Introduction to VPC-DIVPC-SI

This chapter introduces Cisco Virtualized Packet Core—Distributed Instance (VPC-DI). VPC-DI supports scalability for virtualized cloud architectures by extending the boundaries beyond a single virtual machine (VM).

Product Description

This chapter describes the StarOS VPC-DIVPC-SI architecture and interaction with external devices.

VPC-DI distributes the virtualized StarOS beyond a single Virtual Machine (VM), enabling multiple VMs to act as a single StarOS instance with shared interfaces, shared service addresses, load balancing, redundancy, and a single point of management.

The system operates as a fully distributed network of multiple VMs grouped to form a StarOS instance, with VMs performing management and session processing with a standby VM for each type.

VPC-SI consolidates the operations of physical Cisco ASR 5500 chassis running StarOS into a single Virtual Machine (VM) able to run on commercial off-the-shelf (COTS) servers. Each VPC-SI VM operates as an independent StarOS instance, incorporating the management and session processing capabilities of a physical chassis.

For UGP with Control and User Plane Separation (CUPS) VNF deployments, the CF and associated SF VNFCs comprise the Control Plane Function (CPF). The SAEGW service is used to provide Control Plane functionality for Sxa, Sxb, and Sxab (combined Sxa and Sxb) interfaces.

For CUPS-based VNF deployments, the UPF functions as an SAEGW-U. It is deployed as a VNFC running a single, stand-alone instance of the StarOS. Multiple UPF VNFCs can be deployed for scalability based on your deployment requirements.

For Day-N configuration procedures, refer to the *Ultra Packet Core CUPS Control Plane Administration Guide* and *Ultra Packet Core CUPS User Plane Administration Guide*.

Underlying Infrastructure for the System

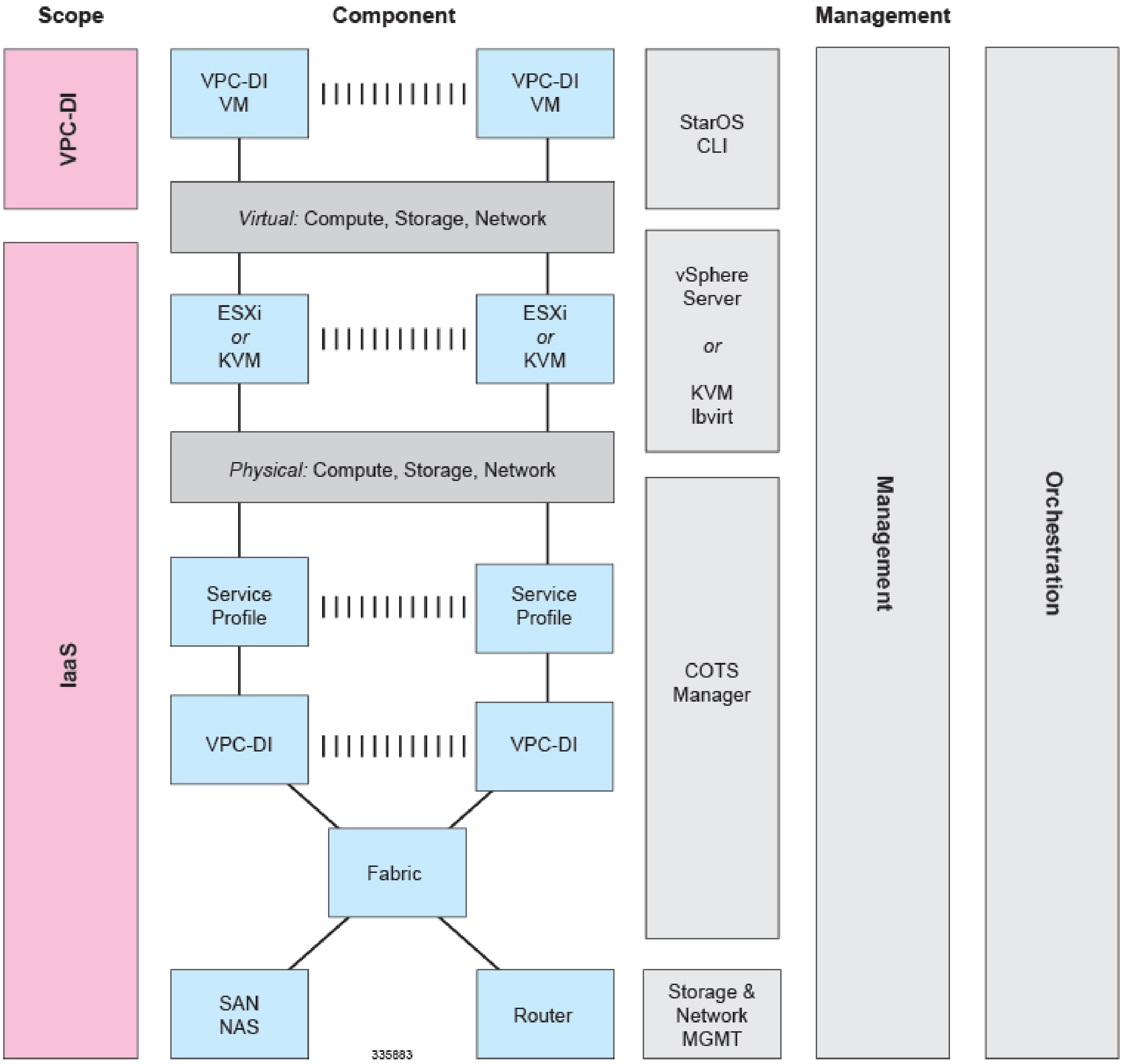
This virtualized system can be deployed into a new or existing Infrastructure as a Service (IaaS) cloud datacenter. VPC-DIVPC-SI runs in a set of virtual machines (VMs) using industry standard hypervisors on Commercial Off The Shelf (COTS) servers. This deployment model allows the management of physical infrastructure to remain outside of the scope of StarOS and VPC-DIVPC-SI.

