Transmit Throughput bit per second	usp:counter64
leaf rx-throughput-bps	Receive Throughput in bits per second	usp:counter64
leaf tx-throughput-pps	Transmit Throughput in Packets per second	usp:counter64
leaf rx-throughput-pps	Receive Throughput in Packets per second	usp:counter64
leaf tx-pkts	Total Number of Transmitted Packets	usp:counter64
leaf rx-pkts 4	Total Number of Received packets	usp:counter64
leaf tx-bytes	Total Number of Transmitted Bytes	usp:counter64
leaf rx-bytes	Total Number of Received Bytes	usp:counter64
leaf tx-pkt-drops	Total Number of transmit packets drop	usp:counter64
leaf tx-byte-drops	Total Number of transmit bytes drops	usp:counter64
leaf rx-pkt-drops	Total Number of receive packets drop	usp:counter64
leaf rx-byte-drops	Total Number of receive bytes drop	usp:counter64
leaf num-of-subscribers	Total number of active subscribers	usp:counter64
leaf num-pcc-rules	Total number of Policy Charging Rules	usp:counter64
leaf num-of-service-functions	Total number of Ultra Service Components (USCs)	type uint16;

<b>KPI name</b>	<b>Description</b>	<b>Data type</b>
leaf flows-per-second	Flow creation rate	type uint16;
leaf cpu-utils	CPU Usage in %	uint16
leaf memory-utils-bytes	Memory usage in bytes	usp:counter64
leaf storage-usage-bytes	Storage usage in bytes	usp:counter64
leaf flow-created	Number of flows created	usp:counter64
leaf flow-terminated	Number of flows terminated	usp:counter64
leaf flow-aged	Number of flows aged out	usp:counter64
leaf flow-hits	FLOW cache hits	usp:counter64
leaf flow-miss	Flow cache misses	usp:counter64



## APPENDIX J

# Backing Up Deployment Information

This chapter provides information on the following topics:

- [Overview, on page 225](#)
- [Identify Component IP Addresses, on page 225](#)
- [Backup Configuration Files, on page 228](#)
- [Backup UAS ConfD Databases, on page 229](#)
- [Collect Logs, on page 230](#)
- [Collect Charging Detail Records, on page 230](#)

## Overview

Prior to performing a deployment deactivation (e.g. as part of an upgrade or downgrade process), it is highly recommended that you make backup copies of key information.

To backup this information:

1. [Identify Component IP Addresses, on page 225.](#)
2. [Backup Configuration Files, on page 228.](#)
3. [Backup UAS ConfD Databases, on page 229.](#)
4. [Collect Logs, on page 230](#)
5. [Collect Charging Detail Records, on page 230.](#)

## Identify Component IP Addresses

To collect the HA-VIP, and floating IP addresses for UAS, ESC, UEM, and CF:

1. Log on to the server on which OSP-D is running.
2. Source the “stack\_namerc-core” file.  

```
source ~/<stack_name>rc-core
```
3. Obtain the floating IP for CF and UEM VMs.  

```
neutron floatingip-list
```

Example command output:

id port_id	fixed_ip_address	floating_ip_address
22936d62-d086-4658-acfc-51b3d8952df6 a9489513-9bec-449b-af8-02487b9fb175	172.168.20.13	10.169.126.155
2fc42615-5254-44ec-af5a-a14440a36812 6a170e76-bb05-4cb9-b09e-e94b8e1ae99c 38a53400-346e-4e12-96b6-989fcad75ab3	172.168.20.5	10.169.126.145
70be87df-97db-4d53-b603-2efb7cfd4a6c		10.169.126.149
72205ae8-c905-4705-a67f-a99cf9078246 a088cc88-3e95-4751-a092-e2f6063d3886	172.168.20.101	10.169.126.154
780e652c-3ee7-47c3-ad25-f1cf17d0c9f6 f186730b-be55-4404-9a5a-84741a8d8032	172.16.182.6	10.169.126.144
871825f2-5d30-4a34-baec-3ba09cf93559 c416552e-5f41-4dbe-bbd1-6a9ef0a39e94	172.168.10.11	10.169.126.140
89c1784d-a8a5-4e91-835c-d4dbff47172e 78afdd69-5f86-48c7-84c0-a10dbff25ea57 89d6c6ac-bf12-45b9-ae52-7fd2a20a2838	172.168.10.13	10.169.126.143
a501bec3-d87f-47de-8e11-f5ce903ea1fe		10.169.126.147
f6ff9566-1514-4d55-b09d-800c19906d9e dde1fe31-e278-443f-bd5e-f434edefe14e	172.168.10.7	10.169.126.146
f8c131b5-a5d6-400e-8936-c407504208da aff0efca-cef0-4852-bdb4-9b1fa5ca373f	172.16.182.11	10.169.126.157
f963b405-3586-4ab2-8815-b76332832e64 2a3ab817-9939-45a9-8774-e062ea74387f	172.168.10.101	10.169.126.142

- Obtain the AutoDeploy address.

```
nova list | grep auto-deploy
```

- Log on to the AutoDeploy VM as the default user, *ubuntu*.

```
ssh ubuntu@<ad_vm_address>
```

- Switch to the *root* user.

```
sudo su
```

- Enter the ConfD CLI.

```
confd_cli -C -u admin
```

- Enter the *admin* user password when prompted.

- Find the deployment details from AutoDeploy:

```
show service-deployment <deployment_name> siter autovnfr
```

Example command output:

```
siter LBUCS002
autovnfr LBPCF100-UAS
  endpoint-info ip-address 10.169.126.141
  endpoint-info port 2022
  status alive
vnfmr LBPCF100-ESC
```

```

endpoint-info ip-address 172.168.10.7
endpoint-info port 830
status alive
vnfr LBPCF100-VNF
status alive
vnf-deploymenttr LBPCF100-DEPLOYMENT
em-endpoint-info ip-address 172.168.10.11
em-endpoint-info port 2022
autovnfr LBPGW100-UAS
endpoint-info ip-address 10.169.126.144
endpoint-info port 2022
status alive
vnfmr LBPGW100-ESC
endpoint-info ip-address 172.168.20.5
endpoint-info port 830
status alive
vnfr LBPGW100-VNF
status alive
vnf-deploymenttr LBPGW100-DEPLOYMENT
em-endpoint-info ip-address 172.168.20.12

```

Record the UAS IP address for each VNF as highlighted in the command output example.

10. Log on to the master AutoVNF VM as the default user, *ubuntu*.

```
ssh ubuntu@<ad_vm_address>
```

11. Switch to the *root* user.

```
sudo su
```

12. Enter the ConfD CLI.

```
confd_cli -C -u admin
```

13. Enter the *admin* user password when prompted.

14. Collect the VIP address for ESC.

In releases prior to 6.0:

```
show autovnf-oper:vnfm
```

Example output:

```

autovnf-oper:vnfm vnfmd
state alive
version 3.1.0.94
transaction-id 1507961257-916914
ha-vip 30.30.62.7
vnfc-instance vnfmd-ESC
compute-host tblano-compute-7.localdomain
interfaces autovnfd-uas-management
ip-address 30.30.61.17
mac-address fa:16:3e:3d:be:31
interfaces autovnfd-uas-orchestration
ip-address 30.30.62.7
mac-address fa:16:3e:68:8e:15

```

In 6.0 and later releases:

```
show vnfr
```

For an example output, see the [Example show vnfr Command Output](#), on page 148.

15. Collect the VIP address for the UEM and CF.

**show autovnf-oper:vip-port**

Example output:

```

vip-port vnf-deployment vnf-deployment
transaction-id 1508009048-329005
port autovnfd-uas-management-30.30.61.103
network autovnfd-uas-management
ha-vip 30.30.61.103
vdu-ref element-manager
port autovnfd-uas-management-30.30.61.104
network autovnfd-uas-management
ha-vip 30.30.61.104
vdu-ref control-function
vip-port vnfmd vnfmd-deployment
transaction-id 1507961257-916914
port vnfmd-ESC-vip
network autovnfd-uas-management
ha-vip 30.30.62.7
vdu-ref esc

```

In 6.0 and later releases:

**show vnfr**

For an example output, see the [Example show vnfr Command Output, on page 148](#).

