Signaling Delivery Controller

Virtual Openstack Installation, Upgrade, and Maintenance Guide

5.1

Catalog Number: RG-016-51-29 Ver. 12

Publication Date: November 2020

Legal Information

Copyright

Trademarks

Patents

This product may be protected by one or more patents indicated at: *http://www.f5.com/about/guidelines-policies/patents*

Confidential and Proprietary

The information contained in this document is confidential and proprietary to F5. The information in this document may be changed at any time without notice.

About F5

F5 (NASDAQ: FFIV) makes the connected world run better. F5 helps organizations meet the demands and embrace the opportunities that come with the relentless growth of voice, data, and video traffic, mobile workers, and applications—in the data center, the network, and the cloud. The world's largest businesses, service providers, government entities, and consumer brands rely on F5's intelligent services framework to deliver and protect their applications and services while ensuring people stay connected. For more information, visit *www.F5.com* or contact us at *Tfx_info@f5.com*.

About this Document

Document Name: F5 Signaling Delivery Controller Virtual Openstack Installation, Upgrade, and Maintenance Guide

Catalog Number: RG-016-51-29 Ver. 12

Publication Date: November 2020

Document Objectives

This document provides the procedures of how to virtually install, upgrade, and maintain an SDC deployment. It also provides an overview of how the SDC works with an Orchestrator.

Note: For this release, the installations and maintenance procedures described in this guide are only supported for an SDC deployment and not an EMS deployment.

Note: In this document, "server" and "machine" are used interchangeably.

Document History

Revision Number	Change Description	Change Location
Ver. 2 – November 2016	Added description of ports used by the SDC	*Port Settings Used by the SDC*
Ver. 3 – February 2017	Edited description of creating a Site Topology File. Added ports that are used by the SDC	*Creating a Site Topology File' Port Settings Used by the SDC*
Ver. 4 – February 2017	Updated the description of data volumes size	*Data Storage Volume*
Ver. 5 – May 2017	Added add/deleteRoute script procedure. Added site topology file structure.	*Adding and Deleting Routes via Script*

Revision Number	Change Description	Change Location
		Appendix A:Site Topology File Structure Adding and Deleting Routes via Script
Ver. 6 – August 2017	Removed post-installation procedure: Changing the SNMP Community String. Updated the ports description	*Port Settings Used by the SDC*
Ver. 7 – September 2017	Added procedure for migrating traffic forma a FEP (FEP failover). Updated the ports description	*Migrating Traffic from a FEP,* *Port Settings Used by the SDC*
Ver. 8 – December 2017	Corrected spacing in traffix.site Stop curl command	*Prerequisite*
Ver. 9 – June 2018	Added nameserver (optional) to site topology file	*Table 51: Elements Defined in siteProperties*
Ver. 10 – Oct 2018	Added note about Site by Site upgrade from CF 12 and below	*Stopping all SDC Services on Site 1*
Ver. 11 – Oct 2020	Added creating the topology.nameserver table for upgrades pre-CF17. Other edits in Upgrading site by site section	*Creating the topology.nameserver table* *Upgrading Site by Site*
Ver. 12 – Nov 2020	Removed references to Splunk, and updated to ELK	*Throughout document*

Conventions

The style conventions used in this document are detailed in Table 1.

Table 1: Conventions

Convention	Use
Normal Text Bold	Names of menus, commands, buttons, user-initiated CLI commands and other elements of the user interface
Normal Text Italic	Links to figures, tables, and sections in the document, as well as references to other documents
`Script`	Language scripts
`Courier`	File names
Note:	Notes which offer an additional explanation or a hint on how to overcome a common problem
Warning:	Warnings which indicate potentially damaging user operations and explain how to avoid them

Table of Contents

1. Introduction 1
1.1 SDC Installed Components 2
1.1.1 Active/Active High Availability 3
2. Prerequisites 4
2.1 General Prerequisites 4
2.2 Completing a Site Survey 4
2.3 Cloud-owner Requirements 5
2.4 Installing the Hardware 8
2.5 Accessing an XSD File 8
2.6 Creating a Site Topology File 8
2.7 Creating a Nova Metadata File 9
2.8 Uploading QCow2 Images 10
3. Working with a Cloud Orchestrator 11
3.1 Connecting to a Cloud Orchestrator 11
3.1.1 Authenticating the SDC-Orchestrator Connection 11
3.1.2 Managing vSDC Component Applications 13
3.2 Monitoring the Status of an API Request 13
4. Installing a New Site 14
4.1 Creating the vInstaller VMs 14
4.2 Creating the OAM VMs 15
4.3 Creating the FEP, CPF, and Tripo Components 15
4.4 Monitoring the Status of a New Installation 16
4.5 Post Installation Procedures 17
4.5.1 Changing the Root Password 17
4.5.2 Add Licenses to FEP IP Addresses 17
5. Adding a CPF Component 18
5.1 Prerequisite 18
5.2 Creating an XML with the Added CPFs 18
5.3 Command Execution Codes 18
5.4 Scaling-out API Request 18
5.4.1 Monitoring the Status of a Scale-out API Request 19
6. Removing a CPF Component 21
6.1 Prerequisite 21
6.2 Scaling-in API Request 21
6.2.1 Monitoring the Status of a Scale-in API Request 22
6.3 Scaling-in Status API Request 23
6.3.1 Monitoring the Status of ScaleInStatus API Request 23
7. Migrating Traffic from a FEP 25
8. Migrating a Server 26
8.1 Prerequisite 26
8.2 Migrating API Request 26
8.2.1 Monitoring the migrate API Request 27
9. Stopping all SDC Services 29
9.1 Prerequisite 29
9.2 siteStop API Request 29
9.2.1 Monitoring the siteStop API Request 30

9.3 siteStop Status API Request 31
9.3.1 Monitoring the siteStop StatusAPI Request 31
10. Restarting Site Services 32
10.1 siteRecovery API Request 32
10.1.1 Monitoring the siteRecovery API Request 32
11. Removing all SDC Components 34
11.1 Prerequisite 34
11.2 siteDecommission API Request 34
11.2.1 Monitoring the Status of a siteDecommission API Request 34
11.3 decommissionStatus API Request 36
11.3.1 Monitoring the Status of a decommissionStatus API Request 36
12. Backing up and Restoring a Site 37
12.1 Prerequisite 37
12.2 Backing up the Data 37
12.3 Restoring the Data 37
13. Backing up and Restoring a Cassandra Database 39
13.1 Backing up a Site's Cassandra Database 39
13.2 Restoring the Cassandra Database 39
14. Upgrading Site by Site 41
14.1 What Happens During a Site by Site Upgrade? 41
14.2 Steps for Shutting Down Site 1 44
14.2.1 Redirecting Traffic from Site 1 to Geo-Redundant Site 44
14.2.2 Creating the topology.nameserver table 44
14.2.3 Stopping all SDC Services on Site 1 45
14.2.4 Shutting down on Site 1 46
14.3 Copying the External Storage 47
14.4 Installing the Upgraded Site 3 47
14.5 Installing the Upgraded Geo-redundant Site 47
14.6 Performing an Upgrade Rollback 48
15. Monitoring an SDC Deployment Status 49
15.1 Prerequisite 49
15.2 Application Status per VM 49
15.2.1 appStatus API Request 49
15.3 Site Status 51
15.3.1 siteStatus API Request 51
15.4 Application Execution Status per VM 53
15.4.1 Application Status API Request 54
16. Using Logs for Troubleshooting 55
17. Configuring SDC-Server Routes 56
17.1 Add Routes 56
17.1.1 Prerequisites 56
17.1.2 Updating the Site Topology File with Added Route 56
17.1.3 addRoutes API Request 57
17.2 Delete Routes 57
17.2.1 Prerequisites 58
17.2.2 Updating the Site Topology File with Deleted Route 58
17.2.3 deleteRoutes API Request 58
17.3 Adding and Deleting Routes via Script 59

17.4 Show Routes 61
17.4.1 Prerequisites 61
17.4.2 showRoutes API Request 61
18. Replacing NTP Servers 63
18.1.1 Prerequisites 63
18.1.2 Updating the Site Topology File with New NTP Servers 63
18.1.3 changeNtp API Request 64
19. Changing the Log Servers 65
19.1 Prerequisites 65
19.1.1 Verifying the Authentication Token 65
19.2 Updating the Site Topology File with the New Log Servers 65
19.3 Sending a changeLogServer API Request 66
19.3.1 changeLogServers API Request 66
Appendix A: Site Topology File Structure 68
Appendix B: Example of Scaling-out Site Topology File 85
Appendix C: Port Settings Used by the SDC 87
C.1 EMS Site Internal Ports 87
C.2 EMS Site External Ports 88
C.3 SDC Site Internal Ports 90
C.4 SDC Site External Ports 92
C.5 HP Integrated Lights-Out (iLO) Port Settings 94
Appendix D: ELK Components 95
Glossary 96

List of Figures

Figure 1: vDRA Deployment 3
Figure 2: Installation Flow 14

List of Tables

Table 1: Conventions IV
Table 2: Cinder Memory Storage Requirements for an SDC Site 6
Table 3: Data Volume Storage Requirements per Application for an SDC Site 6
Table 4: Cinder Memory Storage Requirements for EMS Site 6
Table 5: Data Volume Storage Requirements per Application for an EMS Site 7
Table 6: Mandatory Parameters 9
Table 7: Optional Parameters 10
Table 8: API Status Output Codes 13
Table 9: New Installation Flow Status Output Codes 16
Table 10: Scale-out Command Error Codes 19
Table 11: Scale-out Command Success Codes 19
Table 12: Scale-out API Request Status Output Codes 20
Table 13: Scale-in Command Error Codes 22

Table 14: Scale-in Command Success Codes 22
Table 15: Scale-in API Request Status Output Codes 23
Table 16: ScaleInStatus Command Error Codes 23
Table 17: ScaleInStatus Command Success Codes 24
Table 18: Migrate Command Error Codes 27
Table 19: Migrate Command Success Codes 27
Table 20: Migrate API Request Status Output Codes 27
Table 21: siteStop Command Error Codes 30
Table 22: siteStop Command Success Codes 30
Table 23: siteStop API Request Status Output Codes 30
Table 24: siteStop Status Command Error Codes 31
Table 25: siteStop Status Command Success Codes 31
Table 26: siteRecovery Command Error Codes 32
Table 27: siteRecovery Command Success Codes 33
Table 28: siteDecommission Command Error Codes 35
Table 29: siteDecommission Command Success Codes 35
Table 30: siteDecommission API Request Status Output Codes 35
Table 31: decommissionStatus Command Error Codes 36
Table 32: decommissionStatus Command Success Codes 36
Table 33: appStatus Command Error Codes 49
Table 34: appStatus Return Codes 50
Table 35: siteStatus Command Error Codes 51
Table 36: siteStatus Return Codes 52
Table 37: Elements Defined in Topology 68
Table 38: Elements Defined as Part of each VM 69
Table 39: Elements Defined in each Interface Element 70
Table 40: Elements Defined in Route 71
Table 41: Elements Defined for Each Application Instance 72
Table 42: Elements Defined in Volumes 73
Table 43: Elements Defined in Partitions 74
Table 44: Elements Defined for Each Network Element 75
Table 45: Elements Defined for the FEP Application 77
Table 46: Elements Defined for the CPF Application 78
Table 47: Elements Defined for the Tripo Application 79
Table 48: Elements Defined for the oamDB Application 80
Table 49: Elements Defined for the vip Application 81
Table 50: Elements Defined in siteProperties 83
Table 51: EMS Internal Ports 87
Table 52: EMS External Ports 88
Table 53: SDC Internal Ports 90
Table 54: SDC External Ports 92

Table 55: HP iLO Ports 94
Table 56: Common Terms 96
Table 58: Abbreviations 97

1. Introduction

The installation and maintenance processes install, configure, and enable the necessary hardware, network infrastructure, and site components needed to process F5® Traffix® Signaling Delivery Controller™ (SDC) traffic.

In this release, the installation and maintenance processes are performed using a Cloud Orchestrator. This document describes all procedures as they relate to the installation and maintenance processes using a Cloud Orchestrator.

In virtualized deployments, REST APIs are used to communicate between the Orchestrator and the SDC vInstaller to install, maintain, and upgrade an SDC site.


Note: From 5.1 CF 30, EMS deployments will use ELK components, instead of Splunk components, to manage all SDC reporting functionalities. This change is reflected in version 12 and higher of the *Virtual OpenStack Installation, Maintenance, and Upgrade Guide*.

The following installation and maintenance procedures are supported in this release:

- *Installing a New Site*
- *Adding a CPF Component*
- *Removing a CPF Component*
- *Stopping all SDC Services*
- *Migrating Traffic from a FEP*
- *Migrating a Server*
- *Restarting Site Services*
- *Removing all SDC Components*
- *Backing up and Restoring a Site*

- *Backing up and Restoring a Cassandra Database*
- *Upgrading Site by Site*
- *Monitoring an SDC Deployment Status*
- *Using Logs for Troubleshooting*
- *Configuring SDC-Server Routes*
- *Replacing NTP Servers*
- *Changing the Log Servers*

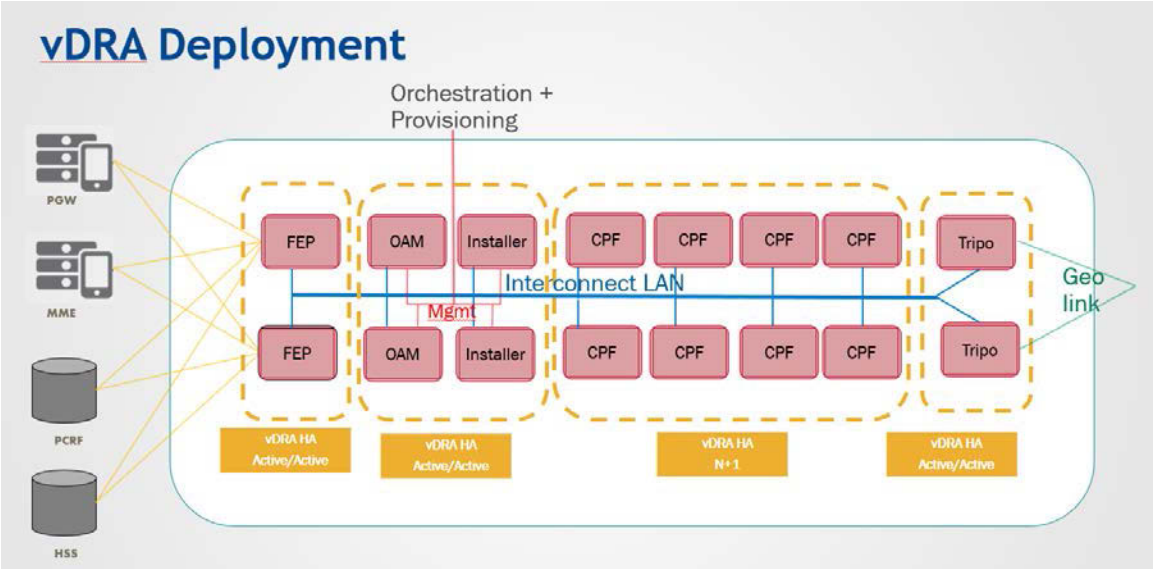
1.1 SDC Installed Components

An SDC site is comprised of the following components (VNFCs) that interact with one another to provide full service and management capabilities:

- vInstaller - manages the installation and upgrade of SDC components. The vInstaller virtual machines also include the system database (Cassandra)
- OAM - provides the configuration, provisioning and management of FEP and CPF, through the configuration manager, the NMS Agent, the Web UI and the system database (Cassandra)
- FEP - provides the connectivity end point to the SDC for Diameter and other supported protocol peers and a Virtual IP address to the peers. The FEP load balances Diameter and other supported protocol messages to the Control Plane Functions (CPFs)
- CPF - provides the rules implementation of Diameter and other supported protocol traffic
- Tripo - maintains session information for session binding and stateful routing.