A typical instance runs on the following NFVi (network function virtual infrastructure):

- IaaS components
  - COTS blade chassis, blades, and fabric interconnects
  - Manager Software
  - Network Attached Storage (NAS) or Storage Area Network (SAN)
  - VMware ESXi or KVM hypervisor on each blade or server
  - VMware vSphere Server or KVM OpenStack
- StarOS software installed within VMs
- Each VM must run on a separate blade so if a blade fails, only a single VM is affected (anti-affinity).
- Existing Management software (service, logging, statistics, etc.)
- Orchestration software (optional) [See Orchestration, on page 19]

A VPC-DIVPC-SI instance is a grouping of VMs that act as a single manageable instance of StarOS. VPC-DIVPC-SI consists of the following major components:

Figure 1: VPC-DI Scope



- Control Function (CF) VMs, on page 3
- Service Function (SF) VMs, on page 4
- DI Network, on page 5

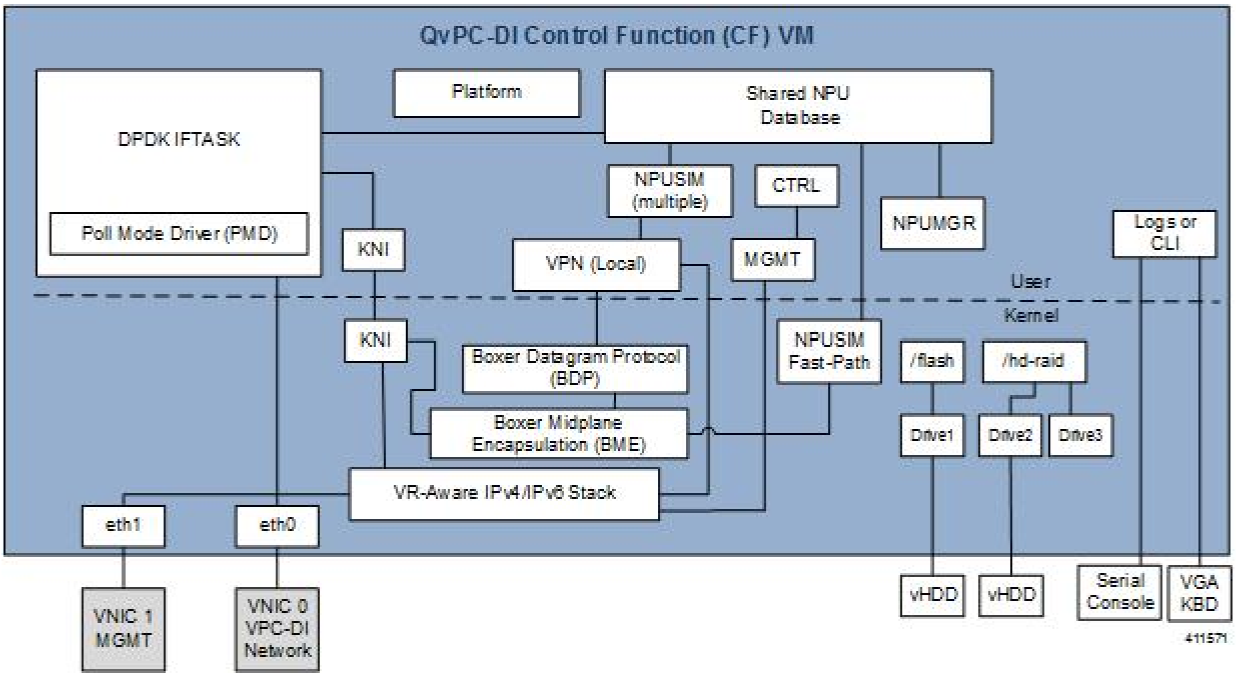
Control Function (CF) VMs

Two CF VMs act as an active:standby (1:1) redundant pair. The active CF is responsible for the following functions:

- Controller tasks

- Local context VPNMGR
- Local context (MGMT) and DI-Network vNICs
- System boot image and configuration storage on vHDD
- Record storage on vHDD
- Out-of-Band (OOB) management (vSerial and vKVM) for CLI and logging

Figure 2: Control Function VM



Service Function (SF) VMs

SF VMs provide service context (user I/O ports) and handle protocol signaling and session processing tasks. VPC-DIVPC-SI

From StarOS 21.23.0 onwards, VPC-DI instance can have a minimum of 4 SF VMs. Note that the maximum number of SF VMs and number of the active SF VMs depend on the product type being deployed. For more information, contact your local sales or service representative.