16. Repeat [15, on page 227](#) for each VNF-UAS.

## Backup Configuration Files

Backing up configuration files involves using SFTP to download copies of these files to a backup directory on a remote server.

**Important**

If SFTP to any of the VMs fails, then remove the respective entry from the *known\_hosts* file under *.ssh* directory and retry.

To backup the configuration files:

1. Create a backup directory, if one does not already exist.
2. SFTP the Day 0 configuration called *system.cfg* from each UGP-based VNF to the backup directory.
3. SFTP the latest Day N configuration file from each UGP-based VNF to the backup directory.

The Day N configuration file specifies the configuration of the various gateway and services deployed on the UGP.

**Important**

UGP-based VNF Day N configuration can also be obtained by logging in to the CF and logging the output of the **show configuration** command. In addition, password information saved in this file is encrypted. Prior to re-applying this configuration to the upgraded/redeployed VNF, you'll need to manually reconfigure the unencrypted passwords in the configuration file.

4. Collect the output of the **show support details** command for each VNF.

- SFTP the latest AutoDeploy configuration file from the AutoDeploy VM to the backup directory.



**Important** You'll need to log in to the AutoDeploy VM using the credentials for the user *ubuntu*.

- SFTP the latest AutoVNF configuration file from the master AutoVNF VM to the backup directory.



**Important** You'll need to log in to the AutoVNF VM using the credentials for the user *ubuntu*.

- SFTP the latest VIM Orchestrator configuration file from the AutoDeploy VM to the backup directory.
- SFTP the latest VIM configuration file from the AutoDeploy VM to the backup directory.

## Backup UAS ConfD Databases

Backing up ConfD databases (CDBs) is done on the UAS software role VMs and involves copying the databased files to a secure location.

### AutoDeploy CDB:

Copy the contents of the `/opt/cisco/usp/uas/confd-6.3.1/var/confd/cdb` directory.

Example directory contents:

```
total 1100
drwxr-xr-x 2 root root 4096 Sep 27 22:27 ./
drwxr-xr-x 8 root root 4096 Sep 27 18:48 ../
-rw-r--r-- 1 root root 10332 Sep 27 22:10 aaa_init.xml
-rw-r--r-- 1 root root 10261 Oct 2 20:20 A.cdb
-rw-r--r-- 1 root root 1086629 Sep 27 22:10 C.cdb
-rw-r--r-- 1 root root 804 Sep 27 22:27 O.cdb
```

### AutoIT CDB

Copy the contents of the `/opt/cisco/usp/uas/confd-6.3.1/var/confd/cdb` directory.

Example directory contents:

```
total 884
drwxr-xr-x 2 root root 4096 Sep 14 18:55 ./
drwxr-xr-x 8 root root 4096 Sep 11 21:56 ../
-rw-r--r-- 1 root root 10234 Sep 12 18:34 aaa_init.xml
-rw-r--r-- 1 root root 7092 Sep 14 18:56 A.cdb
-rw-r--r-- 1 root root 857637 Sep 12 18:34 C.cdb
-rw-r--r-- 1 root root 16363 Sep 14 18:56 O.cdb
```

### AutoVNF

Copy the contents of the `/opt/cisco/usp/uas/confd-6.3.1/var/confd/cdb` directory.

Example directory contents:

```
total 1232
drwxr-xr-x 2 root root 4096 Oct 4 05:39 ./
```

```

drwxr-xr-x 8 root root    4096 Sep 27 18:48 ../
-rw-r--r-- 1 root root   10218 Sep 27 22:22 aaa_init.xml
-rw-r--r-- 1 root root    3789 Sep 27 22:22 A.cdb
-rw-r--r-- 1 root root 1223594 Sep 27 22:22 C.cdb
-rw-r--r-- 1 root root    277 Sep 27 18:48 gilam.xml
-rw-r--r-- 1 root root    2216 Oct  4 05:39 O.cdb
-rw-r--r-- 1 root root    271 Sep 27 18:48 vpc.xml

```

## Collect Logs

Prior to deactivating any part of the deployment, it is recommended that you collect logs from the different components that comprise the USP-based VNF and transfer them to a remote backup server.

- **AutoDeploy Logs:** Refer to [Viewing AutoDeploy Logs, on page 128](#) for information on the logs to collect and their locations.

It is recommended that you copy autodeploy.log to autodeploy\_beforedeactivation.log and then collect logs during de-activation.

- **AutoIT Logs:** Refer to [Viewing AutoIT Logs, on page 132](#) for information on the logs to collect and their locations.

It is recommended that you copy autoit.log to autoit\_beforedeactivation.log and then collect logs during de-activation.

- **AutoVNF Logs:** Refer to [Viewing AutoVNF Logs, on page 137](#) for information on the logs to collect and their locations.

It is recommended that you copy autovnf.log to autovnf\_beforedeactivation.log and then collect logs during de-activation.

- **VNFM (ESC) Logs:** Refer to [Viewing ESC Logs, on page 162](#) for information on the logs to collect and their locations.

- **UEM Logs:** Refer to [Viewing UEM Logs, on page 164](#) for information on the logs to collect and their locations.

## Collect Charging Detail Records

Prior to performing an upgrade or redeployment, it is strongly recommended that you collect or backup copies of all charging detail records (CDRs).

The UGP-based VNF supports the ability to push locally-stored CDRs to a configured collection server based on user-defined intervals or criteria. Refer to the “Configuring CDR Push” section within the “HDD Storage” chapter of the GTPP Interface Administration and Reference. Select the document pertaining to your software version from those available here: <https://www.cisco.com/c/en/us/support/wireless/asr-5000-series/products-installation-and-configuration-guides-list.html>

Prior to initiating the VNF upgrade or redeployment, collect or backup copies of all CDRs using one of these two methods:

- Initiate a manual push of specified CDR files to the configured collection server, OR
- Retrieve CDRs via SFTP

Instructions for using these methods is provided in the GTPP Interface Administration and Reference. Note that additional configuration may be required in order to use these methods.





## APPENDIX **K**

# Example RedHat Network Interface and Bridge Configuration Files

---

- [/etc/sysconfig/network-scripts/ifcfg-eno2](#), on page 233
- [/etc/sysconfig/network-scripts/ifcfg-eno1](#), on page 233
- [/etc/sysconfig/network-scripts/ifcfg-br-ex](#), on page 234
- [/etc/sysconfig/network-scripts/ifcfg-br-ctlplane](#), on page 234

## **/etc/sysconfig/network-scripts/ifcfg-eno2**

```
TYPE=Ethernet
BOOTPROTO=dhcp
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eno2
UUID=ba5aa1e1-c3c9-47ea-8858-e0103f3b9b07
DEVICE=eno2
ONBOOT=yes
BRIDGE=br-ex
NM_CONTROLLED=no
NETMASK=255.255.255.0
GATEWAY=172.25.22.1
```

## **/etc/sysconfig/network-scripts/ifcfg-eno1**

```
TYPE=Ethernet
BOOTPROTO=dhcp
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
```

**/etc/sysconfig/network-scripts/ifcfg-br-ex**

```
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enol
UUID=c8a52d43-2ce7-4a4d-81bd-ca7fce6cebe8
DEVICE=enol
ONBOOT=yes
BRIDGE=br-ctlplane
NM_CONTROLLED=no
```

**/etc/sysconfig/network-scripts/ifcfg-br-ex**

```
DEVICE=br-ex
DEFROUTE=yes
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=static
NM_CONTROLLED=no
DELAY=0
IPADDR=172.25.22.59
NETMASK=255.255.255.0
GATEWAY=172.25.22.1
PREFIX="24"
DNS1="171.70.168.183"
DOMAIN="cisco.com"
IPV4_FAILURE_FATAL="yes"
```

**/etc/sysconfig/network-scripts/ifcfg-br-ctlplane**

```
DEFROUTE=yes
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=static
NM_CONTROLLED=no
DELAY=0
DEVICE=br-ctlplane
```



## APPENDIX L

# Patch Upgrade Procedure

- [VNFM Upgrade Operations, on page 235](#)
- [UAS Upgrade Operations, on page 241](#)
- [UEM Upgrade Operations, on page 248](#)

## VNFM Upgrade Operations

The information provided in this section is applicable only when you upgrade your software to release 6.2 and beyond.