In a virtualized environment, each one of the components is installed on a separate virtual machine (VM). The following diagram shows the basic architecture for a virtual DRA deployment.

Figure 1: vDRA Deployment



1.1.1 Active/Active High Availability

In SDC 5.1, all SDC components are installed in Active-Active mode, ensuring seamless high availability in case one of the component instances fails.

2. Prerequisites

This section describes the prerequisites associated with installing, maintaining, and upgrading a virtualized deployment.

2.1 General Prerequisites

This document assumes that you have a comprehensive understanding of:

- SDC deployments

  Note: Only SDC deployments are supported in virtualized environments in this release.

  Note: When copy-pasting curl requests, make sure the full request is pasted in as appears in this document, so that the correct spacing and "-d" is copied as expected.

- SDC architecture
- SDC pipeline
- Cloud and Orchestrator terminology and functionality for virtualized environments

2.2 Completing a Site Survey

To correctly assess your specific needs and ensure that the installed solution will meet them, a site survey, reviewing your anticipated traffic type and scope, is completed. Based on the site survey, a solution is built and the hardware requirements and site configuration recommendations are decided upon.

Based on the site survey, the number of needed CPFs are calculated and that determines the site deployment size. All other components, two vInstallers, two OAMS, four FEPs, and two Tripos remain constant.

Note: This document assumes that this stage has been successfully completed.

2.3 Cloud-owner Requirements

This section describes what resources must be in place by the cloud-owner (the customer) prior to installation.

Customers are required to supply an IaaS platform for the SDC to be hosted on.

2.3.1.1 CPU

For high performance that is required for SDC the following CPU functions are required:

- VTx and VTd virtualization extensions for HW assisted virtualization
- CPU pinning for VM vCPU

2.3.1.2 Data Storage Volume

The cloud owner needs to allocate two types of data volumes:

- Root disks are used for booting a virtual machine. This type of disk is ephemeral and the QCOW2 image is copied into it, as well as the OS. There can also be a secondary ephemeral disk, that is empty and only exists for the life of the instance. As these disks are not persistent, they may be erased following a VM termination (scaling-in).
- Persistent disks are used to hold the persistent data (KPIs, statistics, logs and configuration). Persistent volume is needed for two types of machines: OAM and the vInstaller. During a migration orchestration flow, the persistent volumes remain intact and all data is saved.

Note: A Warning message is generated in the logs when the volume is not configured correctly, including an invalid volume size or partition name, in the site topology file.

2.3.1.2.1 Data Volume Requirements for an SDC Site

The following are the storage requirements for the vInstaller and OAM VM cinder volumes for persistent disks for an SDC site:

Table 2: Cinder Memory Storage Requirements for an SDC Site

Cinder Name	Attached to VM	Size (GB)	Volume Type
Installer01	vInstaller	300	SATA
Installer02	vInstaller	300	SATA
Oam01	OAM VM	300	SATA
Oam02	OAM VM	300	SATA

Each VM server cinder volume is sub-divided into partitions to host the different applications:

▪ The vInstaller hosts, on its persistent disk, the Cassandra data (oamdb) and the installation repository (repo).

▪ The OAM VM hosts, on its persistent disk, the Cassandra data (oamdb), the Configuration Manager (CM backup), and the NMS Agent (central syslog data).

Table 3: Data Volume Storage Requirements per Application for an SDC Site

VM Server	Application Type	Partition Name	Mount Type	Size (MB)
vInstaller	vInstaller	oamdb	/data/cassandra	128000
vInstaller	vInstaller	repo	/opt/repo/updates	128000
OAM	OAM-DB	oamdb	/data/cassandra	128000
OAM	CM	backup	/data/backup	8200
OAM	NMS	logs	/var/log/rsyslog	128000

2.3.1.2.2 Data Volume Requirements for an EMS Site

The following are the storage requirements for the vInstaller and OAM VM cinder volumes for persistent disks for an EMS site:

Table 4: Cinder Memory Storage Requirements for EMS Site

Cinder Name	Attached to VM	Size (GB)	Volume Type
EMS01 (Installer+OAM)	EMS01	900	SATA

Cinder Name	Attached to VM	Size (GB)	Volume Type
EMS02 (Installer+OAM)	EMS02	900	SATA

Each VM server cinder volume is sub-divided into partitions to host the different applications:

▪ The vInstaller hosts, on its persistent disk, the Cassandra data (oamdb) and the installation repository (repo).

▪ The OAM VM hosts, on its persistent disk, the Cassandra data (oamdb), the Configuration Manager (CM backup), and the NMS Agent (central syslog data).

Table 5: Data Volume Storage Requirements per Application for an EMS Site

VM Server	Application Type	Partition Name	Mount Type	Size (MB)
EMS	OAM-DB	oamdb	/data/cassandra	128000
EMS	CM	backup	/data/backup	8200
EMS	NMS	logs	/var/log/rsyslog	64000
EMS	ELK	data	/data	512000
EMS	vInstaller	repo	/opt/repo/updates	128000

2.3.1.3 Networking (SRIOV/OVS/LBaas…)

The system requires low latency high performance networking. Both internal interconnect (IC) network and signaling networks require:

▪ Low latency – 1 ms

▪ Low jitter – 1ms

▪ High bandwidth – 10 Gbps

The networking characteristics may be achieved using the some of the following:

▪ High performance redundant 10 Gb switches

▪ SR-IOV capable interface cards on the hypervisor

- Intel DPDK Open vSwitch
- CPU pinning for VM vCPUs

2.4 Installing the Hardware

Before initiating the installation process with the Cloud Orchestrator, you need to install (and verify the successful installation) of the required hardware, per the recommendations decided upon based on the site survey.

Note: This document assumes that this stage has been successfully completed.

2.5 Accessing an XSD File

F5 supplies an XSD file to the customer. The XSD file, based on the customer site survey, is a schema that contains the structure for the site topology file and, as such, is used to validate the site topology file.

2.6 Creating a Site Topology File

The site topology file is created jointly by the customer and F5 and is based on the customer site survey. This site topology file is sent to and then used by the Orchestrator to create the vInstaller, OAM, and other SDC component VMs.

The site topology file contains all the network and application information, as well as the site properties that are needed to create each VM, for the site. Therefore, you should have an understanding of the positioning of the SDC in and/or between networks including the relevant IP and network (i.e. port) information needed for your site

The site topology file is saved in Cassandra, the system database, as a keyspace schema.

For more information on the structure of the site topology file, see *Appendix A:Site Topology File Structure*

2.7 Creating a Nova Metadata File

The Nova metadata (params) file contains boot parameters that define a server's role as either a master that hosts the vInstaller or as a minion that hosts the OAM and other SDC components. Configuring the parameters is done from the Nova metadata file and must be completed prior to installation. There are mandatory parameters and optional parameters that are only required if relevant for the deployment.

Note: The parameters must be defined for each server.

The following parameters must be defined in the Nova metadata file:

Table 6: Mandatory Parameters

Name	Value Description
server	master/minion
hostname	the server's hostname Note: The hostname must be identical (case sensitive) to the value defined under the name attribute for the vm element in the site topology file
master0	The IP address on the management network that the first vInstaller uses.
master1	The IP address on the management network that the second vInstaller uses.
ip0	This relates to the IP address for a server that has a management network interface, such as the vInstaller or OAM. Note: This is mandatory for vInstaller and OAM servers.
ip1	This relates to the IP address for all servers (master and minions), that is from the management network interface
netmask0	This relates to the netmask for a server that has a management network interface, such as the master Installer or OAM. (CIDR is not supported on IPv4).

Name	Value Description
netmask1	This relates to the netmask for all servers with an interconnect network interface. (CIDR is not supported on IPv4).

Table 7: Optional Parameters

Name	Value Description
vlan0	vlan number for management interface (if vlan defined)
vlan1	vlan number for interconnect network interface (if vlan defined)
gw	default gateway (need to be mandatory if server = master)
debug	debug=yes enable salt log with debug
dns	DNS

2.8 Uploading QCow2 Images

There are two required QCow2 images that are provided by F5 to the customer to be uploaded to an Open Stack repository:

- Master – contains all RPMs and installation binaries for the Installer VM to be able to install all other VMs according to their roles
- Minion – contains the basic OS (Linux) for the OAM and all other SDC VM components

These images are used by the Orchestrator to create the Installer, OAM, and other SDC component VMs.

3. Working with a Cloud Orchestrator

In a virtualized deployment, an OpenStack Orchestrator sets-up and installs the Master vInstallers, the OAMs, and the other SDC components.

3.1 Connecting to a Cloud Orchestrator

REST APIs are used to communicate between the Orchestrator and the SDC vInstaller. These APIs are based on a standard Salt API interface and the body of the REST API message contains CLI Salt functions.

The interaction between the Orchestrator and the vInstaller is unidirectional. The Orchestrator queries the vInstaller and the vInstaller acts as a REST server.

The generic flow process is:

1. The Orchestrator sends an http request to the vInstaller. The vInstaller is identified as //<master IP address>:8000/ in the API requests.
2. While the request is pending, the Orchestrator sends another request to check its status.
3. When the status request has succeeded or failed, the request flow is completed.

3.1.1 Authenticating the SDC-Orchestrator Connection

Before the Orchestrator can send any requests, the Orchestrator must request and have a valid authentication token.

 Note: An authentication token expires after ten hours.

The following is the token authentication API:

```
curl -ksi https://<master IP address>:8000/login -H "Accept:
application/json" -d username='saltuser' -d password='traffix' -d
eauth='pam'
```

Note: For all API requests, you need to use the minus sign, for example "-d" and not the N-dash "-". If you copy–paste the API request, you may have to type in the "-d" again with the minus sign to avoid syntax conversion errors.

3.1.1.1 Authentication Request Example

The following is an example of a request and answer for an authentication token:

```
[root@vdra01501-master1 ~]# curl -ksi https://127.0.0.1:8000/login -H
"Accept: application/json" -d username='saltuser' -d password='traffix'
-d eauth='pam'
HTTP/1.1 200 OK
Content-Length: 189
Access-Control-Expose-Headers: GET, POST
Vary: Accept-Encoding
Server: CherryPy/3.2.2
Allow: GET, HEAD, POST
Access-Control-Allow-Credentials: true
Date: Thu, 05 May 2016 12:50:07 GMT
Access-Control-Allow-Origin: *
X-Auth-Token: fccedf8534805b56649895c7d802b2550ea80e41
Content-Type: application/json
Set-Cookie: session_id=fccedf8534805b56649895c7d802b2550ea80e41;
expires=Thu, 05 May 2016 22:50:07 GMT; Path=/
```

3.1.1.2 Authentication Request Status Codes

The following are the possible return codes for the authentication API request:

Return Code	Description
200	success
401	authentication required
406	requested Content-Type not available

3.1.2 Managing vSDC Component Applications

In this release, all components run in active/active mode and are managed (start and stopped) using Monit.

3.2 Monitoring the Status of an API Request

Each request flow has an associated API output success/error code depending on its status.

Table 8: API Status Output Codes

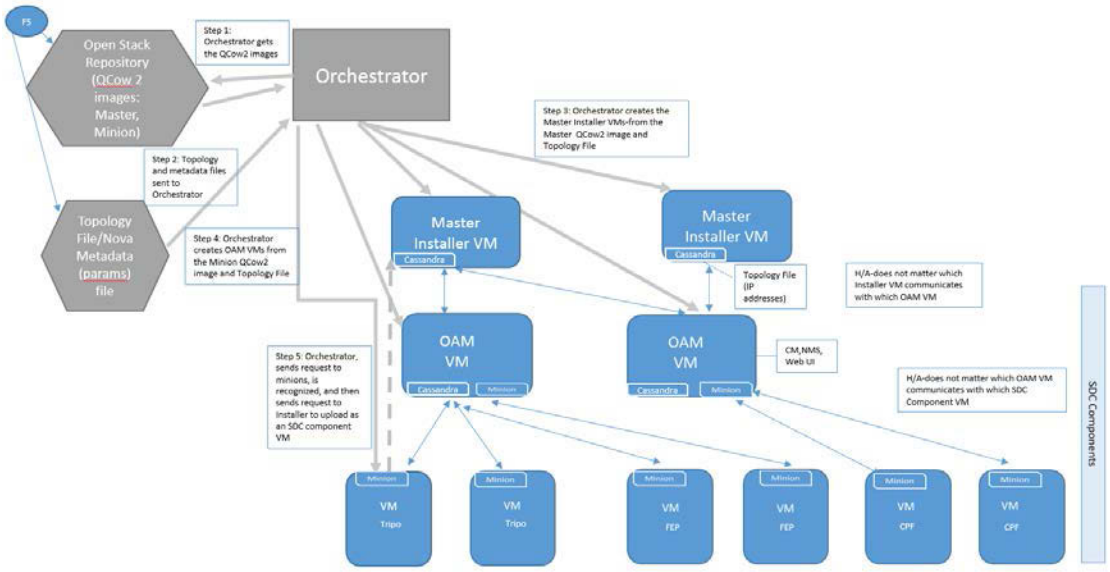
Status	Code	Description
Success	12000	The entire flow was completed successfully
Failure	150xx	The flow failed and was terminated
Pending internal	140xx	The vInstaller is waiting for an internal operation to complete. For example: waiting for a VM to start or waiting for an application to be installed
Pending orchestration	130xx	The relevant VM application service is down.