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

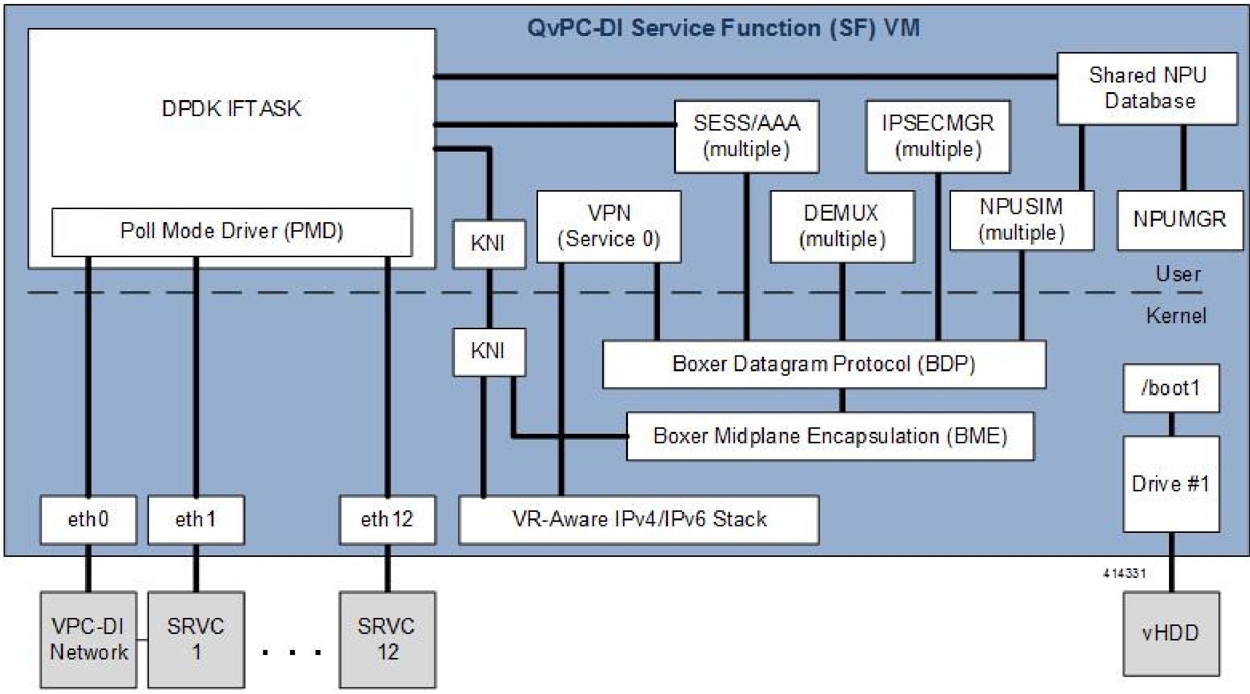
- Demux VM (flow assignments)
- Session VM (traffic handling)
- Standby VM (n+1 redundancy)

An SF is responsible for the following functions:

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

A minimum configuration for a VPC-DIVPC-SI instance requires four SFs two active, one demux and one standby.

Figure 3: Service Function VM



DI Network

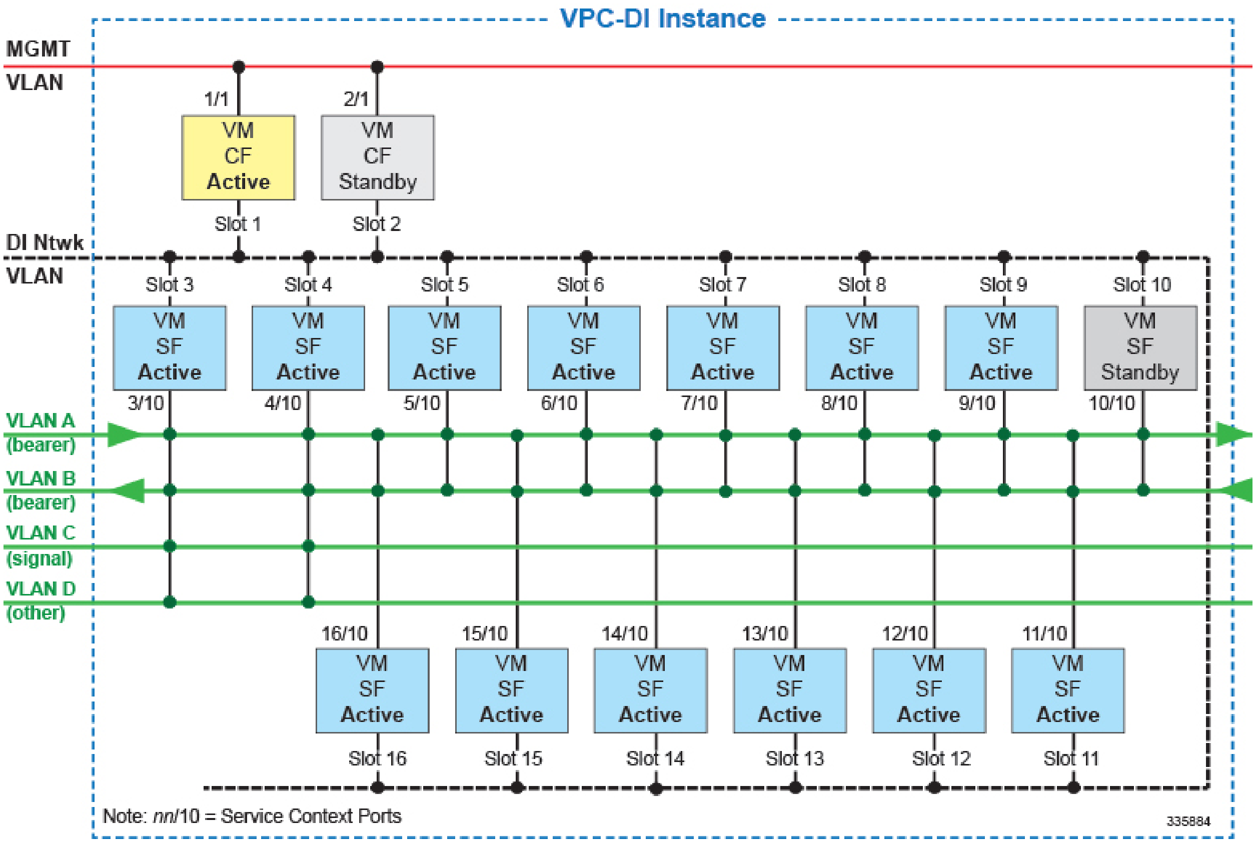
In order for the VMs within a VPC-DIVPC-SI instance to communicate with each other, each instance must have a private L2 network that interconnects the VMs. This network should utilize a VLAN within the IaaS/virtualization infrastructure and be exposed untagged to each VM as the first vNIC.

The DI network must be for the exclusive use of a single VPC-DIVPC-SI instance. No other devices may be connected to this network. It has to be a secure network that limits access.