### Important

If you're upgrading to any releases prior to 6.2, follow the upgrade instructions in the [Upgrading the Ultra Services Platform Deployment, on page 89](#) chapter of this guide.



### Caution

Upgrade/redeployment operations are disruptive as they involve terminating VMs for the various components that comprise the deployment. When upgrading UAS software roles, all related data is lost. As such, it is strongly recommended that you backup all files related to the deployment including configuration files, logs, and images before performing the upgrade or redeployment. Refer to [Backing Up Deployment Information, on page 225](#) for more information.



### Important

The process described in this section is supported only with Ultra M deployments based on OSP 10 and that leverage the Hyper-Converged architecture.

## Limitations

The following limitations exist with the VNFM upgrade feature:

- This functionality is only available after upgrading to the 6.2 release.
- The rolling VNFM patch upgrade process can only be used to upgrade to new releases that have a compatible database schema. As new releases become available, Cisco will provide information as to whether or not this functionality can be used to perform the upgrade.

- For Ultra M deployments, AutoDeploy and AutoIT must be upgraded before using this functionality. Upgrading these products will terminate the VNF deployment.
- For stand-alone AutoVNF deployments, AutoVNF must be upgraded before using this functionality. Upgrading these products will terminate the VNF deployment.
- Make sure there are no additional operations running while performing an upgrade/rolling upgrade process.
- Upgrade/rolling upgrade procedure should be done only in a maintenance window.

## Feature Description



### Important

In 6.2, this feature was not fully qualified and was made available only for testing purposes. In 6.3 and later releases, this functionality is fully qualified. For more information, contact your Cisco Accounts representative.

With this release, the ESC-based VNFM can optionally be upgraded as part of a rolling patch upgrade process in order to preserve the operational state of the VNF and UAS deployments.



### Important

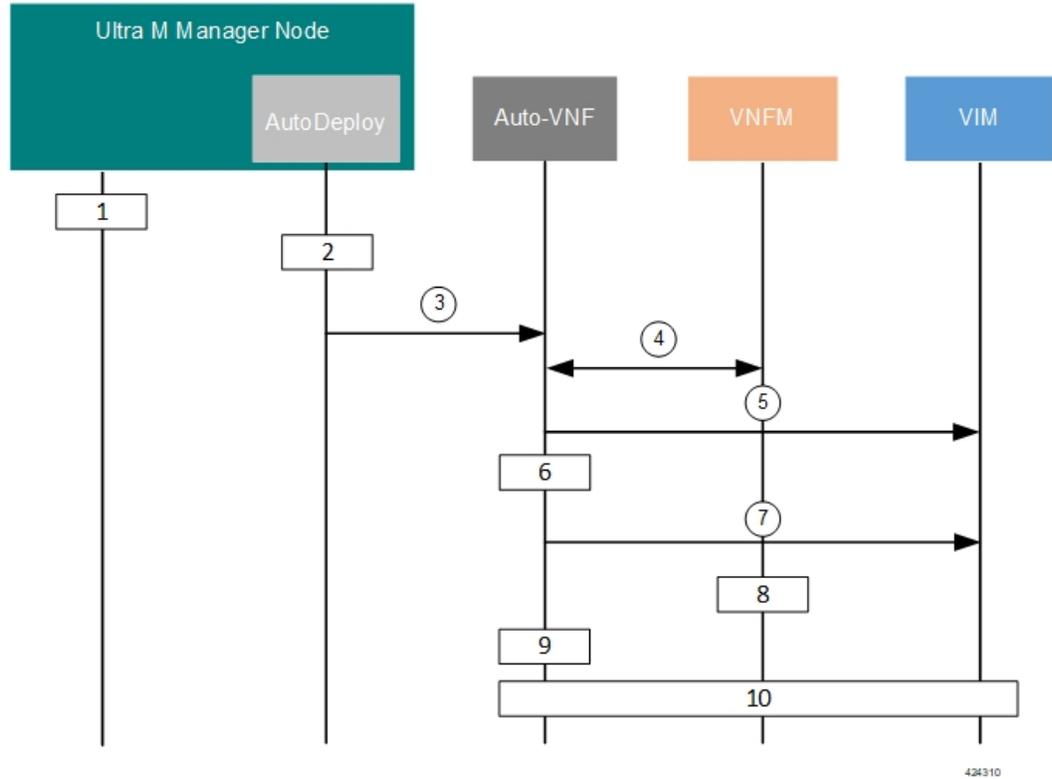
The VNFM upgrade process is supported for Ultra M deployments that leverage the Hyper-Converged architecture and for stand-alone AutoVNF deployments.

## VNFM Upgrade Workflow

This section describes the sequence in which the rolling patch upgrade of VNFM occurs.

[Figure 23: VNFM Upgrade Process Flow, on page 237](#) illustrates the VNFM upgrade process for Ultra M deployments. For stand-alone AutoVNF deployments, the upgrade software image is uploaded to the onboarding server (step 1) and the upgrade command is executed from AutoVNF (step 3).

Figure 23: VNFM Upgrade Process Flow



1. Onboard the new USP ISO containing the VNFM upgrade image to the Ultra M Manager node.
2. Update the deployment network service descriptor (NSD) to identify the new package and onboard it.

```
nsd nsd_name_including_vnfm_vnfd
```

```
vnf-package [ previous_package_descriptor_name upgrade_package_descriptor_name ]
```

Package information is defined in the VNF package descriptor (vnf-packaged) as follows:

```
<---SNIP--->
```

```
vnf-packaged <upgrade_package_descriptor_name>
```

```
  location          <package_url>
```

```
  validate-signature false
```

```
  configuration staros
```

```
    external-url /home/ubuntu/system.cfg
```

```
<---SNIP--->
```

The package must then be referenced in the virtual descriptor unit (VDU) pertaining to the UEM:

```
<---SNIP--->
```

```
vdu esc
```

```
  vdu-type          cisco-esc
```

```
  login-credential esc_login
```

```
  netconf-credential esc_netconf
```

```
  image vnf-package
```

```
  vnf-rack          vnf-rack1
```

```
  vnf-package primary <upgrade_package_descriptor_name>
```

```
  vnf-package secondary <previous_package_descriptor_name>
```

```
...
<---SNIP--->
```

**Important**

The secondary image is used as a fallback in the event an issue is encountered through the upgrade process. If no secondary image is specified, the upgrade process will stop and generate an error log.

3. The rolling upgrade request is triggered through AutoDeploy which initiates the process with AutoVNF.
4. AutoVNF determines which VNFM VM is active and which is standby by communicating with each of the VMs over the management interface.
5. AutoVNF triggers the shutdown of the standby VNFM via the VIM.
6. AutoVNF waits until the VIM confirms that the standby VNFM VM has been completely terminated.
7. AutoVNF initiates the deployment of a new VNFM VM via the VIM using the upgrade image. The VNFM VM is deployed in standby mode.
8. The standby VNFM VM synchronizes data with the active VNFM VM.
9. AutoVNF waits until the VIM confirms that the new VM has been deployed and is in standby mode. If it detects that there is an issue with the VM, AutoVNF re-initiates the VNFM VM with the previous image. If no issues are detected, AutoVNF proceeds with the upgrade process.
10. Repeat the steps [4, on page 238](#) to [7, on page 238](#) for the VNFM VM that is currently active.