4. Installing a New Site

The initial installation process initiation is implicit and is based on the QCow images and the site topology file. The Orchestrator first creates the vInstaller VMs. The Orchestrator also provides the vInstaller VMs with the site topology file, with an indication for being a master installer. The vInstaller VMs, are then recognized as the master installers, and use the pre-provided site topology file to trigger an installation flow to create the OAM VMs and then the FEP, CPF, and Tripo components.

Note: There are no API requests for an initial installation.

Figure 2: Installation Flow



4.1 Creating the vInstaller VMs

The Orchestrator creates the vInstaller VMs based on the Master QCow2 image.

After the vInstaller VMs are successfully installed and are booting up, an auto self-installation process occurs, bringing up a vInstaller that is listening on a designated port, waiting for all the Salt minions of all the VMs that will host the different SDC components to register.

The vInstaller VMs are created with the site topology file and the Nova metadata file. The vInstaller VMs, using an XSD validator, validates the site topology file and saves it as a keyspace schema in the system database, Cassandra.

Note: Check if the vInstallers have been successfully installed before proceeding with the next component, OAM, installation. To do so send an appStatus API request as described in *Application Status per VM* and verify that the exit code is 12000, successfully installed.

4.2 Creating the OAM VMs

The Orchestrator creates the OAM VMs based on the Minion QCow2 image.

Once the Orchestrator's request is sent and recognized by a VM minion, the VM minion sends a request to the vInstaller VM to upload the OAM component, based on the network and application interface information that is included in the Nova metadata file and the topology.

The OAM VMs are created with the system database (Cassandra) and hosts the configuration manager, the NMS Agent, and the Web UI.

Note: Check if the OAM servers have been successfully installed before proceeding with the next component installation. To do so send an appStatus API request as described in *Application Status per VM* and verify that the exit code is 12000, successfully installed.

4.3 Creating the FEP, CPF, and Tripo Components

The Orchestrator creates the FEP, CPF, and Tripo VMs based on the Minion QCow2 image.

Once the Orchestrator's request is sent and recognized by a VM minion, the VM minion sends a request to the vInstaller VM to upload an SDC component, based on the network and application interface information that is included in the Nova metadata file and the topology.

The FEP, CPF, and Tripo components are created.

4.4 Monitoring the Status of a New Installation

The following are the status codes to monitor the flow process of a new installation.

Note: Immediately after installing an SDC site you may encounter multiple occurrences of the sdcMonitProcessRestart alarm in the Web UI. This does not impact performance.

Table 9: New Installation Flow Status Output Codes

Flow Status	Code	Description
Start	14001– success	Pending internal operation by vInstaller
Commit Topology	14002 – success 15001 – failure	vInstaller verifies if the site topology file is successfully committed
Initializing the VMs	14003 – success 15002 – failure	vInstaller verifies if the VMs are up and running
Installing the SDC Components on the VMs	14004 – success 15001 – failure	vInstaller verifies if the SDC components (with their applications) are successfully added to the VMs
Initializing the SDC components	12000 – success 15004 – failure	verifies if the SDC components are up and running Note: For EMS deployments, after salt highstate receives a result code of 12000, you may need to wait up to one minute until the Kibana configurations are complete and **Reports** are displayed as expected in the WebUI.

4.5 Post Installation Procedures

The following procedures are performed after the installation process is successfully completed:

- *Changing the Root Password*
- *Add Licenses to FEP IP Addresses*

4.5.1 Changing the Root Password

During installation, the Root password is assigned a default value. For increased security, change this value.

To change the root password:

1. Run the Unix "passwd" command.

4.5.2 Add Licenses to FEP IP Addresses

Each FEP IP address must have a license. During the installation, IP addresses were added to the FEP instances. These IP addresses much each have their own license. For more information about obtaining the licenses, contact *F5 Support* and refer to the *F5 SDC User Guide* on how to add a new license key.

5. Adding a CPF Component

You can seamlessly add additional CPF components to a site deployment by scaling-out.

Note: This release of vSDC supports scaling-out for CPF components. Scaling-out is limited to adding CPF components that are within a site's maximum capacity criteria. To ensure active-active high availability, two CPF components are added per API request.

5.1 Prerequisite

Verify that you have a valid authentication token that has not expired. The authentication token is needed for the API request (X-Auth-Token:<token>). If the token is no longer valid, generate a new token as described in *Authenticating the SDC-Orchestrator Connection*.

5.2 Creating an XML with the Added CPFs

You need to create an XML which includes the relevant network and application interface information for the added component. For an example of a scaling out site topology file, see *Scaling-out Site Topology File.* The scale-out API request then references the relevant <xml file>.

5.3 Command Execution Codes

Prior to processing any request flow, the system checks if the API request command is valid. If not, the relevant error codes, such as the following are generated:

- -17: Failed to connect to DB

The different return codes are described per request flow in each relevant flow section.

5.4 Scaling-out API Request

The following is the API request that is sent from the Orchestrator to query the vInstaller:

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token:<token>" -d client="runner" -d
fun="traffix.scale" -d role='out' -X POST -d @<xml file>
```

The vInstaller responds to the Orchestrator requests. Only once the scaling-out process is completed, the vInstaller can accept another request from the Orchestrator.

5.4.1 Monitoring the Status of a Scale-out API Request

The following are the command and status codes to monitor the API request flow.

5.4.1.1 Command Error Codes – Scaling-out

The following are the command execution error codes for scaling out:

Table 10: Scale-out Command Error Codes

Exit Code	Description
-10	%s is invalid option
-11	There is another Orchestrator flow currently running on the VNF
-12	Last installation request failed - check logs
-17	Failed to connect to DB
-1	Scale-out delta not valid according to xsd scheme
-2	Exception occurred when parsing
-31	Error occurred when tried to lock flow

5.4.1.2 Command Success Codes – Scaling-out

The following are the command execution success codes for scaling out:

Table 11: Scale-out Command Success Codes

Return Code	Description
0	scale out successful

5.4.1.3 Request Status Codes – Scaling-out

The following are the status codes for a scale-out request flow.

Table 12: Scale-out API Request Status Output Codes

Request Flow Status	Code	Description
Start	14001 – success	Pending internal operation by vInstaller
Initializing the added VM	14003 – success 15002 – failure	vInstaller verifies if added VM is up and running
Installing an SDC Component (i.e. CPF) on the added VM	14004 – success 15001 – failure	vInstaller verifies if SDC is added (with its applications) successfully to the VM
Initializing the added SDC component	12000 – success 15004 – failure	vInstaller verifies if the SDC component is up and running

6. Removing a CPF Component

You can seamlessly remove a CPF component from a site deployments by scaling-in.

Note: This release of vSDC supports scaling-in of a CPF component, and because of active-active high availability, this means removing two CPF components per API request. The CPF components are removed in the order they were added. There is no need to identify which CPF is to be removed.

Prior to removing a component, the vInstaller triggers and then verifies that a graceful shutdown occurred for the component that is being removed. There is no separate API request for graceful shutdown, as it is inherently included in the removal request.

The two required APIs for removing a CPF component are the scaling-in request and then the scaling-in status request to remove the CPF components from the site topology file.

6.1 Prerequisite

Verify that you have a valid authentication token that has not expired. The authentication token is needed for the API request (X-Auth-Token:<token>). If the token is no longer valid, generate a new token as described in *Authenticating the SDC-Orchestrator Connection*.

6.2 Scaling-in API Request

The following is the API request that is sent from the Orchestrator to query the vInstaller to remove a CPF component:

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d
fun="traffix.scale" -d role='in'
```

The response shows that the two relevant CPF components have been stopped following a graceful shutdown. The following is an example of a response:

```
return:
```

```
- initiateVmShutDown: Graceful shutdown request sent to cpf-138, cpf-
139
```

6.2.1 Monitoring the Status of a Scale-in API Request

The following are the command and status codes to monitor the API request flow.

6.2.1.1 Command Error Codes – Scaling-in API Request

The following are the command execution error codes for scaling in:

Table 13: Scale-in Command Error Codes

Exit Code	Description
-10	%s is invalid option
-11	There is another Orchestrator flow currently running on the VNF
-12	Last installation request failed - check logs
-17	Failed to connect to DB
-2	initiateVmShutDown: failure to send shutdown request to server
-1	ScaleIn request: Not enough CPFs to initiate scale down: num cpfs=X
-31	Error occurred when tried to lock flow

6.2.1.2 Command Success Codes – Scaling-in API Request

The following are the command execution success codes for scaling in:

Table 14: Scale-in Command Success Codes

Return Code	Description
0	initiateVmShutDown: Graceful shutdown request sent to <vm names>

6.2.1.3 Request Status Codes – Scaling-in

The following are the status codes for a scale-in request flow.

Table 15: Scale-in API Request Status Output Codes

Request Flow Status	Code	Description
Start	14005 – success	Pending internal operation by vInstaller
Graceful shutdown of SDC VM	14003 – success 15002 – failure	vInstaller verifies that a graceful shutdown occurred
Updating the site topology file	12000 – success 15006 – failure	vInstaller verifies that the updated site topology file is successfully committed to the OAM database (Cassandra)

6.3 Scaling-in Status API Request

The following is the API request to remove the servers from the site topology file:

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token: <Token>" -d client="runner" -d fun="traffix.scaleInStatus"
```

The answer confirms if the selected CPF was successfully removed from the site topology file.

6.3.1 Monitoring the Status of ScaleInStatus API Request

The following are the command and status codes to monitor the API request flow.

6.3.1.1 Command Error Codes – scaleInStatus Request

The following are the command execution error codes for the scaleInStatus request:

Table 16: ScaleInStatus Command Error Codes

Exit Code	Description
-10	%s is invalid option
-11	There is another Orchestrator flow currently running on the VNF
-12	Last installation request failed - check logs
-17	Failed to connect to DB
-20	The following servers failed to stop: <VM NAMES>

Exit Code	Description
-21	Scale in flow is running
-22	No scale in flow running
-1	ScaleIn request: Not enough CPFs to initiate scale down: num cpfs=X

6.3.1.2 Command Success Codes – scaleInStatus Request

The following are the command execution success codes for scaleInStatus request:

Table 17: ScaleInStatus Command Success Codes

Return Code	Description
0	ScaleIn request: %s successfully removed from topology

7. Migrating Traffic from a FEP

You can migrate traffic from one FEP component located on one server to another FEP component located on another server. The SDC supports a FEP failover mechanism that allows you to perform any required maintenance on an active FEP component. The failover mechanism is monitored with Keepalived that checks every six seconds the VIP-FEP connection and when one connection is down the VIP will connect to another FEP component located on another server and traffic is migrated to the newly recognized FEP component. Upon completing the maintenance activity and restarting the first FEP component, the newly recognized FEP component remains the active FEP component.

To migrate the traffic:

2. Stop the FEP on the server that you want to perform maintenance/ not have traffic running:

   a. Connect to the relevant server using SSH.

   b. Run this command: **# monit stop <FEP name>**

After a few seconds, the Keepalived mechanism will recognize that the first FEP is down, and then the VIP will connect to another FEP component and traffic will be migrated through that FEP.

3. Upon completing the maintenance activity, restart the FEP component:

a. Connect to the relevant server using SSH.

b. Run this command: **# monit start <FEP name>**

Traffic continues to be directed to the newly recognized FEP component.

8. Migrating a Server

For maintenance purposes, you can seamlessly migrate from one VM server to a new VM server within a site. As migration is on a server to server level, all the SDC components are migrated to the new server.

During the migration, the vInstaller triggers and then verifies that a graceful shutdown occurred on the source VM, before creating a new VM. The migrated server components' configuration is not changed, instead the components are deleted from one VM and then re-installed to a new VM server within a site.

Note: When migrating a Tripo component, you need to wait 15 seconds between two Tripo graceful shutdowns to make sure the mated Tripo is not down.

8.1 Prerequisite

Verify that you have a valid authentication token that has not expired. The authentication token is needed for the API request (X-Auth-Token:<token>). If the token is no longer valid, generate a new token as described in *Authenticating the SDC-Orchestrator Connection*.

8.2 Migrating API Request

The following is the API request that is sent from the Orchestrator to the vInstaller to migrate a server:

Note: As there is one SDC component per VM, you can identify the SDC component that you want to migrate by identifying the relevant source VM as <server> in the API request.

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d
fun="traffix.scale" -d role='migrate' -d hosts='<server>'
```

8.2.1 Monitoring the migrate API Request

The following are the command and status codes to monitor the API request flow.

8.2.1.1 Command Error Codes – migrate API Request

The following are the command execution error codes for migrating a component:

Table 18: Migrate Command Error Codes

Exit Code	Description
-10	%s is invalid option
-11	There is another Orchestrator flow currently running on the VNF
-12	Last installation request failed - check logs
-13	ERROR - allowed to migrate only one host
-14	ERROR – did not receive hosts to migrate
-15	ERROR - host is not in topology
-16	ERROR - unable to migrate itself
-17	Failed to connect to DB
-31	Error occurred when tried to lock flow

8.2.1.2 Command Success Codes – migrate API Request

The following are the command execution success codes for migrating a component:

Table 19: Migrate Command Success Codes

Return Code	Description
0	initiateVmShutDown: Graceful shutdown request sent to <vm names>

8.2.1.3 Request Status Codes – migrate API Request

The following are the status codes for a migration request flow.

Table 20: Migrate API Request Status Output Codes

Request Flow Status	Code	Description
Start	14005 – success	Pending internal operation by vInstaller

Request Flow Status	Code	Description
Graceful shutdown of SDC VM	13000 – success 15006 – failure	vInstaller verifies from the Orchestrator that a graceful shutdown occurred
Initializing the VM (on target site)	14002 – success 15002 – failure	vInstaller verifies if the VM is up and running on the target site
Installing the SDC Components on the VMs	14004 – success 15003 – failure	vInstaller verifies if the SDC components (with their applications) are successfully added to the VMs
Initializing the SDC components	12000 – success 15004 – failure	vInstaller verifies if the SDC components are up and running

9. Stopping all SDC Services

You can stop all services (with a graceful shutdown) from running on all the SDC components in a site, without removing them from the site topology file. The Orchestrator asks the vInstaller to stop all the services running on a site's components. The vInstaller immediately returns with an acknowledgement. Then the vInstaller monitors the VMs to see if the services were stopped successfully.

The two APIs for stopping all SDC services is the siteStop API request and then the siteStop Status request to verify if the services were stopped.

Note: Use the siteStop API request if you want to temporarily stop all services, as for maintenance purposes. If you want to perform an upgrade, see *Upgrading Site by Site*

. If you want to perform a rollback, use the siteRecovery API request (see *Restarting Site Services*). If you want to permanently shut down a site, then use the siteDecommission API request (see *Removing all SDC Components*).

9.1 Prerequisite

Verify that you have a valid authentication token that has not expired. The authentication token is needed for the API request (X-Auth-Token:<token>). If the token is no longer valid, generate a new token as described in *Authenticating the SDC-Orchestrator Connection*.