Note If more than one instance is instantiated within the same datacenter, each instance must have its own DI network.

All the VMs within an instance must be physically located in the same site, ideally in the same few racks with minimal interconnecting devices. The reliability of the DI network is important for the stability of the VPC-DIVPC-SI instance. Using L2 tunneling protocols across a WAN or congested links is highly discouraged.

Figure 4: DI Network



Note: *nn*/10 = Service Context Ports

Network Requirements

The reliability and performance of the DI network is critical to the reliability and performance of VPC-DIVPC-SI. The DI Network is used for internal control, signaling, and bearer traffic. Bearer traffic may traverse the DI network multiple times, so any packet loss in the DI network would impact the perceivable packet loss of VPC-DIVPC-SI as a whole.

For OpenStack deployments, both the DI-Network and Service network types should be configured as SR-IOV mode flat. For deployments on VMware, the DI-Network type should be configured as VMXNET3 or PCI-PT with bonding and VLAN tagged. On VMware, the Service network type should be configured as VMXNET3 or PCI-PT.

Note The infrastructure connecting the VMs should be 10 Gbps or higher between all VMs and have a redundant configuration. A redundant configuration can be provided in one of these ways:

- on the host using a vSwitch (for Virtio/VMXNET3 interfaces)
- on the hardware, such as the Cisco UCS virtual interface card (VIC)
- in the VPC-DIVPC-SI using network interface bonding

The IaaS/hypervisor must provide a DI network that can:

- Perform as a L2 Ethernet bridge/switch.
- Support independent, per-VLAN address learning.
- Allow promiscuous mode vNICs. (This functionality is disabled by default in VMware ESXi.)
- Support jumbo frames up to at least 7100 bytes. If your installation does not support jumbo frames, you can still use the VPC-DIVPC-SI. You must set the appropriate parameter in the boot parameters file as described in Configure Support for Traffic Above Supported MTU.

The infrastructure/hypervisor should provide a DI network that can:

- Support 802.1p priorities sent from the VMs with VID=0.
- Honor 802.1p priorities end-to-end between all VMs within the instance.
- Provide redundant L2 paths in all physical infrastructure or 802.1p priorities end-to-end between all system VMs within the instance.
- Provide a secure network that limits access. It has to be a secure network in a sense, that it is only locally available.

Specifically, the DI network should have the following minimum reliability requirements:

- Fully redundant L2 paths
- No outage longer than 1.5 seconds, including STP and LACP outages (if applicable)
- Packet prioritization
- Sufficient network bandwidth to minimize any control or bearer packet loss

Disruptions in the DI network or excessive packet loss may cause false failure detection or unpredictable behavior in the VPC-DIVPC-SI instance.

Each system VM monitors the reachability of the other VMs and the reliability of the DI network on an ongoing basis.

Jumbo Frames

We recommend that the DI network support jumbo frames up to at least 7100 bytes. On startup, each VM issues a series of ping commands on the network to determine that jumbo frames are supported. Support of jumbo frames provides better system performance.

If your installation does not support jumbo frames, you can still use the VPC-DIVPC-SI. You must set the appropriate parameter in the boot parameters file as described in Configure Support for Traffic Above Supported MTU.

The CF and SF do not start if an MTU of less than 7100 is detected and the appropriate boot parameter is not set.

Service ports on SFs can also support a maximum MTU up to 9100 bytes in Release 21.4 and higher, or 2048 bytes in older releases if configured appropriately in the StarOS configuration.

Record Storage

Record storage is available on instance-wide storage devices available at **/records**. Both CF VMs are provisioned with a second vHDD (/hd-raid) of suitable size for record storage (minimum of 16GB). The CFs share a RAID configuration to mirror data between their vHDDs. The SFs send data records to the active CF over the DI network for transfer to external ephemeral storage that was created and mounted manually or orchestrated by the VNFM.

Packet Flows

SF ports are used to receive and transmit bearer and signaling packets. To simplify network settings and address usage, only VLANs for high-bandwidth (bearer) packets need to be connected to all SFs. Low-bandwidth interfaces (signaling) can be connected to just two SFs. In the diagrams below, the bearer VLANs are connected to all SFs, while signaling and other VLANs are only connected to the first two SFs.

Note This asymmetric arrangement means that fewer interfaces are needed, however careful consideration should be paid to failures since the loss of two VMs results in loss of services.

ECMP does load-sharing based on a hash and can send traffic to any SF VM.

On ingress, the SFs perform flow lookups and direct packets to the specific SESSMGR task on a specific SF. Some of this ingress traffic is processed by local SESSMGR tasks, otherwise it is relayed via the DI network to the correct SF. On egress, each SF sends out packets from its local port (provided ECMP is used). In most cases, the number of VMs that packets traverse is less than two. However, ACLs and tunneling may increase the number of hops for specific flows depending on the EPC configuration.

Packets Received on SF Demux VM

On the demux and standby SF, all session traffic received is relayed to another SF for session processing. The following table describes the processing of Packets received on the SF Demux VM.

Item	Description
1	Receive NPU does flow lookup to determine SESSMGR, (Ingress)
2	SESSMGR processes packet.
3	Transmit NPU sends packet out local port. (Egress)

Packets Received on SF Session VM

The following table describes the processing of Packets received a on the SF Session VM.

Item	Description
1	Receive NPU does flow lookup to determine SESSMGR, (Ingress)
2	SESSMGR processes packet.
3	Transmit NPU sends packet out local port. (Egress)

Feature Set

Interfaces and Addressing

Each VM in a VPC-DIVPC-SI instance is represented as a virtual card with a single CPU subsystem. This makes many CLI commands, logs, and functions work similarly to StarOS running on ASR 5500 platform.

StarOS concepts of contexts, services, pools, interfaces, cards, and ports exist on each VM just as on ASR 5500 platform.