## Initiating the VNFM Upgrade

VNFM upgrades are initiated through a remote procedure call (RPC) executed from the ConfD command line interface (CLI) or via a NETCONF API.

### Via the CLI

To perform an upgrade using the CLI, log in to AutoDeploy (Ultra M deployments) or AutoVNF (stand-alone AutoVNF deployments) as the ConfD CLI *admin* user and execute the following command:

```
update-sw nsd-id <nsd_name> rolling { true | false } vnfd <vnfd_name>
vnf-package <pkg_id>
```

**NOTES:**

- <nsd\_name> and <vnfd\_name> are the names of the network service descriptor (NSD) file and VNF descriptor (VNFD) (respectively) in which the VNF component (VNFC) for the VNFM VNF component is defined.
- If the **rolling false** operator is used, the upgrade terminates the entire deployment. In this scenario, the **vnfd<vnfd\_name>** operator should not be included in the command. If it is included, a transaction ID for the upgrade is generated and failed. The AutoVNF upstart log reflects this status.
- <pkg\_id> is the name of the USP ISO containing the upgraded VNFM VM image.
- Ensure that the upgrade package is defined as a VNF package descriptor within the NSD and that it is specified as the primary package in the VNFM VDU configuration.

- Ensure that the current (pre-upgrade) package is specified as the secondary package in the VNFM VDU configuration in order to provide rollback support in the event of errors.

## Via the NETCONF API

**Operation:** nsd:update-sw

**Namespace:** xmlns:nsd="http://www.cisco.com/usp/nfv/usp-nsds"

**Parameters:**

Parameter Name	Required	Type	Description
nsd	M	string	NSD name
rolling	M	boolean	Specifies if the rolling is enabled (true) /disabled (false)
vnfd	M	string	VNFD name, mandatory in case of rolling upgrade
package	M	string	Package descriptor name that should be used to update the vnfd instance mentioned by “vnfd”

### NOTES:

- If the **rolling false** operator is used, the upgrade terminates the entire deployment. In this scenario, the **vnfd<vnfd\_name>** operator should not be included in the command. If it is included, a transaction ID for the upgrade is generated and failed. The AutoVNF upstart log reflects this status.
- Ensure that the upgrade package is defined as a VNF package descriptor within the NSD and that it is specified as the primary package in the VNFM VDU configuration.
- Ensure that the current (pre-upgrade) package is specified as the secondary package in the VNFM VDU configuration in order to provide rollback support in the event of errors.

### Example RPC

```
<nc:rpc message-id="urn:uuid:bac690a2-08af-4c9f-8765-3c907d6e12ba" <nsd
xmlns="http://www.cisco.com/usp/nfv/usp-nsds">
  <nsd-id>fremont-autovnf</nsd-id>
  <vim-identity>vim1</vim-identity>
  <vnfd xmlns="http://www.cisco.com/usp/nfv/usp-nsds">
    <vnfd-id>esc</vnfd-id>
    <vnf-type>esc</vnf-type>
    <version>6.0</version>
    <configuration>
      <boot-time>1800</boot-time>
      <set-vim-instance-name>true</set-vim-instance-name>
    </configuration>
    <external-connection-point>
      <vnfc>esc</vnfc>
      <connection-point>eth0</connection-point>
    </external-connection-point>
```

```

    <high-availability>>true</high-availability>
  <vnfc>
    <vnfc-id>esc</vnfc-id>
    <health-check>
      <enabled>>false</enabled>
    </health-check>
    <vdu>
      <vdu-id>esc</vdu-id>
    </vdu>
    <connection-point>
      <connection-point-id>eth0</connection-point-id>
      <virtual-link>
        <service-vl>mgmt</service-vl>
      </virtual-link>
    </connection-point>
    <connection-point>
      <connection-point-id>eth1</connection-point-id>
      <virtual-link>
        <service-vl>orch</service-vl>
      </virtual-link>
    </connection-point>
  </vnfc>
</vnfd>
</nsd>
<vim xmlns="http://www.cisco.com/usp/nfv/usp-uas-common">
  <vim-id>vim1</vim-id>
  <api-version>v2</api-version>
  <auth-url>http://172.21.201.218:5000/v2.0</auth-url>
  <user>vim-admin-creds</user>
  <tenant>abcxyz</tenant>
</vim>
<secure-token xmlns="http://www.cisco.com/usp/nfv/usp-secure-token">
  <secure-id>vim-admin-creds</secure-id>
  <user>abcxyz</user>
  <password>*****</password>
</secure-token>
<vdu xmlns="http://www.cisco.com/usp/nfv/usp-uas-common">
  <vdu-id>esc</vdu-id>
  <vdu-type>cisco-esc</vdu-type>
  <flavor>
    <vcpus>2</vcpus>
    <ram>4096</ram>
    <root-disk>40</root-disk>
    <ephemeral-disk>0</ephemeral-disk>
    <swap-disk>0</swap-disk>
  </flavor>
  <login-credential>esc_login</login-credential>
  <netconf-credential>esc_netconf</netconf-credential>
  <image>
    <vnf-package>usp_throttle</vnf-package>
  </image>
  <vnf-rack>abcxyz-vnf-rack</vnf-rack>
  <vnf-package>
    <primary>usp_6_2t</primary>
    <secondary>usp_throttle</secondary>
  </vnf-package>
  <volume/>
</vdu>
<secure-token xmlns="http://www.cisco.com/usp/nfv/usp-secure-token">
  <secure-id>esc_login</secure-id>
  <user>admin</user>
  <password>*****</password>
</secure-token>
<secure-token xmlns="http://www.cisco.com/usp/nfv/usp-secure-token">

```

```

    <secure-id>esc_netconf</secure-id>
    <user>admin</user>
    <password>*****</password>
  </secure-token>
  <vnf-packaged xmlns="http://www.cisco.com/usp/nfv/usp-uas-common">
    <vnf-package-id>usp_throttle</vnf-package-id>
    <location>http://192.168.200.61:5000/isos/fremont-autovnf_esp_throttle</location>
    <validate-signature>false</validate-signature>
    <configuration>
      <name>staros</name>
    </configuration>
  </vnf-packaged>
</config>
<external-url>http://192.168.200.61:5000/isos/fremont-autovnf_esp_throttle_staros</external-url>

```

## UAS Upgrade Operations

The information provided in this section is applicable only when you upgrade your software to release 6.2 and beyond.



### Important

If you're upgrading to any releases prior to 6.2, follow the upgrade instructions in the [Upgrading the Ultra Services Platform Deployment, on page 89](#) chapter of this guide.



### Caution

Upgrade/redeployment operations are disruptive as they involve terminating VMs for the various components that comprise the deployment. When upgrading UAS software roles, all related data is lost. As such, it is strongly recommended that you backup all files related to the deployment including configuration files, logs, and images before performing the upgrade or redeployment. Refer to [Backing Up Deployment Information, on page 225](#) for more information.



### Important

The process described in this section is supported only with Ultra M deployments based on OSP 10 and that leverage the Hyper-Converged architecture.

## Feature Description (AutoDeploy and AutoIT)



### Important

In 6.2, this feature was not fully qualified and was made available only for testing purposes. In 6.3 and later releases, this functionality is fully qualified. For more information, contact your Cisco Accounts representative.

With this release, these UAS modules can optionally be upgraded as part of a rolling upgrade process in order to preserve the operational state of the VNF and UAS deployments. The rolling upgrade process is possible as long as the AutoDeploy and AutoIT were deployed in high availability (HA) mode. This allows their CDBs to be synchronized between the active and standby instances.

**Important**

The AutoDeploy and AutoIT rolling upgrade processes are supported for Ultra M deployments that leverage the Hyper-Converged architecture and for stand-alone AutoVNF deployments.