9.2 siteStop API Request

The following is the API request that is sent from the Orchestrator to the vInstaller to stop all services:

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d fun
"traffix.siteStop"
```

9.2.1 Monitoring the siteStop API Request

The following are the command and status codes to monitor the API request flow.

9.2.1.1 Command Error Codes – siteStop

The following are the command execution error codes for siteStop request.

Table 21: siteStop Command Error Codes

Exit Code	Description
-11	There is another Orchestrator flow currently running on the VNF
-12	Last installation request failed - check logs
-17	Failed to connect to DB
-18	Failed to update DB: flow was taken by another master
-31	Error occurred when tried to lock flow

9.2.1.2 Command Success Codes – siteStop

The following are the command execution success codes for siteStop request.

Table 22: siteStop Command Success Codes

Return Code	Description
0	siteStop request has been accepted

9.2.1.3 Request Status Codes – SiteStop API Request

The following are the status codes for a SiteStop request flow.

Table 23: siteStop API Request Status Output Codes

Request Flow Status	Code	Description
Start and pending Stop command	14020 – success	Pending internal operation by vInstaller
SiteStop command completed	12020– success 15020 – failure	vInstaller verifies that a SiteStop process has finished

9.3 siteStop Status API Request

The following is the API request to confirm that all the services from all SDC components have stopped:

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d fun="traffix.siteStop Status"
```

9.3.1 Monitoring the siteStop StatusAPI Request

The following are the command and status codes to monitor the API request flow.

9.3.1.1 Command Error Codes – siteStop Status

The following are the command execution error codes for siteStopStatus request.

Table 24: siteStop Status Command Error Codes

Exit Code	Description
-17	Failed to connect to DB
-22	No decommission flow is running

9.3.1.2 Command Success Codes – siteStopStatus

The following are the command execution error codes for siteStop Status request.

Table 25: siteStop Status Command Success Codes

Return Code	Description
14020	siteStop process still running
15020	siteStop request failed
12020	siteStop process finished successfully

10. Restarting Site Services

With the siteRecovery API Request, you can restart site services that had previously been stopped.

10.1 siteRecovery API Request

The following is the API request to return all the services that had previously been stopped:

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token: <Token>" -d client="runner" -d fun="traffix.siteRecovery"
```

10.1.1 Monitoring the siteRecovery API Request

The following are the command and status codes to monitor the API request flow.

10.1.1.1 Command Error Codes – siteRecovery

The following are the command execution error codes for siteStopRecovery request.

Table 26: siteRecovery Command Error Codes

Exit Code	Description
-17	Failed to connect to DB
-22	No site stop flow was running - not able to perform recovery
-23	site stop flow failed - not able to perform recovery
-24	site stop flow failed - not able to perform recovery
-18	Failed to update DB: flow was taken by another master
-31	Error occurred when tried to lock flow

10.1.1.2 Command Success Codes – siteRecovery

The following are the command execution error codes for siteStopRecovery request.

Table 27: siteRecovery Command Success Codes

Return Code	Description
0	Site recovery request has been accepted

11. Removing all SDC Components

You can remove all SDC components from a site with the siteDecommission API request.

Note: It is recommended that the siteDecommission API request only be applied in multi-site environments.

After decommissioning, no SDC service will be provided from the site. If you want to have the option of performing a rollback use the siteStop API request.

During the decommission process, the vInstaller triggers and then verifies that a graceful shutdown of all SDC components has occurred on all the VMs (except the vInstaller VM).

Note: Upon concluding decommissioning, you should shut down and remove the vInstallers.

11.1 Prerequisite

Verify that you have a valid authentication token that has not expired. The authentication token is needed for the API request (X-Auth-Token:<token>). If the token is no longer valid, generate a new token as described in *Authenticating the SDC-Orchestrator Connection*.

11.2 siteDecommission API Request

The following is the API request that is sent from the Orchestrator to the vInstaller to remove all SDC components:

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d fun="traffix.siteDecommission"
```

11.2.1 Monitoring the Status of a siteDecommission API Request

The following are the command and status codes to monitor the API request flow.

11.2.1.1 Command Error Codes – Decommission API Request

The following are the command execution error codes for decommission request.

Table 28: siteDecommission Command Error Codes

Exit Code	Description
-11	There is another Orchestrator flow currently running on the VNF
-12	Last installation request failed - check logs
-17	Failed to connect to DB
-18	Failed to update DB: flow was taken by another master
-31	Error occurred when tried to lock flow

11.2.1.2 Command Success Codes – Decommission API Request

The following are the command execution error codes for decommission request.

Table 29: siteDecommission Command Success Codes

Return Code	Description
0	site decommission request has been accepted

11.2.1.3 Request Status Codes – Decommission API Request

The following are the status codes for a decommission request flow.

Table 30: siteDecommission API Request Status Output Codes

Request Flow Status	Code	Description
Start	14020 – success	Pending internal operation by vInstaller
Graceful shutdown of SDC VM	13000 – success 15006 – failure	vInstaller verifies from the Orchestrator that a graceful shutdown occurred
Removed all SDC components from a site	12020 – success 15020 – failure	vInstaller verifies that decommission, all SDC components, have been successfully removed

11.3 decommissionStatus API Request

The following is the API request to confirm that all the removed components show as deleted in the Topology keyspace in the Cassandra database:

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d fun="traffix.decommissionStatus"
```

11.3.1 Monitoring the Status of a decommissionStatus API Request

The following are the command and status codes to monitor the API request flow.

11.3.1.1 Command Error Codes – decommissionStatus API Request

The following are the command execution error codes for decommissionStatus request.

Table 31: decommissionStatus Command Error Codes

Exit Code	Description
-17	Failed to connect to DB
-22	No decommission flow is running

11.3.1.2 Command Success Codes – decommissionStatus API Request

The following are the command execution success codes for decommissionStatus request.

Table 32: decommissionStatus Command Success Codes

11.3.1.3 Request Status Codes – decommissionStatus API Request

The following are the status codes for a decommission request flow.

Return Code	Description
14020	Decommission process still running
15020	Decommission request failed
12020	Decommission process finished successfully

12. Backing up and Restoring a Site

The backup procedure provides a way for you to back up an SDC site's Cassandra data. In the event that an SDC site shuts down, and you install a new site to replace the old one, you can restore the backed up Cassandra data to the newly replaced site.

During site restoration, the SDC services are migrated to a new site, while maintaining the services on the original geo-redundant site. This means the new site is installed with a new QCow 2 image and a site topology file.

12.1 Prerequisite

The restore procedure assumes that the deployment is geo-redundant, so that when services are stopped on one site they can continue on the geo-redundant site.

12.2 Backing up the Data

The following steps explains how to back up the site data.

To back up the site data:

1. Copy the site's external storage (cinder volumes) to separate cinder volumes.
2. On each master Installer server, backup the /srv/salt/<> folder.

The data is now backed up.

12.3 Restoring the Data

When restoring a site, you can also restore the Cassandra data that was previously backed up.

Note: The site topology file for the restored site must have the same interconnect and signaling IP addresses as was previously defined for it to be synched with the XML schema saved in the external storage. This supports the synchronization of Tripos, CPF and FEP components between existing and restored site. It also ensures that the new site will point

to the cloned external storage (cinder volumes that store the Cassandra data). Only the management IP addresses are changed for the new servers in the new site.

To restore the backed up data to the newly restored site's OAMs:

1. Install the site from scratch.

   Once the site is up, there is automatic Tripo synchronization from the geo-redundant site (Site 2). The geo-redundant site (Site2) and the new site point to the cloned external storage (backed up Cassandra data). All the relevant data will be restored from cinders that are connected to the new setup

2. Compare the backed up Salt file with the newly created Salt file from the installation and if any differences are found, copy any changed files from the backed up folder to the new folder (/srv/salt/ folder) for each master Installer server.

3. On one of the master Installer servers, run the following commands:

   /etc/init.d/salt-master restart

   salt "*" state.highstate

   The site configuration data is now restored on the newly restored site.

13. Backing up and Restoring a Cassandra Database

This backup and restore procedure provides a way for you to back up an SDC site's Cassandra data. In the event that Cassandra stops working, you can restore the backed-up Cassandra data.

13.1 Backing up a Site's Cassandra Database

The following steps explains how to back up the Cassandra database by creating a snapshot.

To back up the Cassandra data:

1. Create Cassandra backup snapshots for each OAM Cassandra with a <backup name>:

   /opt/cassandra/bin/nodetool -h localhost -p 7199 snapshot -t <backup-name>

   Under */data/cassandra/data*, each table in each Cassandra keyspace folder, now includes the new snapshots named with the user given name <backup-name>.

2. Go to */data/cassandra/data/<keyspace_name>/<table_name>* that has the newly created snapshots and copy the snapshots to a backup location.

Note: To ensure full backup, it is recommended to copy the newly created snapshots to a separate backup server.

13.2 Restoring the Cassandra Database

Assuming you have backed up the Cassandra data, you will be able to restore the data in the event that Cassandra shuts down unexpectedly.

To restore the backed up Cassandra data:

1. Perform the following steps on each OAM database:

c. **monit stop Cassandra_instance_name**

d. Clear all files in the commitlog directory:

rm -rf /data/cassandra/commitlog/*

e. Delete all *.db files in:
data_directory_location/keyspace_name/table_name directory.

Note: Do not delete the */snapshots and /backups* subdirectories:

find /data/cassandra/data -name "*\.db" | egrep -v "snapshot|backup" | xargs rm –rf

---

f. Copy each table snapshot folder content into the relevant table directory:

/data/cassandra/data/<keyspace_name>/<table_name>.

Note: Check each keyspace directory, in each table, in each sub-folder, for the relevant snapshots. For example, a new snapshot, named "latest" will be saved in each table in each keyspace:

/data/cassandra/data/topology/webui-4243235252/snapshots/latest/

---

g. **monit start cassandra**

2. After all the OAM servers are up and running, run the following command on one of the OAM servers:

/opt/cassandra/bin/nodetool repair

3. Wait for the process to finish.

4. Verify that the Cassandra data is restored:

a. Run **monit summary** on all the OAM servers

b. Verify that each OAM process is up and running.

14. Upgrading Site by Site

You can upgrade from this release to a later release, including to a cumulative fix (CF) or to another major release, by performing a site by site upgrade.

Note: The upgrade procedure supported in this release is only for SDC deployments.

This upgrade procedure assumes that the deployment is geo-redundant, so that when services are stopped on one site they can continue on the geo-redundant site.

14.1 What Happens During a Site by Site Upgrade?

During a site by site upgrade, the SDC services are migrated to a new site, while maintaining the services on the original geo-redundant site. This means the new site is installed with a new QCow 2 image and a site topology file.

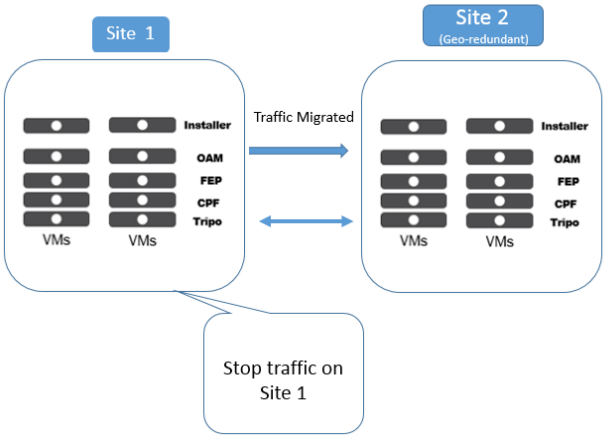
Note: The site topology file for the upgraded site must have the same interconnect and signaling IP addresses as was previously defined for it to be synched with the XML schema saved in the external storage. This then supports the synchronization of Tripos, CPF and FEP components between existing and upgraded site. It also ensures that the new site will point to the cloned external storage (cinder volumes that store the Cassandra data). Only the management IP addresses are changed for the new servers in the new site.

Note: The site topology file must be manually updated to reflect any changes made to the site topology after it was initially installed. For example, if CPF components were added after the site was installed, their information must be manually added to the site topology file before beginning the upgrade process.

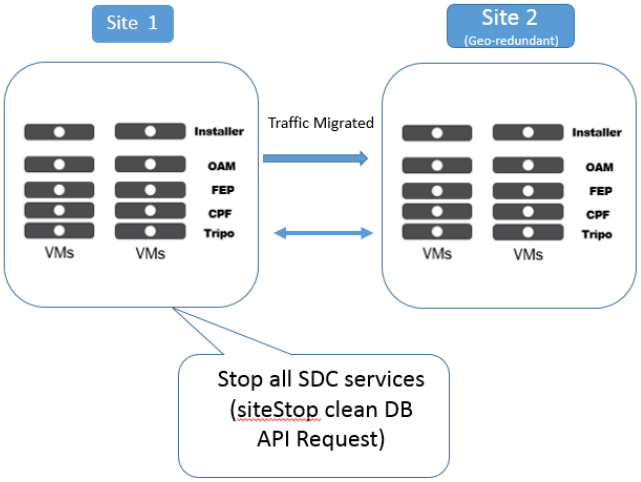
In the initial phase of the upgrade, the deployment is in mix-mode, as Site 2 runs on the old SDC release version and Site 3 runs on the new SDC release version.

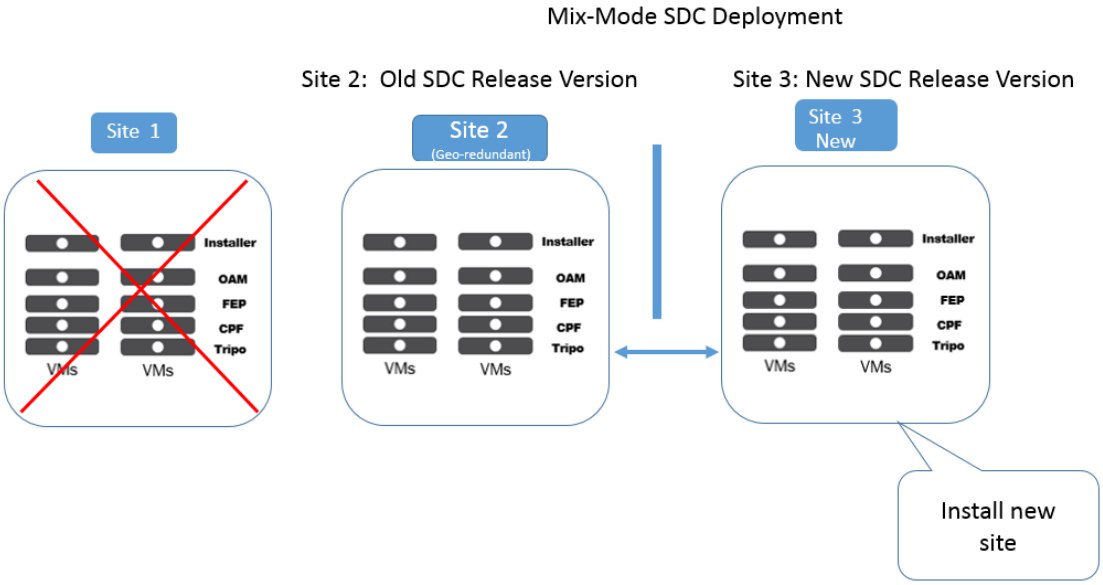
The process is illustrated and described below:

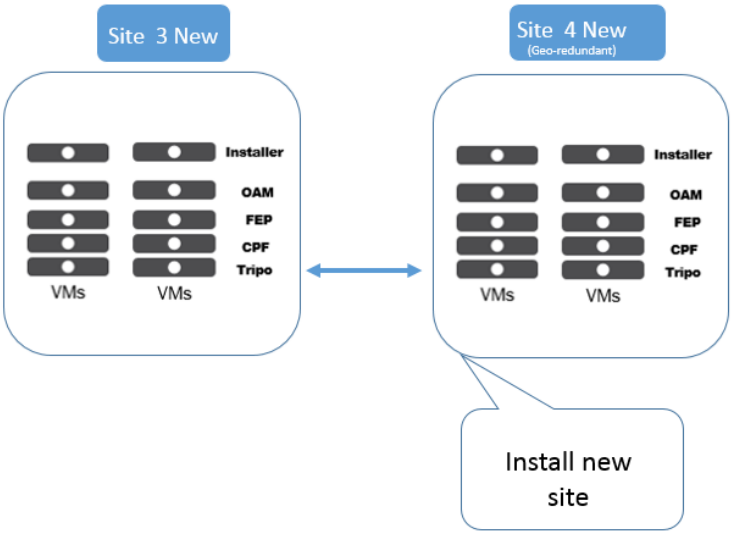
Existing SDC Deployment



Existing SDC Deployment







Note: You should perform the site by site upgrade during a defined maintenance window, to have minimal impact on services.

14.2 Steps for Shutting Down Site 1

There are three steps to shutting down Site 1 that must be done before uploading the new upgraded site.

14.2.1 Redirecting Traffic from Site 1 to Geo-Redundant Site

Per your deployment, redirect all traffic from Site 1 to Site 2, the geo-redundant site.

Check that traffic was automatically moved to the geo-redundant site.

14.2.2 Creating the topology.nameserver table

Note: When upgrading from before CF17 to CF 17 or higher, you need to create the topology.nameserver table. If you are upgrading from CF 17 or higher, the topology.nameserver table is automatically included.

You first need to verify if the topology.nameserver table exists, and if not, you need to create it.

To verify and create the topology.nameserver table:

1. From the Cassandra CLI, run the following command:

   exec cassandra cql:

   /opt/cassandra/bin/cqlsh <ip address>

2. Check that the topology.nameserver table exists:

   SELECT * FROM topology.nameserver;

   If the topology.nameserver table does not exist, the following error is displayed:

   InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured table nameserver

3. If the topology.nameserver table does not exist, run the following command:

exec query to create topology.nameserver table