When the VM boots, the vNICs configured in the VM profile are detected and an equivalent number of "Virtual Ethernet" type ports appear in the StarOS CLI.

Refer to Creating a Boot Parameters File to manually specify the order of the vNICs.

By default, the system assigns the vNIC interfaces in the order offered by the hypervisor.

- CF VMs (slots 1 and 2)
  - First interface offered (1/0 or 2/0) is for the DI Network.
  - Second interface offered (1/1 or 2/1) is for the management network.
- SF VMs (Slots 3 through 16)
  - First interface offered (*slot*/0) is for the DI Network.
  - Traffic Interfaces *slot*/10 through *slot*/21 are for IaaS VLAN control and data traffic.

Note StarOS supports up to 12 service ports, but the actual number of ports may be limited by the hypervisor.

It is critical to confirm that the interfaces listed in the supported hypervisors line up with the KVM bridge group or VMware vSwitch in the order in which you want them to match the VM interfaces.

Note You cannot be guaranteed that the order of the vNICs as listed in the hypervisor CLI/GUI is the same as how the hypervisor offers them to the VM. On initial setup you must use the **show hardware** CLI command to walk through the MAC addresses shown on the hypervisor vNIC configuration and match them up with the MAC addresses learned by the VMs. This confirms that the VM interfaces are connected to the intended bridge group or VMware vSwitch.

Encryption

VMs within a VPC-DIVPC-SI instance perform software-based encryption and tunneling of packets (as opposed to the higher-throughput hardware-based services). Call models that make heavy use of encryption for bearer packets or have significant PKI (Public Key Infrastructure) key generation rates may require significant compute resources.

If your COTS server hardware uses the Coleto Creek chipset based on the Intel 89xx chip, the system automatically utilizes this hardware chip for encryption and decryption of packets. However, all service function VMs must use this chipset in order for the system to use the hardware chipset for encryption and decryption.

Security

Security of external traffic including tunneling, encryption, Access Control Lists (ACLs), context separation, and user authentication function as on existing StarOS platforms. User ports and interfaces on the CFs and SFs are protected through StarOS CLI configuration.

The virtual system adds additional security concerns on the customer because network communication travel over the DI network on datacenter equipment.

The DI network must be isolated from other hosts within the datacenter by limiting membership in the system network's VLAN to VMs within that specific VPC-DIVPC-SI instance. Unauthorized access to the DI network through other hosts being inadvertently added to that network or the compromise of a router, switch or hypervisor could disrupt or circumvent the security measures of StarOS. Such disruptions can result in failures, loss of service, and/or exposure of control and bearer packets. Properly securing access to the DI network is beyond the control of StarOS.

Communication between DI network component (e.g. CF and SF) VMs is now only possibley via authentication over externally supplied SSH keys. In addition, the system enforces public/private key-based SSH authentication for logins within the DI network. No passwords, keys or LI information are stored or sent in clear text.

If an operator requires physical separation of networks, such as management versus bearer versus LI (Lawful Intercept), then physical separation of the DI network should also be done since it carries sensitive data. In a virtualized environment, the physical separation of networks may not be possible or practical. Operators that have these requirements may need to qualify their hypervisor and infrastructure to confirm that it will provide sufficient protection for their needs.

Licensing

For this release, VPC-DIVPC-SI employs chassis-independent (no chassis serial number) licenses of fixed duration.

By default the expiry date is 12 months from the software build date, regardless of the license expiry date. If the license expiry date is earlier than the 12-month period, then the license expiry date will be used.

For additional information, contact your Cisco account representative.

Redundancy and Availability

Platform Requirements

The virtual system relies on the underlying hardware and hypervisor for overall resource redundancy and availability. The StarOS handles the system redundancy on top of that.

The hardware and hypervisor should provide:

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

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

Interchassis Session Recovery (ICSR) is also recommended to improve availability and recovery time in the case of a non-redundant hardware failure (such as CPU, memory, motherboard, hypervisor software). ICSR provides redundancy at the session level for gateways only.

CF Redundancy

The two CF VMs are 1:1 redundant for control of the VPC-DIVPC-SI instance and the local context/management port.

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

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

SF Redundancy

Each SF VM provides network connectivity for service ports. Each SF provides one or more ports and associated interfaces, but the SFs do not provide 1:1 redundancy as they are not paired together.

Redundancy of SF ports should be achieved using ECMP or another supported L3 protocol.

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

It is recommended to use BFD for detection of path failures between an SF and the peer router so ECMP paths are excluded in the event of a failure.

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

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

IfTask Redundancy

If Active Standby task gets killed or stopped, the transition to Standby IFTASK takes place. This transition process involves initialization of certain parameters on the Standby IFTASK to Active IFTASK. At the current heartbeat interval, the initialization might take more than three seconds and leads to card reboot.

To avoid card reboot, set the HAT HB interval to high value using the following configuration:

```
configure
   [ default ] high-availability fault-detection card hb-loss value
```

ICSR Support

VPC-DIVPC-SI supports ICSR between two instances for services that support ICSR in the StarOS software release. When more than one service type is in use, only those services that support ICSR will be able to use ICSR.

Note The two VPC-DI instances must be on a different DI-Network.

ICSR supports redundancy for site/row/rack/host outages, and major software faults. To do so, the two instances should be run on non-overlapping hosts and network interconnects. ICSR is supported only between like-configured instances. ICSR between a VPC-DIVPC-SI instance and another type of platform is not supported.

L3 ICSR is supported.

For additional information, refer to the *Interchassis Session Recovery* chapter in this guide.

Hypervisor Requirements

VPC-DIVPC-SI has been qualified to run under the following hypervisors:

- OpenStack based virtualized environments
- VMware ESXi
  - Version 6.0: Supported in releases prior to Release 21.8
  - Version 6.5: Supported in Release 21.8 and 21.9
  - Version 6.7: Supported from Release 21.10 onwards

Heat templates (for OpenStack) and OVF/OVA templates (for VMware vSphere) are provided for CF and SF VMs.