## AutoDeploy and AutoIT Upgrade Workflow

The rolling upgrade process for AutoDeploy and AutoIT occurs as follows:

1. Onboard the new USP ISO containing the UAS upgrade image to the Ultra M Manager node.
2. The rolling upgrade is triggered via a script on baremetal server or undercloud system, wherever the AutoDeploy/AutoIT is deployed.
3. The script terminates the first AutoDeploy or AutoIT VM instance.
4. Upon successful termination of the VM, the script deploys a new VM instance. If it detects that there is an issue with the VM, the script re-initiates the VM with the previous image. If no issues are detected, the script proceeds with the upgrade process.
5. Repeat the steps [3, on page 242](#) and [4, on page 242](#) for the second AutoDeploy or AutoIT VM instance.

**Important**

If AutoDeploy and AutoIT were not deployed with HA mode enabled, or if you prefer to perform an upgrade through a complete reinstall, you must first terminate the current installation using the information and instructions in the *Ultra Services Platform Deployment Automation Guide*.

## Upgrading AutoDeploy or AutoIT

AutoDeploy and AutoIT upgrades are performed by executing a script manually.

1. Log on to the AutoDeploy VM as the *root* user.
2. Initiate the upgrade from another VM:
  - a. Execute the upgrade script:

```
./boot_uas.py --kvm { --autodeploy | --autoit } --upgrade-uas
```
  - b. Enter the password for the user *ubuntu* at the prompt.
  - c. Enter the path and name for the upgrade image at the prompt.
3. Upon completion of the upgrade, check the software version.
  - a. Login to the ConfD CLI as the *admin* user.

```
confd_cli -u admin -C
```
  - b. Enter the *admin* user password when prompted.
  - c. View the status.

```
show uas
```

Example command output:

```

uas version                6.2.0
uas state                  active
uas external-connection-point 172.28.185.132
-----
INSTANCE IP      STATE  ROLE
-----
172.28.185.133  alive  CONFD-MASTER
172.28.185.134  alive  CONFD-SLAVE
-----
NAME              LAST HEARTBEAT
-----
AutoIT-MASTER   2018-03-24 21:24:30
USPCFMWorker    2018-03-24 21:24:30
USPCHEWorker    2018-03-24 21:24:30
USPCWorker      2018-03-24 21:24:30

```

## Limitations

The following limitations exist with the API-based AutoDeploy, AutoIT and AutoVNF upgrade feature:

- This functionality is only available after upgrading to the 6.2 release.
- Regardless of the UAS component (AutoDeploy, AutoIT, or AutoVNF), the rolling patch upgrade process can only be used to upgrade to new releases that have a compatible database schema. As new releases become available, Cisco will provide information as to whether or not this functionality can be used to perform the upgrade.
- For Ultra M deployments, AutoDeploy and AutoIT must be upgraded before using this functionality to upgrade AutoVNF. Upgrading these products will terminate the VNF deployment.
- Make sure there are no additional operations running while performing an upgrade/rolling upgrade process.
- Upgrade/rolling upgrade procedure should be done only in a maintenance window.

## Feature Description (AutoVNF)



### Important

In 6.2, this feature was not fully qualified and was made available only for testing purposes. In 6.3 and later releases, this functionality is fully qualified. For more information, contact your Cisco Accounts representative.

With this release, AutoVNF can optionally be upgraded as part of a rolling upgrade process in order to preserve the operational state of the VNF and UAS deployments.



### Important

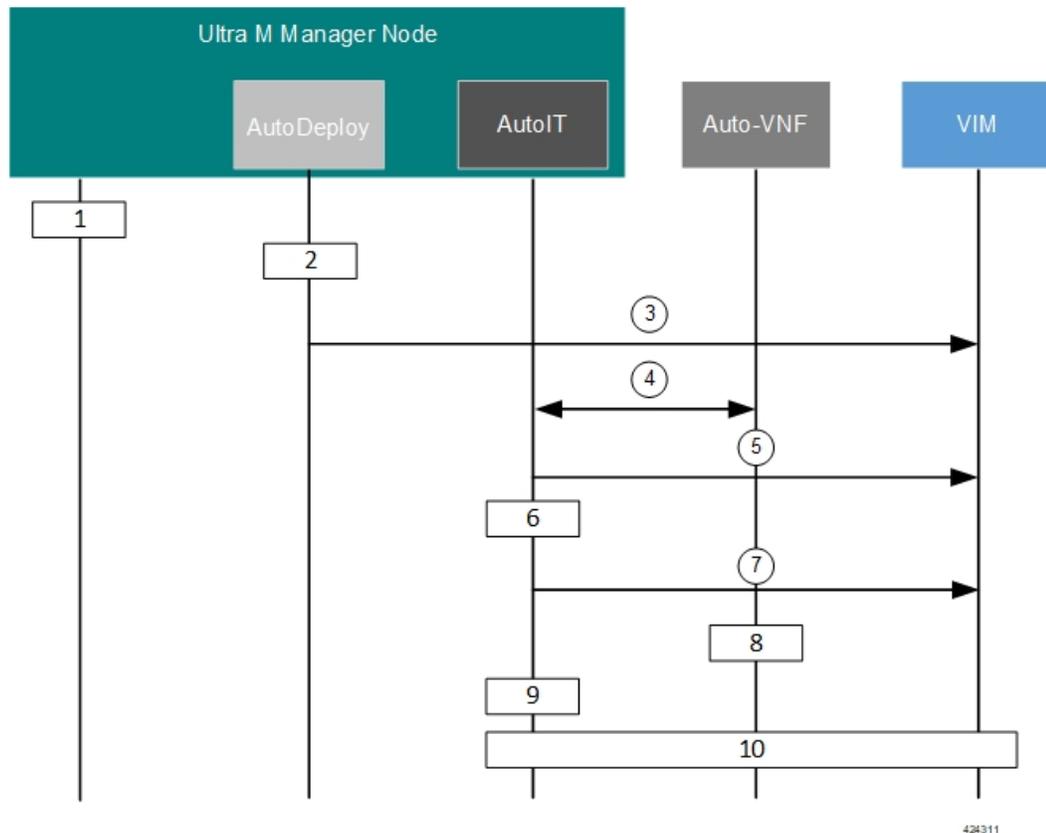
The AutoVNF upgrade process is supported for Ultra M deployments that leverage the Hyper-Converged architecture and for stand-alone AutoVNF deployments.

## AutoVNF Upgrade Workflow

This section describes the sequence in which the AutoVNF upgrade procedure will be performed.

Figure 24: AutoVNF Upgrade Process Flow, on page 244 illustrates the AutoVNF upgrade process for Ultra M deployments. For stand-alone AutoVNF deployments, the upgrade software image is uploaded to the onboarding server (step 1) and the upgrade command is executed from AutoVNF (step 3).

Figure 24: AutoVNF Upgrade Process Flow



1. Onboard the new USP ISO containing the UAS upgrade image to the Ultra M Manager node.
2. Update the deployment network service descriptor (NSD) to identify the new package and onboard it.

```
nsd nsd_name_including_vnfm_vnfd
```

```
vnf-package [ previous_package_descriptor_name upgrade_package_descriptor_name ]
```

Package information is defined in the VNF package descriptor (vnf-packaged) as follows:

```
<---SNIP--->
vnf-packaged <upgrade_package_descriptor_name>
  location <package_url>
  validate-signature false
  configuration staros
  external-url /home/ubuntu/system.cfg
<---SNIP--->
```

The package must then be referenced in the virtual descriptor unit (VDU) pertaining to the UEM:

```

<---SNIP--->
vdu autovnf
  vdu-type automation-service
  login-credential autovnf_login
  scm scm
  image vnf-package
  vnf-rack vnf-rack1
  vnf-package primary <upgrade_package_descriptor_name>
  vnf-package secondary <previous_package_descriptor_name>
  ...
<---SNIP--->

```

**Important**

The secondary image is used as a fallback in the event an issue is encountered through the upgrade process. If no secondary image is specified, the upgrade process will stop and generate an error log.