```
CREATE TABLE topology.nameserver (
    "siteId" text,
    "index" int,
    ip text,
    PRIMARY KEY ("siteId", "index")
) ;
```

4. Check that the topology.nameserver table was successfully created:

SELECT * FROM topology.nameserver ;

If validated, then you should see a table with Id | index | ip fields.

14.2.3 Stopping all SDC Services on Site 1

The siteStop clean DB=True API request stops all the services running on Site 1.

As part of stopping the services on Site 1, the site topology file database is backed up in external storage as a backup file.

14.2.3.1 Site by Site Upgrade API Request

For a site by site upgrade, use the siteStop API request with the following parameter: clean DB. The <master IP address> must relate to Site 1.

14.2.3.1.1 Prerequisite

Verify that you have a valid authentication token that has not expired. The authentication token is needed for the API request (X-Auth-Token:<token>). If the token is no longer valid, generate a new token as described in Authenticating the SDC-Orchestrator Connection.

```
curl -ksi https:// <master IP address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d
fun="traffix.siteStop" -d cleanDB=True
```

The following is the response for successfully stopping the services on a site.

```
return:
- - 0
  - siteStop request has been accepted with db backup
```

14.2.3.1.2 Monitoring the Site by Site Upgrade API Request

The command and status codes to monitor this API request flow are the same as those for the siteStop API request (see *Monitoring the siteStop API Request*).

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d
fun="traffix.siteStopStatus"
```

The following is the response for successfully stopping the services on a site.

```
Waiting for return:
return:
- - 12020
  - siteStop process finished successfully
```

Note: When upgrading from CF-12 (or below), you need to run a nodetool command after sending a siteStop request:

Send the siteStop request.

Execute the following command only on the OAM machines: **/opt/cassandra/bin/nodetool -h localhost decommission**

14.2.4 Shutting down on Site 1

You need to shut down services on each of the Master Installer VMs and OAM VMs in Site 1.

To shut down the services:

1. Execute the following command: **monit stop all**
2. From the OpenStack controller, shutdown all SDC VMs in Site 1.
3. Verify that all of Site 1's machines are down.

14.3 Copying the External Storage

It is recommended that you copy Site 1's external storage (cinder volumes) once all of Site 1's virtual machines are shut down and the SiteStop flow ends successfully. In this way, the Cassandra data is cloned and will be automatically related to the new site, once the new site is installed.

14.4 Installing the Upgraded Site 3

The site with the upgraded version with the new QCOW2 images and Site Topology File (provided byF5) is now ready to be installed. For more information on how to install a new site, see *Installing a New Site*.

Note: The site topology file for Site 3 must have the same interconnect and signaling IP addresses as were defined for Site 1. Only the management IP addresses are changed for the new upgraded servers.

Once Site 3 is up, there is automatic Tripo synchronization from the geo-redundant site (Site 2) to Site 3 and the new site points to the cloned external storage (backed up Cassandra data). At this phase, the deployment in running in mix-mode, Site 2 is running on the older SDC release version and Site 3 is running on the upgraded SDC release version.

14.5 Installing the Upgraded Geo-redundant Site

Once traffic is running as expected to Site 3, and you no longer want to run your deployment in mix-mode, then install the upgraded version on Site 4.

Install the new version on Site 4, with the new QCOW2 images and site topology File (provided by F5) is now ready to be installed. For more information on how to install a new site, see *Installing a New Site*.

14.6 Performing an Upgrade Rollback

You can rollback changes made during the upgrade procedure or once the new site, Site 3, is up and running you can rollback changes made during the upgrade procedure.

To perform a rollback once Site 3 is up and running:

1. Remove all network traffic from Site 3– command per customer deployment.
2. Shutdown Site 3 with the siteStop API request. For more information about this API request, see *siteStop API Request*.
3. Start Site 1.
4. Run **monit start all** on Site 1 Master and OAM.
5. Execute siteRecovery API request on site 1. For more information about this API request, see *siteRecovery API Request*.
6. Wait for an automatic Tripo synchronization from geo-redundant site (Site 2) to Site 1.

15. Monitoring an SDC Deployment Status

The Orchestrator, through specific REST APIs, can query the status of a VM and its supported applications (i.e. Web UI, Configuration Manager, NMS, Cassandra) from the vInstaller VM.

15.1 Prerequisite

Verify that you have a valid authentication token that has not expired. The authentication token is needed for the API request (X-Auth-Token:<token>). If the token is no longer valid, generate a new token as described in *Authenticating the SDC-Orchestrator Connection*.

15.2 Application Status per VM

This API request checks the status of a specific VM. The response includes the relevant status code for successfully installed applications. In addition, as with all other API requests, there are related command execution codes.

15.2.1 appStatus API Request

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d fun="traffix.appStatus" -d tgt="*"-d apps=True (optional for apps list)
```

15.2.1.1 Command Execution Codes for appStatus API Request

Table 33: appStatus Command Error Codes

Exit Code	Description
-50	Failed to validate site topology file - check site topology file
-51	Installation not started yet
-52	Could not get information from DB

15.2.1.2 Return Codes for appStatus API Request

Table 34: appStatus Return Codes

Exit Code	Description
14002	Pending VM Start
14003	Pending SDC Installation
14004	Pending SDC Start
14006	Pending SDC Stop
15002	Fail VM Start
15003	Fail To Install SDC
15004	Fail To Start SDC
15006	Failed To Stop SDC
13000	Suspended
12000	Successfully installed

15.2.1.3 Status Query Answer Example

The following is an example of an answer from the vInstaller to a Status Query. See *Table 34: appStatus Return Codes* for more information about the returned status code.

```
HTTP/1.1 200 OK
Content-Length: 51
Access-Control-Expose-Headers: GET, POST
Access-Control-Allow-Credentials: true
Vary: Accept-Encoding
Server: CherryPy/3.2.2
Allow: GET, HEAD, POST
Cache-Control: private
Date: Tue, 18 Aug 2015 17:49:56 GMT
Access-Control-Allow-Origin: *
Content-Type: application/x-yaml
Set-Cookie: session_id=80627af4346afc7c04b187cd9aab6b4d2cbc56d0;
expires=Wed, 19 Aug 2015 03:49:56 GMT; Path=/
```

```
return:
- <VM>:
    status-code: '<Status code, e.g., 12000>'
    installed-apps:
                                                        - webui
                                                        - cm
                                                        - nms
                                                        - Cassandra
```

15.3 Site Status

This API request checks the status of all VMs within a site. The response includes the relevant status codes for the successfully installed applications on all VMs within a site. In addition, as with all other API requests, there are related command execution codes.

15.3.1 siteStatus API Request

The following is the API siteStatus request:

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d
fun="traffix.siteStatus" -d apps=True (optional for apps list)
```

15.3.1.1 Command Execution Codes for siteStatus API Request

Table 35: siteStatus Command Error Codes

Exit Code	Description
-50	Failed to validate site topology file - check site topology file
-51	Installation not started yet
-52	Could not get information from DB

15.3.1.2 Return Codes for siteStatus API Request

Table 36: siteStatus Return Codes

Exit Code	Description
14010	Installation is Running
15010	Installation Failed
12010	Installation Finished Successfully

15.3.1.3 siteStatus Answer Example

The following is an example of an answer from the vInstaller to a Site Status Query (apps =True). See *Table 36: siteStatus Return Codes* for more information about the returned status code.

```
Result:

HTTP/1.1 200 OK
Content-Length: 613
Access-Control-Expose-Headers: GET, POST
Access-Control-Allow-Credentials: true
Vary: Accept-Encoding
Server: CherryPy/3.2.2
Allow: GET, HEAD, POST
Cache-Control: private
Date: Tue, 15 Dec 2015 15:32:11 GMT
Access-Control-Allow-Origin: *
Content-Type: application/x-yaml
Set-Cookie: session_id=50479aecb5a14f333a37a0656c8b2694dc2ad4d8;
expires=Wed, 16 Dec 2015 01:32:11 GMT; Path=/

return:
- - - Site-Status-Code: 12010
    - Installed-Apps:
        sdclab005-16-cpf-1:
        - cpf1
```

```
        sdclab005-16-fep-1:
        - fep1
        sdclab005-16-master-1:
        - oamDB-unique
        - vnf
        sdclab005-16-master-2:
        - oamDB-unique
        - vnf
        sdclab005-16-oam-1:
        - CM1
        - NmsAgent1
        - Cassandra1
        - WebUI1
        sdclab005-16-oam-2:
        - CM1
        - NmsAgent1
        - Cassandra1
        - WebUI1
        sdclab005-16-tripo-1:
        - tripo1
        sdclab005-16-tripo-2:
        - tripo1

:49:56 GMT; Path=/

return:
-                                                    <VM name>:
    <app>:
    - installed
```

15.4 Application Execution Status per VM

This API request checks if a specific application was already started successfully on a specific VM. Specifically, it checks if the application's process is up by querying "Monit" on the machine.

15.4.1 Application Status API Request

The following is the API request to check the application status:

```
curl -ksi https://<master IP address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token: <Token>" -d client='local' -d tgt='<VM name>*'
-d fun='monit.summary' -d arg='<hostname-instance, e.g., vm105-fep-
102>' | no arg given
```

15.4.1.1 Answer Example

The following is an example of an answer from the vInstaller to a Query Type fun='monit.summary'.

```
HTTP/1.1 200 OK
Content-Length: 58
Access-Control-Expose-Headers: GET, POST
Access-Control-Allow-Credentials: true
Vary: Accept-Encoding
Server: CherryPy/3.2.2
Allow: GET, HEAD, POST
Cache-Control: private
Date: Tue, 18 Aug 2015 17:57:53 GMT
Access-Control-Allow-Origin: *
Content-Type: application/x-yaml
Set-Cookie: session_id=80627af4346afc7c04b187cd9aab6b4d2cbc56d0;
expires=Wed, 19 Aug 2015 03:57:53 GMT; Path=/
  return:
-                                                    <VM name>:
    Process:
      <app>: Running
```

16. Using Logs for Troubleshooting

You can refer to the following logs for troubleshooting the installation process:

Log	Troubleshoot problems with…
/var/log/salt-install.log	installation
/var/log/salt/master	python script, uploading topology to Cassandra, traffix api, communication problems with minion servers
/var/log/salt/minion	state logs, installation, configurations, communication with the master Installer
/var/log/rsyslog	verification of file creation, per the OAM node, central storage of logs

17. Configuring SDC-Server Routes

Using API commands, you can add or remove routes when servers are added/removed with networks that were not previously recognized by the SDC. The configuration is on the interface network signaling level as these changes affect routes between the FEP and external servers.

17.1 Add Routes

The Orchestrator sends an addRoutes API Request to the master vInstaller to add the route to the server. The addRoutes API request refers to the new XML snippet file (@addroutes.xml) and by doing so the newly created XML is saved and merged with the site topology file in Cassandra.

17.1.1 Prerequisites

17.1.1.1 Verifying the Authentication Token

Adding a route is performed with an API request. To apply the REST API, you need to have a valid authentication token that is not expired.

If the authentication token is expired, you need to request a new one (see *Authenticating the SDC-Orchestrator Connection*).

17.1.2 Updating the Site Topology File with Added Route

The site topology file must be updated to include definitions for the added route. This is done by creating a new topology XML file (a snippet) that is then uploaded to the Cassandra database and merged with the site topology file, once the addRoutes API flow request is completed.

To create a new topology XML file:

1. Fill in the following parameters:
   - vm name
   - interface network

Note: The vm name is the name of the server that the route is being added to.