A VMware vApp bundles an entire VPC-DI instances's VMs together for easy deployment.

It is recommended to use Cisco Elastic Services Controller (ESC) to deploy VPC-DI with OpenStack. Refer toOnboarding the VPC-DIVPC-SI with ESC on OpenStack .

Note Deviations from the supplied templates must be approved by Cisco engineering in order to maintain performance and reliability expectations.

CF VM Configuration

The system requires that each CF VM be configured with:

- 8 vCPUs
- 16 GB RAM
- First vNIC is the DI Network.
- Second vNIC is the management port.
- First vHDD is for boot image and config storage (/flash, non-RAID, 4GB recommended).
- Second vHDD is for record storage [optional] (hd-local1, RAID, 16GB minimum recommended).

Note Both CF VMs must be identically configured.

SF VM Configuration

The system requires that each SF VM be configured with:

- 12 or more vCPUs (see vCPU and vRAM Options, on page 14).

  Note An SF VM will not boot and will report the following error if this minimum vCPU requirements is not met.

  ```
  Found hardware disparate in minimum number of cores, found n cores,
   minimum expected cores are 12.
  ```

- 32 GB or more vRAM (see vCPU and vRAM Options, on page 14).
- First vNIC is the DI Network.
- Second and subsequent vNICs are service ports. The system supports up to 12 vNICs, but this number may be limited by the hypervisor.
- vHDD is for boot image, 2 GB recommended.

Note All SF VMs must be identically configured. Refer to VM Hardware Verification for information on monitoring the VM hardware configuration.

vCPU and vRAM Options

A CPU is a single physical computer chip that can have more than one physical CPU core that is fully capable of running the entire system and applications by itself. Virtual core technology supports multiple logical processors (vCPUs) per physical core. The total number of vCPUs supported on a specific platform varies based on the number of available physical cores and the type of virtual core technology implemented in each core.

It is best practice to avoid spanning VM across NUMA nodes. The vCPU shall align with the underlying CPU. A VM should not span across NUMA node for performance optimization. Limitations for the number of numa nodes is 2.

CF and SF run within VMs that are assigned a number of vCPUs, each supporting one thread (sequence of instructions). The number of available vCPUs supported by the platform CPU may exceed the maximum number of vCPUs that can be assigned to the VM via the hypervisor.

Note The number vCPUs per VM should never exceed the maximum number of vCPUs supported by the platform CPU.

To maximize performance, it may be desirable to adjust the number of vCPUs or vRAM to align with the underlying hardware. SF supports varied vCPU and vRAM combinations, however all SFs must share the same combination within an instance.

Software will determine the optimal number of SESSMGR tasks per SF on startup of the SF based on the number of vCPUs and amount of vRAM on that SF.

Note Dynamic resizing of vCPU count, vRAM size or vNIC type/count (via hotplug, ballooning, etc.) is not supported. If these values need to be changed after provisioning, all VMs must be shut down and reconfigured. Reconfiguration can be performed only on all VMs at once. VMs cannot be reconfigured one at a time since the CPUs and RAM would not match the other instances.

vNIC Options

In this release the supported vNIC options include:

- VMXNET3—Paravirtual NIC for VMware
- VIRTIO—Paravirtual NIC for KVM
- ixgbe—Intel 10 Gigabit NIC virtual function
- enic—Cisco UCS NIC

- SR-IOV—Single root I/O virtualization ixgbe and enic interfaces

Support for vhost-net and vhost-user

The system implements a Virtio front-end based on a DPDK-based user application which can interact with both vhost-net and vhost-user based back-end mechanisms. Vhost-user and vhost-net provide an implementation of the vhost paradigm of using shared memory based, event, and interrupt descriptors. The DPDK-based front-end driver in combination with vhost-net and vhost-user provide a higher performance data path compared to a Linux bridge-based data path.

- Vhost-user provides packet handling completely in the user space, which improves performance. The system implements the front-end in a DPDK-based user space application, while a host user space application implements the back-end based on the vhost-user interface.
- Vhost-net provides the kernel-level back-end for Virtio networking that reduces virtualization overhead by moving Virtio packet processing tasks out of the user space (the QEMU process) and into the kernel (the vhost-net driver). This allows device emulation code to directly call into kernel subsystems, instead of performing system calls from the user space.

  The system supports single queue in vhost-user.

Hard Drive Storage

In additional to the mandatory /flash (non-RAID) drive, the system supports RAID1 under a virtual machine (VM). For each VM, Virtual SCSI disks can be created, on CF only, matching the SCSI ID shown in this table. The minimum disk size must be greater than 16 GB.

Table 1: Disk Mapping

Type	/flash (non-RAID)	hd-local1	Notes
KVM	SCSI 0:0:0:0	SCSI 0:0:1:0	Raw disk hd-local1 uses RAID1
VMware	SCSI 0:0:0:0	SCSI 0:0:1:0	Raw disk hd-local1 and hd-remote1 use RAID1

For record storage (CDRs and UDRs) the CF VM should be provisioned with a second vHDD sized to meet anticipated record requirements (minimum 16GB). Records will be written to /records on the second vHDD.

Cisco UCS C220 M6 Server

The Cisco UCS C220 M6 Rack Server is a 2-socket, 1-Rack-Unit (1RU) rack server, which facilitates various workloads such as virtualization, collaboration, and bare-metal applications. Cisco UCS C-Series M6 Rack Servers can be deployed as standalone servers. RHOSP-based VPC-DI deployments are supported on Cisco UCS C220 M6 servers that meet the configuration requirements specified in the following sections.