3. The rolling upgrade request is triggered through AutoDeploy which initiates the process with the VIM through AutoIT.
4. AutoIT determines which AutoVNF VM is active and which is standby by communicating with each of the VMs over the management interface.
5. AutoIT triggers the shutdown of the standby AutoVNF VM via the VIM.
6. AutoIT waits until the VIM confirms that the standby AutoVNF VM has been completely terminated.
7. AutoIT initiates the deployment of a new AutoVNF VM via the VIM using the upgrade image. The AutoVNF VM is deployed in standby mode.
8. The standby AutoVNF VM synchronizes data with the active AutoVNF VM.
9. AutoIT waits until the VIM confirms that the new VM has been deployed and is in standby mode. If it detects that there is an issue with the VM, AutoIT re-initiates the AutoVNF VM with the previous image. If no issues are detected, AutoIT proceeds with the upgrade process.
10. Repeat the steps [4, on page 245](#) to [7, on page 245](#) for the AutoVNF VM that is currently active.

## Initiating the AutoVNF Upgrade

AutoVNF upgrades are initiated through a remote procedure call (RPC) executed from the ConfD command line interface (CLI) or via a NETCONF API.

### Via the CLI

To perform an upgrade using the CLI, log in to AutoDeploy (Ultra M deployments) or AutoVNF (stand-alone AutoVNF deployments) as the ConfD CLI *admin* user and execute the following command:

```

update-sw nsd-id <nsd_name> rolling { true | false } vnfd <vnfd_name>
vnf-package <pkg_id>

```

**NOTES:**

- <nsd\_name> and <vnfd\_name> are the names of the network service descriptor (NSD) file and VNF descriptor (VNFD) (respectively) in which the VNF component (VNFC) for the VNF component is defined.

- If the **rolling false** operator is used, the upgrade terminates the entire deployment. In this scenario, the **vnfd**<vnfd\_name> operator should not be included in the command. If it is included, a transaction ID for the upgrade is generated and failed. The AutoVNF upstart log reflects this status.
- <pkg\_id> is the name of the USP ISO containing the upgraded VNFM VM image.
- Ensure that the upgrade package is defined as a VNF package descriptor within the NSD and that it is specified as the primary package in the AutoVNF VDU configuration.
- Ensure that the current (pre-upgrade) package is specified as the secondary package in the AutoVNF VDU configuration in order to provide rollback support in the event of errors.

## Via the NETCONF API

**Operation:** nsd:update-sw

**Namespace:** xmlns:nsd="http://www.cisco.com/usp/nfv/usp-nsds"

**Parameters:**

Parameter Name	Required	Type	Description
nsd	M	string	NSD name
rolling	M	boolean	Specifies if the rolling is enabled (true) /disabled (false)
vnfd	M	string	VNFD name, mandatory in case of rolling upgrade
package	M	string	Package descriptor name that should be used to update the vnfd instance mentioned by “vnfd”

### NOTES:

- If the **rolling false** operator is used, the upgrade terminates the entire deployment. In this scenario, the **vnfd**<vnfd\_name> operator should not be included in the command. If it is included, a transaction ID for the upgrade is generated and failed. The AutoVNF upstart log reflects this status.
- Ensure that the upgrade package is defined as a VNF package descriptor within the NSD and that it is specified as the primary package in the AutoVNF VDU configuration.
- Ensure that the current (pre-upgrade) package is specified as the secondary package in the AutoVNF VDU configuration in order to provide rollback support in the event of errors.

### Example RPC

```
<nc:rpc message-id="urn:uuid:bac690a2-08af-4c9f-8765-3c907d6e12ba" <nsd
xmlns="http://www.cisco.com/usp/nfv/usp-nsds">
  <nsd-id>fremont-autovnf</nsd-id>
  <vim-identity>vim1</vim-identity>
  <vnfd xmlns="http://www.cisco.com/usp/nfv/usp-nsds">
    <vnfd-id>esc</vnfd-id>
```

```

<vnf-type>esc</vnf-type>
<version>6.0</version>
<configuration>
  <boot-time>1800</boot-time>
  <set-vim-instance-name>true</set-vim-instance-name>
</configuration>
<external-connection-point>
  <vnfc>esc</vnfc>
  <connection-point>eth0</connection-point>
</external-connection-point>
<high-availability>true</high-availability>
<vnfc>
  <vnfc-id>esc</vnfc-id>
  <health-check>
    <enabled>>false</enabled>
  </health-check>
  <vdu>
    <vdu-id>esc</vdu-id>
  </vdu>
  <connection-point>
    <connection-point-id>eth0</connection-point-id>
    <virtual-link>
      <service-vl>mgmt</service-vl>
    </virtual-link>
  </connection-point>
  <connection-point>
    <connection-point-id>eth1</connection-point-id>
    <virtual-link>
      <service-vl>orch</service-vl>
    </virtual-link>
  </connection-point>
</vnfc>
</vnfd>
</nsd>
<vim xmlns="http://www.cisco.com/usp/nfv/usp-uas-common">
  <vim-id>vim1</vim-id>
  <api-version>v2</api-version>
  <auth-url>http://172.21.201.218:5000/v2.0</auth-url>
  <user>vim-admin-creds</user>
  <tenant>abcxyz</tenant>
</vim>
<secure-token xmlns="http://www.cisco.com/usp/nfv/usp-secure-token">
  <secure-id>vim-admin-creds</secure-id>
  <user>abcxyz</user>
  <password>*****</password>
</secure-token>
<vdu xmlns="http://www.cisco.com/usp/nfv/usp-uas-common">
  <vdu-id>esc</vdu-id>
  <vdu-type>cisco-esc</vdu-type>
  <flavor>
    <vcpus>2</vcpus>
    <ram>4096</ram>
    <root-disk>40</root-disk>
    <ephemeral-disk>0</ephemeral-disk>
    <swap-disk>0</swap-disk>
  </flavor>
  <login-credential>esc_login</login-credential>
  <netconf-credential>esc_netconf</netconf-credential>
  <image>
    <vnf-package>usp_throttle</vnf-package>
  </image>
  <vnf-rack>abcxyz-vnf-rack</vnf-rack>
  <vnf-package>
    <primary>usp_6_2t</primary>
  </vnf-package>
</vdu>

```

```

        <secondary>usp_throttle</secondary>
    </vnf-package>
    <volume/>
</vdu>
<secure-token xmlns="http://www.cisco.com/usp/nfv/usp-secure-token">
    <secure-id>esc_login</secure-id>
    <user>admin</user>
    <password>*****</password>
</secure-token>
<secure-token xmlns="http://www.cisco.com/usp/nfv/usp-secure-token">
    <secure-id>esc_netconf</secure-id>
    <user>admin</user>
    <password>*****</password>
</secure-token>
<vnf-packaged xmlns="http://www.cisco.com/usp/nfv/usp-uas-common">
    <vnf-package-id>usp_throttle</vnf-package-id>
    <location>http://192.168.200.61:5000/isos/fremont-autovnf_esp_throttle</location>
    <validate-signature>false</validate-signature>
    <configuration>
        <name>staros</name>

<external-url>http://192.168.200.61:5000/isos/fremont-autovnf_esp_throttle_staros</external-url>

    </configuration>
</vnf-packaged>
</config>

```

## UEM Upgrade Operations

The information provided in this section is applicable only when you upgrade your software to release 6.2 and beyond.



### Important

If you're upgrading to any releases prior to 6.2, follow the upgrade instructions in the [Upgrading the Ultra Services Platform Deployment, on page 89](#) chapter of this guide.



### Caution

Upgrade/redeployment operations are disruptive as they involve terminating VMs for the UEM, CF, and SF components that comprise the VNF. It is strongly recommended that you backup all files related to the deployment including configuration files, logs, and images before performing the upgrade or redeployment. Refer to [Backing Up Deployment Information, on page 225](#) for more information.



### Important

The process described in this section is supported only with Ultra M deployments based on OSP 10 and that leverage the Hyper-Converged architecture.

## Limitations

The following limitations exist with the UEM upgrade feature:

- This functionality is only available after upgrading to the 6.2 release.