- route name, net4, net6, ip4sub, ip6sub, gateway

Refer to the *F5 SDC Guidelines for Creating a Site Topology File* for more information about these parameters.

The following is an example of a Topology file for adding a route on a server:

```
hosts=<?xml version="1.0" encoding="UTF-8"?>
<topology>
        <vms>
                <vm name="sdclab005-12-fep-1">
                        <interfaces>
                                <interface network="sig-1"
ip4="10.1.81.164" ip6="" dev="eth2" name="sig-1-ip1">
                                        <route name="fep1-vip1-route1"
net4="10.1.19.0" net6="" ip4sub="255.255.255.0" gateway="10.1.81.1"/>
                                </interface>
                        </interfaces>
                </vm>
        </vms>
</topology>
```

17.1.3 addRoutes API Request

The following is the addRoutes API request:

```
# curl -ksi https://<master_IP_address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d
fun="traffix.addRoutes" -X POST -d @addroutes.xml
```

17.2 Delete Routes

The Orchestrator sends a deleteRoutes API Request to the master vInstaller to delete the route to the server. The deleteRoutes API request refers to the new XML snippet file

(@deleteroutes.xml) and by doing so the newly created XML is saved and merged with the site topology file in Cassandra.

17.2.1 Prerequisites

17.2.1.1 Verifying the Authentication Token

Deleting a route is performed with an API request. To apply the REST API, you need to have a valid authentication token that is not expired.

If the authentication token is expired, you need to request a new one (see *Authenticating the SDC-Orchestrator Connection*).

17.2.2 Updating the Site Topology File with Deleted Route

The site topology file must be updated to remove the previously defined route network definitions for the deleted route. This is done by creating a new Topology XML file (a snippet) that is then uploaded to the Cassandra database and merged with the site topology file, once the deleteRoutes API flow request is completed.

To create a new Topology XML file:

1. Fill in the following parameters:
   - vm name
   - interface network

Note: The vm name is the name of the server that the route is being deleted from.

   - route name, net4, net6, ip4sub, ip6sub, gateway

Refer to the *F5 SDC Guidelines for Creating a Site Topology File* for more information about these parameters.

17.2.3 deleteRoutes API Request

The following is the deleteRoutes API request:

```
# curl -ksi https://<master_IP_address>:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d fun="traffix.deleteRoutes" -X POST -d @addroutes.xml
```

17.3 Adding and Deleting Routes via Script

The addRoute/deleteRoute script can be applied when you want to add or delete routes one by one or a group of routes.

To add/delete a route:

1. Change the directory on one of the master Installer servers:

   cd/srv/traffix/pillar/

   The following script is included for adding/deleting a single route:

```
./route-api.py -r {addRoutes/deleteRoutes} -v 4/6 -i {source IP} -n {destination Net} -m {destination Netmask} -g {GATEWAY} -N {Name}
```

2. Customize the script by selecting the following:

<addRoutes/deleteRoutes>

a. Fill in the following parameters:

i. < 4/6>: IP version 4 or 6

ii. <source IP>

iii. <destination Network>

iv. <destination netmask>: destination network mask

v. < GATEWAY >: source IP gateway

vi. < route name >:-N {Name}

The following is an example of an add route script:

```
 ./route-api.py -r addRoutes -v 4 -i 10.3.118.10 -n 10.1.71.0 -m 255.255.255.0 -g 10.3.118.1 -N route1
```

3. Run the script in file process mode with the following command:

./route-api.py -f

As part of this command, the script validates the inputted data. Following validation, the selected action (add/delete) is performed.

To add/delete a group of routes:

1. Create a */tmp/input* file on the same master Installer server that you run the **cd/srv/traffix/pillar/** command.

   The */tmp/input* file follows the format of the./route-api.py -r script with the defined parameters:

```
./route-api.py -r {addRoutes/deleteRoutes} -v 4/6 -i {source IP} -n
{destination Net} -m {destination Netmask} -g {GATEWAY} -N {Name}
```

The following is an example of a group of routes to be added by the script:

```
addRoutes:4:10.3.118.11:10.1.82.0:255.255.255.0:10.3.118.1;route-1
addRoutes:4:10.3.118.10:10.1.82.0:255.255.255.0:10.3.118.1;route-2
addRoutes:4:10.3.118.12:10.1.82.0:255.255.255.0:10.3.118.1;route-3
addRoutes:4:10.3.118.13:10.1.82.0:255.255.255.0:10.3.118.1;route-4
addRoutes:4:10.3.118.14:10.1.82.0:255.255.255.0:10.3.118.1;route-5
addRoutes:4:10.3.118.3:10.1.82.0:255.255.255.0:10.3.118.1;test-route-1
addRoutes:4:10.3.118.4:10.1.82.0:255.255.255.0:10.3.118.1;test-route-2
where:
column 1 is the operation addRoutes/deleteRoutes
column 2 is the IP version 4 or 6
column 3 is the <source IP>
column 4 is the <destination network>
column 5 is the <destination network mask>
column 6 is the <source IP gateway>
column 7 is the <route name>
```

2. Run the script in file process mode with the following command:

./route-api.py -f

As part of this command, the script validates the inputted data. Following validation, the selected action (add/delete) is performed.

Note: When deleting a route, the same subnet mask IP value must be used as was included in the addRoute script. For troubleshooting, refer to the */var/log/route-api.log*.

17.4 Show Routes

The showRoutes API request provides information of all the existing network routes between a server and the SDC.

17.4.1 Prerequisites

17.4.1.1 Verifying the Authentication Token

Showing all the routes is performed with an API request. To apply the REST API, you need to have a valid authentication token that is not expired.

If the authentication token is expired, you need to request a new one (see *Authenticating the SDC-Orchestrator Connection*).

17.4.2 showRoutes API Request

The API request refers to the SDC component <hostname> in which you want to retrieve the network routes information. The request can be for more than one server. The following is the showRoutes API request:

```
# curl -ksi https://<master_IP_address>:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d fun="traffix.showRoutes" -d hosts='<hostname1>,<hostname2>'
```

17.4.2.1 showRoutes API Request Example

The following is an example of the showRoutes API request and answer based on showing the routes for one FEP-server connection.

```
# curl -ksi https://localhost:8000 -H "Accept: application/x-yaml" -H "X-Auth-Token:cd245f57008d3a0054d510d9b6d6237902440bd7" -d
```

```
client="runner" -d fun="traffix.showRoutes" -d hosts='sdclab005-12-fep-
1'

HTTP/1.1 200 OK
Content-Length: 284
Access-Control-Expose-Headers: GET, POST
Access-Control-Allow-Credentials: true
Vary: Accept-Encoding
Server: CherryPy/3.2.2
Allow: GET, HEAD, POST
Cache-Control: private
Date: Tue, 09 Feb 2016 13:32:38 GMT
Access-Control-Allow-Origin: *
Content-Type: application/x-yaml
Set-Cookie: session_id=cd245f57008d3a0054d510d9b6d6237902440bd7;
expires=Tue, 09 Feb 2016 23:32:38 GMT; Path=/

return:
- sdclab005-12-fep-1:
  - - name: fep1-vip1-route12
    - interface: sig-1-ip1
    - net4: 10.1.29.0
    - ip4sub: 255.255.255.0
    - gateway: 10.1.81.1
  - - name: fep1-vip1-route2
    - interface: sig-1-ip1
    - net4: 10.1.70.0
    - ip4sub: '24'
    - gateway: 10.1.81.1
```

18. Replacing NTP Servers

Using an API command, you can replace the NTP servers that were previously configured to synchronize time zones between other servers.

The Orchestrator sends a changeNtp API sends a request to the master Installer to replace the NTP servers. This request deletes all the existing NTP servers and replaces them with the new NTP servers that are defined in the site topology XML snippet file. The changeNtp API request refers to the new XML snippet file (@changentp.xml) and by doing so the newly created XML is saved and merged with the site topology file in Cassandra.

18.1.1 Prerequisites

18.1.1.1 Verifying the Authentication Token

Changing an NTP server is performed with an API request. To apply the REST API, you need to have a valid authentication token that is not expired.

If the authentication token is expired, you need to request a new one (see *Authenticating the SDC-Orchestrator Connection*).

18.1.2 Updating the Site Topology File with New NTP Servers

The site topology file must be updated to include the definitions for any new NTP server. This is done by creating a new Topology XML file (a snippet) that is then uploaded to the Cassandra database and merged with the site topology file, once the changeNtp API flow request is completed.

To create a new Topology XML file:

1. Fill in the following parameters:
   - ntpServer name
   - ip

Note: The ntpServers are the new NTP servers and for each one you need to define a name (ntpServer name) and IP address. You can add multiple NTP servers, but no matter how many are added, all of the previously configured NTP servers are removed.

Refer to the *F5 SDC Guidelines for Creating a Site Topology File* for more information about these parameters.

The following is an example of a Topology file for changing an NTP server:

```
ntpServers=<?xml version="1.0" encoding="UTF-8"?>
<topology>
        <siteProperties name="SDC_Site">
                <ntpServers>
                        <ntpServer name="first" ip="192.168.16.5"/>
                        <ntpServer name="second" ip="192.168.16.6"/>
                </ntpServers>
        </siteProperties>
</topology>
```

18.1.3 changeNtp API Request

The following is the changeNtp API request:

```
# curl -ksi https://<master_IP_address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d fun="traffix.
changeNtp " -X POST -d @changentp.xml
```

A successful response will include the following response:

```
return:
- 0
- Successfully changed the NTP servers
```

19. Changing the Log Servers

This release supports central logging. Each SDC component sends its logging information to the OAM machine. The OAM machines have a mount point for persistent storage. You have an option to change the log server IP addresses. This configuration is done by updating the site topology File and using the changeLogServers API request.

Note: This logging method is in addition to the direct forwarding method in which logs are forwarded from the CPF or FEP. To configure the log servers for the direct forwarding method, refer to the *F5 SDC User Guide*.

19.1 Prerequisites

The following are the prerequisites for changing the log servers.

19.1.1 Verifying the Authentication Token

Changing the log servers is performed with an API request. To apply the REST API, you need to have a valid authentication token that is not expired.

If the authentication token is expired, you need to request a new one (see *Authenticating the SDC-Orchestrator Connection*).

19.2 Updating the Site Topology File with the New Log Servers

The site topology file must be updated to include definitions for the specific configuration elements for the new log server IP addresses. This is done by creating a new Topology XML file (a snippet) that is then uploaded to the Cassandra database and merged with the site topology file, once the changeLogServers API flow request is completed. The API request refers to the XML file (@changelogservers.xml.), as shown in the *changeLogServers* API Request.

To create a new Topology XML file:

1. Fill in the following parameter for the new log servers:

   - logServer name

- Refer to the *F5 SDC Guidelines for Creating a Site Topology File* for more information about this parameter.
- The following is an example of a topology file for changing the log servers:

```
logServers=<?xml version="1.0" encoding="UTF-8"?>
<topology>
        <siteProperties name="SDC_Site">
        <logServers>
            <logServer name="peervm-118-01" ip="10.240.13.74"
type="syslog" protocol="tcp" port="10514"/>
            <logServer name="peervm-118-02" ip="10.240.13.75"
type="syslog" protocol="tcp" port="10514"/>
        </logServers>
        </siteProperties>
</topology
```

19.3 Sending a changeLogServer API Request

The changeLogServers API Request is sent from the Orchestrator to query the vInstaller to change the system log servers' IP addresses. The changeLogServers API request refers to the new XML snippet file (@changelogservers.xml) and by doing so the newly created XML is saved and merged with the site topology file in Cassandra.

19.3.1 changeLogServers API Request

The following is the changeLogServers API request:

```
curl -ksi https://<master_IP_address>:8000 -H "Accept: application/x-
yaml" -H "X-Auth-Token:<Token>" -d client="runner" -d
fun="traffix.changeLogServers" -X POST -d @changelogservers.xml
```

19.3.1.1 changeLogServers API Request Example

The following is an example of the request and answer:

```
# curl -ksi https://localhost:8000 -H "Accept: application/x-yaml" -H
"X-Auth-Token:aa6790aa67ae5ce87715b66bf5bd58fc3ea4bdb5" -d
```

```
client="runner" -d fun="traffix.changeLogServers" -X POST -d
@changelogservers.xml

HTTP/1.1 200 OK
Content-Length: 292
Access-Control-Expose-Headers: GET, POST
Access-Control-Allow-Credentials: true
Vary: Accept-Encoding
Server: CherryPy/3.2.2
Allow: GET, HEAD, POST
Cache-Control: private
Date: Tue, 20 Oct 2015 17:46:10 GMT
Access-Control-Allow-Origin: *
Content-Type: application/x-yaml
Set-Cookie: session_id=3b3f4aec793bc67d5b562d8129609bcf7045158b;
expires=Wed, 21 Oct 2015 03:46:10 GMT; Path=/

return:
- 0
- Successfully changed the remote log servers
```

Appendix A: Site Topology File Structure

The site topology configuration is defined under the topology element. This element contains four mandatory core elements, each defining a different aspect of the site topology. *Table 37* lists these elements.

Table 37: Elements Defined in Topology

Element (Mandatory/Optional)	Description
Vms (Mandatory)	This element defines and configures the specific virtual machines included in the site, the storage and SDC applications installed on them, and their communication paths. For more information, see **Virtual Machines (topology:vms)**.
Networks (Mandatory)	This element defines and configures the networks used by the applications installed on the site's virtual machines for internal and external communication. For more information, see **Networks (topology:networks)**
Applications (Mandatory)	This element defines default values for the applications that are run on the virtual machines in the site. For more information, see **Applications (topology:applications)**
siteProperties (Mandatory)	This element defines and configures site-wide values. For more information, see Site Properties (topology:siteProperties)

19.4 Virtual Machines (topology:vms)

The vms element is one of the four core elements of the site topology file. This element contains virtual machine attributes and elements. The vms element defines the virtual machines that are part of the site, the installed storage on each virtual machine, the applications (SDC components) that will run on each virtual machine, and the communication paths that the applications are going to use.

The vms element contains one or more vm elements, corresponding to the number of virtual machines in the site. Each vm element defines a specific virtual machine in the site. *Table 38* lists the attributes and elements that are defined as part of each vm element.

Table 38: Elements Defined as Part of each VM

Element (Mandatory/Optional)	Description	Guidelines
Name (Mandatory)	The name of the virtual machine.	Enter a unique name.
defaultGateway (Mandatory)	The gateway used to connect the host network to the destination network.	Enter the default gateway.
VNFC (Optional)	The name of the component.	Enter the relevant name.
Interfaces (Mandatory)	The network interfaces used by the applications installed on the specific virtual machine.	The interfaces must belong to a network defined in the networks element.
applicationInstance (Mandatory)	The applications installed on the specific virtual machine.	The values defined per VM will override the default values defined in the applications element.
Volumes (Mandatory)	The file systems and persistent storage installed for the specific virtual machine.	Refer to the relevant Installation Guide, Data Volume Requirement section for more information.