Hardware Configuration

Hardware	Configuration
Server	UCSC-C220-M6S
Processor	2 x Intel(R) Xeon(R) Gold 6338N CPU @ 2.20GHz
RAM	512 G
CIMC version	4.3(2.230207)
NIC – Type	2 x Intel E810CQDA2 2x100 GbE QSFP28
NIC – Driver version	1.12.6
NIC - Firmware	4.20 0x80018f67 0.387.18

Software Version Matrix

Software	Configuration for OSP 16.2 VIM	Configuration for OSP 17.1 VIM
OS / Hypervisor	RHEL 8.4 (KVM)	RHEL 9.2 (KVM)
VIM	OSP16.2	OSP 17.1
StarOS	21.28.mh14	21.28.mh21 or later versions
Deployment	• VPC-DI CP • ePDG	• VPC-DI CP • ePDG

VNF Flavor Details for M6 Server

Use the following VNF details for CUPS-CP deployment.

VM Type for CUPS-CP Deployment	vCPU	Memory(GB)	Root Disk (GB)	Data Disk (GB)	vCPU Core Distribution
CF	8	64	16	200	–
SF	60	304	6	–	24 (iftask) + 36 (sessmgr

Use the following VNF details for ePDG deployment.

VM Type for ePDG Deployment	vCPU	Memory(GB)	Root Disk (GB)	Data Disk (GB)	vCPU Core Distribution
CF	8	64	16	200	vCPU core distributions for ePDG deployments depends on the characteristics of the deployment. Contact your local sales or service representative for more information.
SF	60	304	6		

For more information about Tunable IFTASKS, see the Configuring IFTASK Tunable Parameters section in the *VPC-DI System Administration Guide*.

VIM Level Tuning

VIM Configuration	Description
HugePages	Enabled
HugePages Size	1G
CPU Reservation for VIM	2 pCPUs/ socket
Hyperthreading	Disabled
CPU Pinning	Enabled (Ex -: vcpu_pin_set=2-31,34-63)
ISOL-CPU Config	Enabled (Ex -: “isolcpus=2-31,34-63")
SR-IOV	FLAT
VF per PF	16
VF Mode	Trusted
DI-Net	Bonded Interface
Service-Net	Mapped to the VFs corresponding to PFs from both the NUMA nodes
VNFM-Net/ Mgmt-net	Virtio based networks mapped to MLOM ports

Host Level Tuning

BIOS Options	Preferred Value
SpeedStep (Pstates)	Disabled
Turbo Mode	Disabled
Hyper-Threading [ALL]	Disabled
CPU Performance	Enterprise
Intel VT for directed IO	Enabled
Package C State	C0 C1 State
Processor C1E	Disabled
Processor C6 Report	Disabled
EIST PSD Function	HW ALL
Energy/Performance Bias Config	Performance
Power Performance Tuning	BIOS
Hardware P-States	HWPM Native Mode
Workload Configuration	I/O Sensitive
NUMA	Enabled
Sub NUMA Clustering	Disabled
XPT Prefetch	Enabled
UPI Prefetch	Enabled
Patrol Scrub	Disabled
Fan Policy	Maximum Power

DPDK Internal Forwarder

The Intel Data Plane Development Kit (DPDK) is an integral part of the VPC architecture and is used to enhance system performance. The DPDK Internal Forwarder (IFTASK) is a software component that is responsible for packet input and output operations and provides a fast path for packet processing in the user space by bypassing the Linux kernel. It is required for system operation. Upon CF or SF instantiation, DPDK allocates a certain proportion of the CPU cores to IFTASK depending on the total number of CPU cores. The remaining CPU cores are allocated to applications.

Refer to Configuring IFTASK Tunable Parameters for more information.

To determine which CPU cores are used by IFTASK and view their utilization, use the **show npu utilization table** command as shown here:

```
[local]mySystem# show npu utilization table

Wednesday July 06 10:53:55 PDT 2017
          -------iftask-------
```

```
   lcore     now   5min  15min
--------  ------ ------ ------
  01/0/1     38%    53%    52%
  01/0/2     51%    55%    55%
  02/0/1     66%    72%    68%
  02/0/2     66%    63%    67%
  03/0/1     57%    55%    55%
  03/0/2     51%    47%    45%
  03/0/3     89%    89%    89%
  03/0/4     88%    88%    89%
  04/0/1     67%    59%    58%
  04/0/2     54%    40%    48%
  04/0/3     89%    89%    90%
  04/0/4     90%    89%    89%
  05/0/1     55%    55%    56%
  05/0/2     68%    45%    45%
  05/0/3     90%    89%    89%
  05/0/4     90%    89%    89%
  06/0/1     50%    58%    58%
  06/0/2     24%    24%    25%
  06/0/3     89%    90%    90%
  06/0/4     91%    90%    90%
```

To view CPU utilization for the VM without the IFTASK cores, use the **show cpu info** command. For more detailed information use the **verbose** keyword.

```
[local]mySystem# show cpu info card 6
Tuesday July 05 10:39:52 PDT 2017
Card 6, CPU 0:
  Status                : Active, Kernel Running, Tasks Running
  Load Average          : 7.74, 7.62, 7.54 (9.44 max)
  Total Memory          : 49152M
  Kernel Uptime         : 4D 5H 7M
  Last Reading:
    CPU Usage           : 25.4% user, 7.8% sys, 0.0% io, 0.1% irq, 66.7% idle
    Poll CPUs           : 4 (1, 2, 3, 4)
    Processes / Tasks   : 177 processes / 35 tasks
    Network            : 164.717 kpps rx, 1025.315 mbps rx, 164.541 kpps tx, 1002.149 mbps
 tx
    File Usage          : 8256 open files, 4941592 available
    Memory Usage        : 21116M 43.0% used
  Maximum/Minimum:
    CPU Usage           : 32.9% user, 8.9% sys, 0.0% io, 0.4% irq, 59.1% idle
    Poll CPUs           : 4 (1, 2, 3, 4)
    Processes / Tasks   : 184 processes / 36 tasks
    Network            : 178.388 kpps rx, 1270.977 mbps rx, 178.736 kpps tx, 1168.999 mbps
 tx
    File Usage          : 8576 open files, 4941272 available
    Memory Usage        : 21190M 43.1% used
```

Orchestration

When a VPC-DIVPC-SI instance is deployed, there are several expectations of the environment on which VPC-DIVPC-SI is running that are beyond the control of StarOS. Most of these fall into requirement of the Orchestration System.