- The rolling UEM patch upgrade process can only be used to upgrade to new releases that have a compatible database schema. As new releases become available, Cisco will provide information as to whether or not this functionality can be used to perform the upgrade.
- For Ultra M deployments, AutoDeploy and AutoIT must be upgraded before using this functionality. Upgrading these products will terminate the VNF deployment.
- For stand-alone AutoVNF deployments, AutoVNF must be upgraded before using this functionality. Upgrading these products will terminate the VNF deployment.
- Make sure there are no additional operations running while performing an upgrade/rolling upgrade process.
- Upgrade/rolling upgrade procedure should be done only in a maintenance window.

## Feature Description



### Important

In 6.2, this feature was not fully qualified and was made available only for testing purposes. In 6.3 and later releases, this functionality is fully qualified. For more information, contact your Cisco Accounts representative.

With this release, the UEM can optionally be upgraded as part of a rolling patch upgrade process in order to preserve the operational state of the VNF, UAS, and VNFM deployments.

## UEM Upgrade Workflow

The upgrade flow depends on the number of VM instances in UEM cluster.

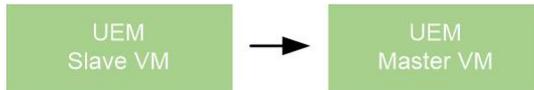
If there are two VM instances in the UEM cluster then UEM Master VM acts also as UEM Standby VM. That is, one instance plays the role of both Master and Standby, while the other instance acts as Slave.

If there are three VM instances in the UEM cluster then there are dedicated VM instances for each role: Master, Slave and Standby. That is, one instance acts as Master, the 2nd instance acts as Slave and the 3rd instance acts as Standby.

In the rolling patch upgrade process, each of the VMs in the UEM Zookeeper cluster is upgraded one at a time. By default, the upgrade attempts to upgrade the slave VM first and the Zookeeper-elected leader VM last as illustrated in [Figure 27: UEM Patch Upgrade Process Flow, on page 251](#).

**Figure 25: UEM VM Upgrade Order for 2 VM based UEM Cluster**

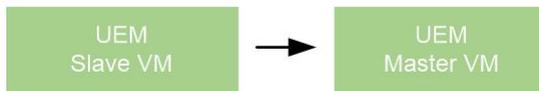
If the UEM Standby VM is the Zookeeper leader:



If the UEM Slave VM is the Zookeeper leader:



If the UEM Master VM is the Zookeeper leader:



427766

**Figure 26: UEM VM Upgrade Order for 3 VM based UEM Cluster**

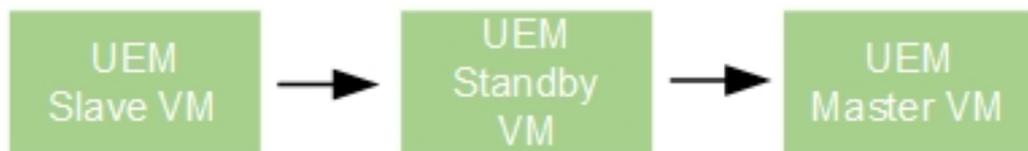
If the UEM Standby VM is the Zookeeper leader:



If the UEM Slave VM is the Zookeeper leader:



If the UEM Master VM is the Zookeeper leader:



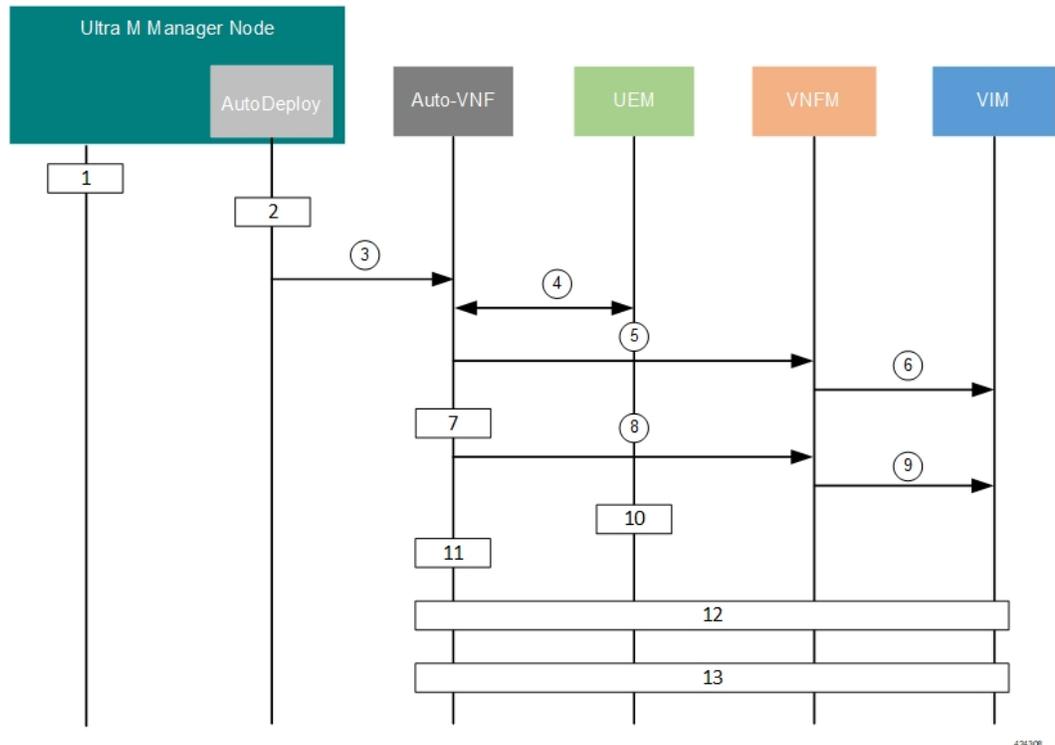
424309



**Important** The UEM patch upgrade process is supported for Ultra M deployments that leverage the Hyper-Converged architecture and for stand-alone AutoVNF deployments.

Figure 27: UEM Patch Upgrade Process Flow, on page 251 illustrates the UEM patch upgrade process for Ultra M deployments. For stand-alone AutoVNF deployments, the upgrade software image is uploaded to the onboarding server (step 1) and the upgrade command is executed from AutoVNF (step 3).

**Figure 27: UEM Patch Upgrade Process Flow**



1. Onboard the new USP ISO containing the UEM upgrade image to the Ultra M Manager node.
2. Update the deployment network service descriptor (NSD) to identify the new package and onboard it.

```
nsd nsd_name_including_vnfm_vnfd
```

```
vnf-package [ previous_package_descriptor_name upgrade_package_descriptor_name ]
```

Package information is defined in the VNF package descriptor (vnf-packaged) as follows:

```
<---SNIP--->
```

```
vnf-packaged <upgrade_package_descriptor_name>
```

```
  location <package_url>
```

```
  validate-signature false
```

```
  configuration staros
```

```
    external-url /home/ubuntu/system.cfg
```

```
<---SNIP--->
```

The package must then be referenced in the virtual descriptor unit (VDU) pertaining to the UEM:

```

<---SNIP--->
vdu em
vdu-type          element-manager
login-credential  em_login
scm               scm
image vnf-package
vnf-rack          vnf-rack1
vnf-package primary <upgrade_package_descriptor_name>
vnf-package secondary <previous_package_descriptor_name>
...
<---SNIP--->

```

**Important**

The secondary image is used as a fallback in the event an issue is encountered through the upgrade process. If no secondary image is specified, the upgrade process will stop and generate an error log.

3. The rolling upgrade request is triggered through AutoDeploy which initiates the process with AutoVNF.
4. AutoVNF obtains the UEM HA VIP from the Oper data and communicates with the corresponding UEM to determine the IP addresses of the eth0 interface for each VM in the UEM cluster. This information is maintained in a file on the VM named *ip.txt*.

AutoVNF then uses the address information to communicate with each UEM to determine their Zookeeper state (master, slave, and standby).