19.4.1 Interfaces (topology:vms:vm:interfaces)

The interfaces element is a sub-element of a vm element., and contains one or more interface element. Each interface element corresponds to a specific interface that will be used by the virtual machine.

Note: The interfaces must belong to the networks defined in the networks element. Verify that all necessary networks have been defined for the site in the networks element before defining specific interfaces for the virtual machines.

Each interface element contains attributes and sub-elements. *Table 39* lists the attributes that are defined as part of the interface element.

Table 39: Elements Defined in each Interface Element

Element (Mandatory/Optional)	Description	Guidelines
Network (Mandatory)	The name of the network associated with this interface.	The network name should match the name as it appears in the networks element.
ip4 (Optional)	The IPv4 address that is associated with this interface, corresponding to the relevant network.	Enter the relevant IPv4 address.
ip6 (Optional)	The IPv6 address that is associated with this interface, corresponding to the relevant network.	Enter the relevant IPv6 address.
Dev (Mandatory)	The name of the device on the guest OS that uses this interface.	If the interface is applied to a bond, define the element value with the bond name. For example, “bond0”.
bondDev (Optional)	The two interfaces that make up the bond defined in the “Dev” element.	Interfaces should be specified in comma separated format. E.g.: eth0, eht1
bondingOpts (Optional)	The bonding attributes on the OS level.	Enter the bonding interface configuration. For example, bondingOpts="mode=1 miimon=100 num_grat_arp=10 primary=eth10"
Name (Optional)	The name of the interface. This is used to differentiate between multiple IP addresses	This value must match the “listeninterfacename” value defined

Element (Mandatory/Optional)	Description	Guidelines
	defined for the same interface, or when a VIP application is configured for the VM.	for the application instance using the interface.
Route (Optional)	The route is the specific route needed to be configured for a specific interface or network.	Enter the gateway for the IP

Note: The ip4 and ip6 attributes can be configured together on the same network or separately on different interfaces according to the matching network.

19.4.1.1 Route

Table 40: Elements Defined in Route

Element (Mandatory/Optional)	Description	Guidelines
Name (Mandatory)	Route name.	Enter a unique name
net4 (Optional)	The IPv4 destination network address to which to set the route.	Enter the relevant IPv4 address.
net6 (Optional)	The IPv6 destination network address to which to set the route.	Enter the relevant IPv6 address.
ip4sub (Optional)	The IPv4 destination network subnet mask.	The subnet mask can be stated in regular IPv4 format or in CIDR notation (/32, /24, etc.)
ip6sub (Optional)	The IPv4 destination network subnet mask.	The subnet mask can be stated only in CIDR notation (/64, /127, etc.)
Gateway (Mandatory)	The gateway address from which the route is configured.	This can be configured for IPv4 or IPv6.

19.4.2 Application Instances (topology:vms:vm:applicationInstance)

The applicationInstances element is a sub-element of a vm element, and contains one or more applicationInstance element. Each applicationInstance element corresponds to a specific instance of an application (SDC component) installed on the VM.

Each applicationInstance element contains attributes and sub-elements. *Table 41* lists the attributes that are defined as part of the applicationInstance element.

Each application is configured by default with the values defined – per application type – in the applications element. To override these values for this specific instance of the application, include the specific values with the correct value in the applicationInstance element.

Table 41: Elements Defined for Each Application Instance

Element (Mandatory/Optional)	Description	Guidelines
type (Mandatory)	The type of the application.	The valid values are: 1. fep 2. vip 3. cpf 4. cpfss7 5. tripo 6. cm 7. webui 8. nms 9. oamDB 10. vInstaller 11. elk
name (Mandatory)	The name of the application instance. Each application instance should have a unique instance name to be referred to later on during the application configuration phase.	Each application instance should have a unique instance name.

IPv (Mandatory)	Specifies whether the application uses IPv4 or, IPv6	Enter either IPv4 or IPv6.
listenInterfaceName (Mandatory)	The name of the network interface (s) used by the application.	12. This value must match the name value defined for the interface that the application uses. Note: For example, for the Tripo application instance, the listenInterfaceName field must always be defined as “ic” and the IPv field must be defined according to the IP version of the ic network. 13. All interfaces that the applications are listening to must be defined. 14. For VIP applications, all interfaces with the same routerID must be defined.

19.4.3 Volumes (topology:vms:vm:volumes)

The volumes element is a sub-element of a vm element, and defines the persistent storage that will be installed on the specific virtual machines.

The volumes element should include all persistent volumes that are expected to be attached to this specific VM, and should also include file system definition for each one of them.

Table 42: Elements Defined in Volumes

Element (Mandatory/Optional)	Description	Guidelines
cinderName (Mandatory)	The name of the storage volume.	Enter a relevant name.

Element (Mandatory/Optional)	Description	Guidelines
vmDev (Mandatory)	The device name that is expected to be seen on the guest OS level. For example: E.g. /dev/DRAOAM1	Enter the device path.
Partition (Mandatory)	A list of partitions to be defined on the guest OS level.	List all relevant partition names.

19.4.3.1 Partitions (topology:vms:vm:partition)

The partitions element is a sub-element of a volumes element, and defines a list of partitions on the guest OS level.

Note: For more information about the volume requirements, see the Data Volume Requirement section in the relevant Installation Guide.

Table 43: Elements Defined in Partitions

Element (Mandatory/Optional)	Description	Guidelines
Name (Mandatory)	Logical name for the handled partition.	Enter a relevant name, such as data, oamdb, repo, backup, logs
Size (Mandatory)	Size of the partition in GB.	Enter a relevant size per the data volume requirements.
mountPoint (Mandatory)	Mount point to which this partition will be mounted. For example: /data/Cassandra	Enter a relevant mount type per the data volume requirements.
fsType (Optional)	File system type. For example: ext4	Enter a relevant file system type.
fsParams (Optional)	Any other parameters needed for file system mount command	Enter any other relevant parameters.

19.5 Networks (topology:networks)

The networks element is one of the four core elements of the site topology file. This element contains network attributes and elements. The site topology file defines the SDC components that will run on each site server, and the communication paths – both between the SDC components and between the SDC site and external networks. The networks element defines the networks that the communication paths are going to run on.

- Management networks – this network connects between SDC components the internal SDC components for management purposes and is used to communicate between the SDC components and the Orchestrator.
- Interconnect networks – this network connects between all internal SDC components within a site.
- Signaling networks – this network connects the SDC's FEP components with the external networks, including between geo-redundant sites.
  - Custom networks – this network is based on specific customer-requests and is defined on a case by case basis with the customer.

The networks element contains one or more network elements. Each network element defines a communication path for the site. *Table 44* lists the attributes that are defined as part of the network element.

Note: The networks element must include at least two network elements – one network element defining a management network and one network element defining an interconnect network.

Table 44: Elements Defined for Each Network Element

Element (Mandatory/Optional)	Description	Guidelines
Name (Mandatory)	The name of the network.	Enter a value that clearly indicates the nature of the

Element (Mandatory/Optional)	Description	Guidelines
		network. For example: sig-tcp-1 or sig-sctp-2.
net4 (Optional)	The IPv4 destination network address for this network.	Enter the relevant IPv4 address.
net6 (Optional)	The IPv6 destination network address for this network.	Enter the relevant IPv6 address.
ip4sub (Optional)	The network subnet – the range of IP addresses that belong to the IPv4 network.	The subnet mask can be stated in regular IPv4 format or in CIDR notation (/32, /24, etc.)
ip6sub (Optional)	The network subnet – the range of IP addresses that belong to the IPv6 network.	The subnet mask can be stated only in CIDR notation (/64, /127, etc.)
vlan (Optional)	The VLAN tag given for this specific network by the customer. The VLAN tag defines the VLAN-aware partitioning of the network.	Enter the relevant Vlan ID.
Role (Mandatory)	The type of communication that the network is going to be used for.	Define the value as follows: 15. For a management network, define the value as “mgmt” 16. For an interconnect management network, define the value as “ic” 17. For a signaling management network, define the value as “sig” 18. For a custom network, define the value as needed.

19.6 Applications (topology:applications)

The applications element is one of the four core elements of the site topology file. This element contains sub-elements, each sub-element corresponding to one of the applications (SDC components) that can be installed in the site. These sub-elements define the default values for the general properties for each application – what version of the application is used, the communication paths that the application will use, etc.

19.6.1 FEP

The properties defined under the FEP element are listed in *Table 45*.

Note: In most cases, the FEP properties will be defined per specific instance of the FEP application, under the applicationInstances element. Any properties define per application instance will override the default values defined here.

Table 45: Elements Defined for the FEP Application

Element (Mandatory/Optional)	Description	Guidelines
fepType (Mandatory)	The protocol that the FEP application supports.	The supported FEP types are: 19. diameter 20. http 21. radius 22. ldap The default value is “diameter”.

19.6.2 CPF

The properties defined under the CPF element are listed in *Table 46: Elements Defined for the CPF Application.*

Note: In most cases, the specific CPF application instances will use the default values defined in this element, and the only attribute that will need to be defined under the applicationInstances element is the name.

Table 46: Elements Defined for the CPF Application

Element (Mandatory/Optional)	Description	Guidelines
cpfPort (Mandatory)	The CPF port used by the Diameter FEP.	The default value is 13868.
cpfRadiusPort (Optional)	The CPF port used by the RADIUS FEP.	The default value is 11812.
cpfHttpPort (Optional)	The CPF port used by the HTTP FEP.	The default value is 18080.
cpfGtpPort (Optional)	The CPF port used by the GTP FEP.	The default value is 13386.
cpfLdapPort (Optional)	The CPF port used by the LDAP FEP.	The default value is 1389.

19.6.3 Tripo

The elements defined under the tripo element are listed in *Table 47*. These elements define the default values used by instances of the Tripo application in the site.

Note: Configuring the tripo element under the applications element is optional.

When defining the Tripo application instance, the listenInterfaceName must always be defined as “ic” and the IPv field must be defined according to the defined IP version (IPv4/Ipv6) of the ic network.

For example: <applicationInstance type="tripo" name="tripo1" IPv="IC_IPv" listenInterfaceName="ic">

Table 47: Elements Defined for the Tripo Application

Element (Mandatory/Optional)	Description	Guidelines
secondSiteIP1 (Mandatory)	The IP address of the first Tripo instance on the second SDC server.	Enter the relevant IP address. Note: When this IP address is 1.1.1.1, replication between sites is disabled.
secondSiteIP2 (Mandatory)	The IP address of the second Tripo instance on the second SDC server.	Enter the relevant IP address. Note: When this IP address is 1.1.1.1, replication between sites is disabled.
maxSessions (Mandatory)	The maximum number of records allowed to be loaded per Tripo instance.	The value can be between 10,000 and 170,000,000.
tripoVersion (Mandatory)	The Tripo product version.	The available values are: 23. latest 24. stated version The default is "latest".
srrInterface (Mandatory)	The name of the network that the Tripo uses to communicate with the Tripo instances on the geo-redundant site to replicate session data between sites.	The name must match the name of the network as it is defined in the networks element.
srrListenPort (Mandatory)	The port on the Tripo instance on the geo-redundant site that is used for replications.	It is assumed that both Tripo mates are using the same port.

Element (Mandatory/Optional)	Description	Guidelines
secondSiteSrrPort (Mandatory)	The port on the Tripo instance on the geo-redundant site that is used for replications.	It is assumed that both Tripo mates are using the same port.

19.6.4 CM

The cm element is currently defined as part of the applications element without any sub-elements or attributes.

19.6.5 Webui

The webui element is currently defined as part of the applications element without any sub-elements or attributes.

19.6.6 Nms

The nms element is currently defined as part of the applications element without any sub-elements or attributes.

19.6.7 oamDB

The elements defined under the oamDB element are listed in *Table 48*. These elements define the default values used by instances of the oamDB application in the site.

Note: In most cases, the oamDB properties will also be defined per specific instance of the FEP application, under the applicationInstances element. Any properties define per application instance will override the default values defined here.

Table 48: Elements Defined for the oamDB Application

Element (Mandatory/Optional)	Description	Guidelines
networkName (Mandatory)	The name of the network that the oamDB uses for internal communication and heartbeat.	The name must match the name of the network as it is

Element (Mandatory/Optional)	Description	Guidelines
		defined in the networks element. The default network is a network defined as a mgmt network.

19.6.8 vInstaller

The vInstaller element is currently defined as part of the applications element without any sub-elements or attributes.

19.6.9 VIP

The VIP application is used to enable active-standby mode for FEP applications, by associating a dedicated VIP application instance with each FEP IP address. The elements defined under the vip element are listed in *Table 49*. These elements define the default values used by instances of the vip application in the site.

Note: Verify that a dedicated VIP application is associated with each IP running on the FEP applications supported in active-standby mode. For example, to support two FEP applications using the TCP protocol, two VIP applications must be configured. To support two FEP application using the SCTP protocol, four VIP applications must be configured.

Note: Configuring the vip element under the applications element is optional.

Table 49: Elements Defined for the vip Application

Element (Mandatory/Optional)	Description	Guidelines
routerID (Optional)	A numeric value representing a group of one or more VIP	Verify that the same value is defined for all VIP applications associated with the FEP

Element (Mandatory/Optional)	Description	Guidelines
	applications that will operate in active-standby mode.	applications that will operate in active-standby mode.
transportProtocol (Optional)	Indicates whether the VIP is managing TCP or SCTP transport protocol.	Valid values are: 25. tcp 26. sctp
applicationInstanceName (Optional)	The name of the application that the VIP is attached to. The VIP application is only valid if it is attached to an application. This attribute defines which application will be monitored for VIP failovers.	The value must be identical to the value defined for the name attribute of an application instance defined for this VM.

19.6.10 ELK

Note: Configuring the elk element under the applications element is only mandatory when the site is part of a deployment managed by an EMS site. Adding the ELK application instance to the topology.xml, will trigger the installation and configuration of the ELK components (on the EMS site: Fluentd, Elasticsearch and Kibana and on the SDC site, Fluentd).

When defining the ELK application instance, the listenInterfaceName must always be defined as “mgmt” and the IPv field must be defined according to the defined IP version (IPv4) of the mgmt network:

<applicationInstance type="elk" name="elk" IPv="v4" listenInterfaceName="mgmt">

19.7 Site Properties (topology:siteProperties)

The siteProperties element is one of the four core elements of the site topology file. This element contains site attributes and elements that define general site properties.

Table 50 lists the attributes and elements that are defined as part of the siteProperties element.

Table 50: Elements Defined in siteProperties

Element (Mandatory/Optional)	Description	Guidelines
name (Mandatory)	The name of the EMS/SDC site.	Each site should have a unique name.
sdcVersion (Mandatory)	The name of the SDC software.	The default value is “latest”.
timeZone (Mandatory)	The different time zone options by geographic areas that the system is configured to work in.	The relevant time zone geographic area.
ntpServers (Mandatory)	The server(s) used to synchronize time zones between servers.	Each ntpServer attribute within this element must contain the following two attributes: 27. name – the name of the NTP server 28. ip – the server IP address
traffixFolder (Optional)	The folder that all site configuration files are saved to.	The default is /opt/traffix
isManager (Mandatory)	Indicates if site is an EMS site or SDC site.	The default value is **False**, indicating the site is an SDC site. If the site is an EMS site, set the value to **True**.
isMultiSiteEnv (Mandatory)	Indicates if the site is an EMS site, an SDC site managed by an EMS site, or a standalone SDC site.	The default value is **False**, indicating the site is a standalone SDC site.

Element (Mandatory/Optional)	Description	Guidelines
		If the site is an EMS site, or an SDC site managed by an EMS site, set the value to **True**.
emsServers (Optional)	The elk Master instance(s) on the EMS site server(s).	Each emsServer attribute within this element must contain the following two attributes: ▪ name – the name of the EMS server ▪ ip – the IP address of the elk Master on the EMS server Note: If the isMultiSiteEnv element is defined with a value of **False**, make sure to leave the emsServers element empty.
nameservers (Optional)	Sets the DNS IP address. For example: <nameserver index="1" ip="192.168.16.5"/>	Can add 1-3 nameserver IP addresses.

Appendix B: Example of Scaling-out Site Topology File

The following is an example of a site topology file that indicates the added CPF component:

```
<?xml version="1.0" encoding="UTF-8"?>
<topology>
    <vms>
        <vm name="cpf-134" defaultGateway="10.240.9.1">
            <interfaces>
              <interface network="ic11" ip4="10.1.82.144" ip6="" dev="eth1"/>
            </interfaces>
            <applicationInstances>
               <applicationInstance type="cpf" name="cpf1"/>
            </applicationInstances>
        </vm>
        <vm name="cpf-135" defaultGateway="10.240.9.1">
            <interfaces>
              <interface network="ic11" ip4="10.1.82.145" ip6="" dev="eth1"/>
            </interfaces>
            <applicationInstances>
               <applicationInstance type="cpf" name="cpf1"/>
            </applicationInstances>
```

```
        </vm>

                                                    </vms>

</topology>
```

Appendix C: Port Settings Used by the SDC

During an installation or upgrade, a set of ports was enabled to ensure communication both between the different SDC components within the deployment, and between the SDC components and the necessary network elements.

This section describes the ports that have been validated for use by the SDC.

C.1 EMS Site Internal Ports

Table 51: EMS Internal Ports

Transport Protocol	Port	Network	Description
TCP	2812	IC	Monit
TCP	9200/9201 9300/9301	IC	ElasticSearch Discovery
TCP	5601	MGMT	Kibana Web Access
TCP	13868	IC	Traffic load balancing between the FEP and CPF instances
TCP	61616	IC	Communication between the configuration manager and the SDC components
TCP	61657	IC	Web UI Communication on cluster
UDP	161	IC	SNMP GET functions provided by OS snmpd service
UDP	162	IC	SNMP traps listener
UDP	1162	IC	OS trap daemon listener

Transport Protocol	Port	Network	Description
TCP	7000	MGMT	Cassandra Database inter-site communication
TCP	7001	MGMT	Cassandra Database inter-site communication
TCP	7199	MGMT	Cassandra JMX monitoring inter-site communication
TCP	9042	MGMT	Cassandra client

C.2 EMS Site External Ports

Table 52: EMS External Ports

Transport Protocol	Port	In/Out	Network	Description
TCP	22/443	In	MGMT	SSH remote consoles
TCP	80	In	MGMT	HP Blade System web consoles
UDP	123	Out	OAM	NTP Process
UDP	514	Out	OAM	Syslog Process
UDP	1161	Out	MGMT	For External EMS Statistics Analysis
UDP	User-defined Ports (and IPs)	Out	MGMT	Trap Forwarding: For External EMS Trap listeners

Transport Protocol	Port	In/Out	Network	Description
TCP	9300 & 9320	In/Out	MGMT	Elastic search sync between two EMS nodes
TCP	5601	In/Out	MGMT	Kibana Web Access
TCP &UDP	10046	In	MGMT	Fluentd Fwd EMS (Receive TDR & Traces from site to EMS; UDP for Hearbeats)
TCP	3868	In/Out	H-TCP	Inter-site communication link for geo-redundancy
SCTP	3868	In/Out	H-SCTP-A	Primary SCTP path for domestic traffic
SCTP	3868	In/Out	H-SCTP-B	Secondary SCTP path for domestic traffic
TCP	4505/6	In/Out	MGMT	Salt Master
TCP	8000	In/Out	MGMT	Salt API
TCP	8080/ 8443	In	MGMT	SDC web console (Web UI)
TCP	10040	Out	MGMT	NMS Agent to NMS Manager

Transport Protocol	Port	In/Out	Network	Description
				for system status synchronization
TCP	61617	In	MGMT	Communication between the EMS and the SDC servers for new configuration propagation
TCP	7000	In/Out	MGMT	Cassandra Database inter-site communication
TCP	7001	In/Out	MGMT	Cassandra Database inter-site communication
TCP	7199	In/Out	MGMT	Cassandra JMX monitoring inter-site communication
TCP	9042	In/Out	MGMT	Cassandra client

C.3 SDC Site Internal Ports

Table 53: SDC Internal Ports

Transport Protocol	Port	Network	Description
TCP	2812	IC	Monit

Transport Protocol	Port	Network	Description
TCP	61616	IC	Communication between the configuration manager and the SDC components
TCP	13868	IC	Traffic load balancing between the FEP and the CPF instances
TCP	11812	IC	RADIUS listening port between the FEP and the CPF
TCP	18080	IC	HTTP listening port between the FEP and the CPF
TCP	13386	IC	GTP listening port between the FEP and the CPF
TCP	1389	IC	LDAP listening port between the FEP and the CPF
TCP	4444	IC	NMS to CPF communication port
TCP	23210	IC	Tripo - CPF connection to Tripo
TCP	43211	IC	Tripo – inter-site connection
TCP	23212	IC	Tripo - connection between Tripo mates within the same site
TCP & UDP	10046	MGMT	Fluentd Fwd Site (FWD TDR & Traces from site to EMS UDP for Hearbeats)
TCP	61627	IC	Default configuration REST communication

Transport Protocol	Port	Network	Description
TCP	61637	IC	Default configuration REST communication
TCP	61647	IC	Default configuration REST communication NMS Agent
TCP	61657	IC	Default configuration rest communication - UI
UDP	4545	IC	Port prefix is 4545 and the postfix is the UID of the CPF or FEP (4545 + UID)
TCP	5555	MGMT	Tripo Web statistics
TCP	7000	MGMT	Cassandra Database inter-site communication
TCP	7001	MGMT	Cassandra Database inter-site communication
TCP	7199	MGMT	Cassandra JMX monitoring inter-site communication
TCP	9042	MGMT	Cassandra client

C.4 SDC Site External Ports

Table 54: SDC External Ports

Transport Protocol	Port	In/Out	Network	Description
TCP	4505/6	In/Out	MGMT	Salt Master
TCP	8000	In/Out	MGMT	Salt API
TCP	8080/8443	In	MGMT	SDC web console (Web UI)

Transport Protocol	Port	In/Out	Network	Description
TCP	80	In	MGMT	HP Blade System web consoles
UDP	162	Out	MGMT	SNMP traps toward the EMS or third-party NMS servers
TCP	3868	In/Out	H-TCP	Inter-site communication link for geo-redundancy
SCTP	3868	In/Out	H-SCTP-A	Primary SCTP path for domestic traffic
SCTP	3868	In/Out	H-SCTP-B	Secondary SCTP path for domestic traffic
TCP	61617	In	MGMT	Communication between the EMS and the SDC servers for new configuration propagation (internal and external data)
TCP	22/80/443/623/17990/ 17988	In	MGMT	HP iLO4 management consoles and virtual media
TCP	10030	Out	OAM	NMS Agent
UDP	123	Out	OAM	NTP Process
UDP	514	Out	OAM	Syslog Process
TCP	7000	In/Out	MGMT	Cassandra Database inter-site communication

Transport Protocol	Port	In/Out	Network	Description
TCP	7001	In/Out	MGMT	Cassandra Database inter-site communication
TCP	7199	In/Out	MGMT	Cassandra JMX monitoring inter-site communication
TCP	9042	In/Out	MGMT	Cassandra client

C.5 HP Integrated Lights-Out (iLO) Port Settings

The following information is not specific to SDC, but relates to relevant ports configured on different servers.

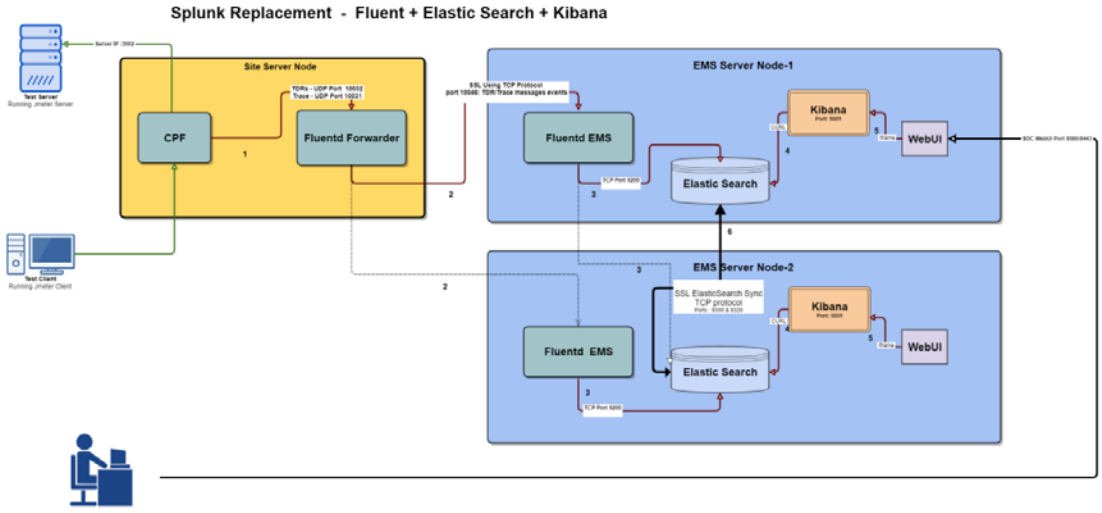
Table 55: HP iLO Ports

Transport Protocol	Port	iLO Function
CP	22	Secure Shell (SSH)
TCP	80	Web Server Non-SSL
TCP	443	Web Server SSL
TCP	3389	Terminal Services
TCP	17988	Virtual Media
TCP	9300	Shared Remote Console
TCP	17990	Console Replay

Appendix D: ELK Components

As of CF 30, for EMS deployments, Splunk is replaced with ELK. There are three ELK components on the EMS (Fluentd, Elasticsearch and Kibana) and one component on the SDC (Fluentd Forwarder). These components receive and forward information to create an overview of the deployment's performance and support shared configuration across multiple sites.

The following diagram shows the full flow of how information is forwarded and collected between an SDC site and EMS sites.



All of the ELK components are managed by monit and their status (up/down) is easily viewed, as all other components, on the WebUI SDC components page.

Glossary

The following tables list the common terms and abbreviations used in this document.

Table 56: Common Terms

Term	Definition
Answer	A message sent from one Client/Server Peer to the other following a request message
Client Peer	A physical or virtual addressable entity which consumes AAA services
Data Dictionary	Defines the format of a protocol's message and its validation parameters: structure, number of fields, data format, etc.
Destination Peer	The Client/Server peer to which the message is sent
Geo Redundancy	A mode of operation in which more than one geographical location is used in case one site fails
Master Session	The session for which the routing selection is performed based on the routing rules (Slave Sessions are applied with routing rules inherited from the Master Session)
Orchestrator	A workflow management solution to automate the creation, monitoring, and deployment of resources in your environment
Origin Peer	The peer from which the message is received
Pool	A group of Server Peers
QCOW2	A file format for disk image files
RADIUS	Remote Authentication Dial In User Service
REST	Representation of a resource between a client and server **(Re**presentational **S**tate **T**ransfer)
Request	A message sent from one Client/Server peer to the other, followed by an answer message

Term	Definition
RPM	RPM Package Manager
Salt-API	Manages and communicates between an Orchestrator and network master and minion servers
SDC Site	The entire list of entities working in a single site
Server Peer	A physical or virtual addressable entity which provides AAA services
Session	An interactive information interchange between entities
Slave (Bound) Session	A session which inherits properties from a master session
Transaction	A request message followed by an answer message
Tripo	Session data repository
vCenter	Vmware Virtual Infrastructure tool for centralized management of multiple hypervisors and enabling functionalities
Virtual Server	A binding point used by SDC to communicate with the Remote Peers (Clients and Servers)

Table 57: Abbreviations

Term	Definition
AAA	Authentication, Authorization and Accounting
ACL	Access Control List
AF	Application Function
API	Application Programming Interface
AVP	Attribute Value Pair
CLI	Command Line Interface
CPF	Control Plane Function
DEA	Diameter Edge Agent

Term	Definition
DRA	Diameter Routing Agent
EMS Site	Element Management System Site
FEP-In	In-Front End Proxy
FEP-Out	Out-Front End Proxy
HA	High Availability
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IMS	IP Multimedia Subsystem
JMS	Java Message Service
KPI	Key Performance Indicator
LDAP	Lightweight Directory Access Protocol
LTE	Long Term Evolution
MME	Mobility Management Entity
NGN	Next Generation Networking
Node	Physical or virtual addressable entity
OAM	Operation, Administration and Maintenance
OCS	Online Charging System
OVF	Open Virtualization Format
PCEF	Policy and Charging Enforcement Function
PCRF	Policy and Charging Rules Function
PLMN	Public Land Mobile Network
SCCP	Signaling Connection Control Part

Term	Definition
SCTP	Stream Control Transmission Protocol
SDC	Signaling Delivery Controller
SNMP	Simple Network Management Protocol
SS7	Signaling System No. 7
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UE	User Equipment
URI	Universal Resource Identification.
VIP	Virtual IP
VM	Virtual Machine
VNFC	Virtualized Network Function Component
VPLMN	Visited Public Land Mobile Network
Web UI	Web User Interface
WS	Web Service