The upgrade order is illustrated in [Figure 26: UEM VM Upgrade Order for 3 VM based UEM Cluster, on page 250](#). The rest of this procedure assumes that that the standby UEM VM is the Zookeeper-elected leader.

5. AutoVNF triggers the shutdown of the slave UEM VM via the VNFM.
6. The VNFM works with VIM to remove the slave UEM VM.
7. AutoVNF waits until the VNFM confirms that the slave UEM VM has been completely terminated.
8. AutoVNF initiates the deployment of a new UEM VM via the VNFM using the upgrade image.
9. The VNFM works with the VIM to deploy the new UEM VM.

**Important**

- If the ESC does not receive the “SERVICE UPDATE” notification for the newly added VM instances, the upgrade will fail and require a manual intervention.
- If ESC state (service/VM state ) is not ‘ACTIVE’, then the upgrade will not proceed. You need to manually verify the logs to determine the reason for the inactive state.

10. The slave UEM VM synchronizes data with the master UEM VM.
11. AutoVNF waits until the VNFM confirms that the new VM has been deployed and is in slave mode. If AutoVNF detects that there is an issue with the VM, it re-initiates the UEM VM with the previous image if it was identified as a secondary image in the UEM VDU. If no issues are detected, AutoVNF proceeds with the upgrade process.

12. Repeat the steps 4, on page 252 to 10, on page 252 for the UEM VM that is currently the master. Once the master goes down, the slave UEM becomes the master.

If an issue is encountered during the upgrade of the second UEM VM (e.g. the master UEM VM in this scenario), then the process stops completely and AutoVNF upstart logs are generated.

13. Repeat the steps 4, on page 252 to 8, on page 252 for the standby VM. In this case, the UEM is re-deployed as the standby VM.

## Initiating the UEM Patch Upgrade

UEM patch upgrades are initiated through a remote procedure call (RPC) executed from the ConfD command line interface (CLI) or via a NETCONF API.

### Via the CLI

To perform an upgrade using the CLI, log in to AutoDeploy (Ultra M deployments) or AutoVNF (stand-alone AutoVNF deployments) as the ConfD CLI *admin* user and execute the following command:

```
update-sw nsd-id <nsd_name> rolling { true | false } vnfd <vnfd_name>
vnf-package <pkg_id>
```

#### NOTES:

- <nsd\_name> and <vnfd\_name> are the names of the network service descriptor (NSD) file and VNF descriptor (VNFD) (respectively) in which the VNF component (VNFC) for the UEM VNF component is defined.
- If the **rolling false** operator is used, the upgrade terminates the entire deployment. In this scenario, the **vnfd<vnfd\_name>** operator should not be included in the command. If it is included, a transaction ID for the upgrade is generated and failed. The AutoVNF upstart log reflects this status.
- <pkg\_id> is the name of the USP ISO containing the upgraded UEM VM image.
- Ensure that the upgrade package is defined as a VNF package descriptor within the NSD and that it is specified as the primary package in the UEM VDU configuration.
- Ensure that the current (pre-upgrade) package is specified as the secondary package in the UEM VDU configuration in order to provide rollback support in the event of errors.

### Via the NETCONF API

**Operation:** nsd:update-sw

**Namespace:** xmlns:nsd="http://www.cisco.com/usp/nfv/usp-nsds"

#### Parameters:

Parameter Name	Required	Type	Description
nsd	M	string	NSD name
rolling	M	boolean	Specifies if the rolling is enabled (true) /disabled (false)

Parameter Name	Required	Type	Description
vnfd	M	string	VNFD name, mandatory in case of rolling upgrade
package	M	string	Package descriptor name that should be used to update the vnfd instance mentioned by “vnfd”

**NOTES:**

- If the **rolling false** operator is used, the upgrade terminates the entire deployment. In this scenario, the **vnfd<vnfd\_name>** operator should not be included in the command. If it is included, a transaction ID for the upgrade is generated and failed. The AutoVNF upstart log reflects this status.
- Ensure that the upgrade package is defined as a VNF package descriptor within the NSD and that it is specified as the primary package in the UEM VDU configuration.
- Ensure that the current (pre-upgrade) package is specified as the secondary package in the UEM VDU configuration in order to provide rollback support in the event of errors.

**Example RPC**

```
<nc:rpc message-id="urn:uuid:bac690a2-08af-4c9f-8765-3c907d6e12ba" <nsd
xmlns="http://www.cisco.com/usp/nfv/usp-nsds">
  <nsd-id>fremont-autovnf</nsd-id>
  <vim-identity>vim1</vim-identity>
  <vnfd xmlns="http://www.cisco.com/usp/nfv/usp-nsds">
    <vnfd-id>esc</vnfd-id>
    <vnf-type>esc</vnf-type>
    <version>6.0</version>
    <configuration>
      <boot-time>1800</boot-time>
      <set-vim-instance-name>true</set-vim-instance-name>
    </configuration>
    <external-connection-point>
      <vnfc>esc</vnfc>
      <connection-point>eth0</connection-point>
    </external-connection-point>
    <high-availability>true</high-availability>
    <vnfc>
      <vnfc-id>esc</vnfc-id>
      <health-check>
        <enabled>>false</enabled>
      </health-check>
    </vnfc>
    <vdu>
      <vdu-id>esc</vdu-id>
    </vdu>
    <connection-point>
      <connection-point-id>eth0</connection-point-id>
      <virtual-link>
        <service-vl>mgmt</service-vl>
      </virtual-link>
    </connection-point>
    <connection-point>
      <connection-point-id>eth1</connection-point-id>
      <virtual-link>
        <service-vl>orch</service-vl>
      </virtual-link>
    </connection-point>
  </vnfd>
</nsd>
```

```

        </connection-point>
    </vnfc>
</vnfd>
</nsd>
<vim xmlns="http://www.cisco.com/usp/nfv/usp-uas-common">
    <vim-id>vim1</vim-id>
    <api-version>v2</api-version>
    <auth-url>http://172.21.201.218:5000/v2.0</auth-url>
    <user>vim-admin-creds</user>
    <tenant>abcxyz</tenant>
</vim>
<secure-token xmlns="http://www.cisco.com/usp/nfv/usp-secure-token">
    <secure-id>vim-admin-creds</secure-id>
    <user>abcxyz</user>
    <password>*****</password>
</secure-token>
<vdu xmlns="http://www.cisco.com/usp/nfv/usp-uas-common">
    <vdu-id>esc</vdu-id>
    <vdu-type>cisco-esc</vdu-type>
    <flavor>
        <vcpus>2</vcpus>
        <ram>4096</ram>
        <root-disk>40</root-disk>
        <ephemeral-disk>0</ephemeral-disk>
        <swap-disk>0</swap-disk>
    </flavor>
    <login-credential>esc_login</login-credential>
    <netconf-credential>esc_netconf</netconf-credential>
    <image>
        <vnf-package>usp_throttle</vnf-package>
    </image>
    <vnf-rack>abcxyz-vnf-rack</vnf-rack>
    <vnf-package>
        <primary>usp_6_2t</primary>
        <secondary>usp_throttle</secondary>
    </vnf-package>
    <volume/>
</vdu>
<secure-token xmlns="http://www.cisco.com/usp/nfv/usp-secure-token">
    <secure-id>esc_login</secure-id>
    <user>admin</user>
    <password>*****</password>
</secure-token>
<secure-token xmlns="http://www.cisco.com/usp/nfv/usp-secure-token">
    <secure-id>esc_netconf</secure-id>
    <user>admin</user>
    <password>*****</password>
</secure-token>
<vnf-packaged xmlns="http://www.cisco.com/usp/nfv/usp-uas-common">
    <vnf-package-id>usp_throttle</vnf-package-id>
    <location>http://192.168.200.61:5000/isos/fremont-autovnf_esp_throttle</location>
    <validate-signature>false</validate-signature>
    <configuration>
        <name>staros</name>
</configuration>
</vnf-packaged>
</config>
<external-url>http://192.168.200.61:5000/isos/fremont-autovnf_esp_throttle_staros</external-url>

```