- Provisioning of VPC-DIVPC-SI VMs including install and parameters assignment: configuration, connectivity, and persistent block storage for each VM.

- L2 provisioning of the DI network to ensure that the DI network meets reliability requirements.
- Policy enforcement of network separation, if applicable.
- Physical placement of VMs that enforce redundancy rules.
- Providing useful monitoring tools for physical resources, such as CPU, RAM, NIC, etc.

If an orchestration system is not used to deploy a VPC-DIVPC-SI instance, these requirements must still be maintained. However, they should be enforced manually or through other means. Refer to VM Hardware Verification for information on monitoring the VM hardware configuration.

Provisioning

Provisioning of a VPC-DIVPC-SI instance has two phases:

- VMs and network interconnections are created and linked.
- VPC-DIVPC-SI instance is configured for services.

IaaS administrators set up and interconnect the servers and use hypervisor VM templates or orchestration software to create a set of VMs, the DI network, and the redundancy configuration to meet Service Level Agreement (SLA) requirements.

Deploying a VPC-DIVPC-SI instance requires a detailed configuration plan that addresses the operator's deployment requirements.

Boot Sequence

StarOS is installed on each VM using pre-installed disk templates in QCOW2 format. Slot numbers are managed by ESC and OpenStack. A slot number is assigned as part of the VM configuration. The slot number is auto-detected during install. Installation of the installer image is completely automated provided that the slot number can be detected from the hypervisor. For additional information, see Software Installation and Network Deployment, on page 23.

For information regarding how to control the configuration of vNICs from the VM, refer to Creating a Boot Parameters File.

Each VM will reboot and attempt to join the VPC-DIVPC-SI Instance. A bootloader boots the instance via automated (scripted), network or manual booting.

Upon completion of the virtual BIOS, the VM boots from its local vHDD and runs CFE (Common Firmware Environment). CFE looks on the vHDD for the presence of the parameters file that was created during installation. If this file is found and parses correctly, CFE takes different paths depending on the VM's type and slot number. In all cases, the first vNIC becomes the interface for the network.

CF Boot Sequence

The CF performs the following functions during its boot sequence:

- Checks to see if the other CF is alive (via the DI network).
- If other CF is alive, attempts to boot from it.
  - Tries to obtain parameters and a boot image from the other CF.

    - If successful, transfers the boot image and runs it.

- If the other CF is not alive or booting from it fails, boots independently.
    - Finds and parses a boot.sys file on the local vHDD for boot/config priorities.
    - Performs a boot via instructions in the boot.sys unless interrupted by a user (via the Management network or local vHDD).

CFE on a CF supports downloading a starfile (bootable image) from the peer CF, via the CF management vNIC to an external HTTP or TFTP server, or from a local file on its vHDD. This is driven by the boot.sys and the StarOS **boot** CLI command.

Note HTTP and TFTP booting are only supported on VIRTIO and VMXNET3 interface types.

A network protocol on the DI network determines which CF is master. Mastership is then communicated over the DI network to SF VMs.

SF Boot Sequence

An SF boots from its vHDD. It then contacts the active CF via the DI network to determine if it booted the correct software version. If the SF did not boot the correct software version, it transfers the correct version from the CF and reboots itself. Once it boots the correct the software version, the boot sequence is complete.

Bandwidth Requirements

Modeling of bandwidth requirements on the L2 switches that host a VPC-DIVPC-SI instance is required for each operator deployment.

In addition to the predominant bearer traffic, the DI network also passes session signaling and internal control data between the VMs.

Internal control traffic will be heavy during redundancy operations, but significantly less under normal operation. Heavy use of control traffic occurs during:

- Migrations of tasks from an active SF VM to the standby SF
- Startup or restart of a standby SF
- Startup or restart of an SF
- Startup or restart of an SF or standby CF
- Heavy signaling traffic (high Call Events per Second [CEP] rate)
- Significant CLI and/or Bulkstats usage

Depending on the CEPS rate, configuration, and management operations, each VM places a load on its DI network interface regardless of bearer throughput. This load is expected to be highly variable, but average less than 1 Gbps per VM with some VMs having higher usage than others.

Contact the Cisco Technical Assistance Center (TAC) or Advanced Services (AS) for assistance when calculating bandwidth requirements.

Capacity, CEPS and Throughput

Sizing a VPC-DIVPC-SI instance requires modeling of the expected call model.

Many service types require more resources than others. Packet size, throughput per session, CEPS (Call Events per Second) rate, IPSec usage (site-to-site, subscriber, LI), contention with other VMs, and the underlying hardware type (CPU speed, number of vCPUs) will further limit the effective number of maximum subscribers. Qualification of a call model on equivalent hardware and hypervisor configuration is required.

Software-based transmit batching greatly enhances the system performance.

Diagnostics and Monitoring

Because VPC-DIVPC-SI runs within VMs, no hardware diagnostics or monitoring are provided. Retrieval of hardware sensor data (temperature, voltage, memory errors) are accomplished via the hypervisor and external monitoring systems. To determine the configuration of the underlying VMs, refer to VM Hardware Verification.

VPC-DIVPC-SI monitors and exports vCPU, vRAM, and vNIC usage per VM through existing mechanisms including CLI **show** commands, bulkstats and MIB traps. However, an operator may find that monitoring physical CPU, RAM, and NIC values per host in the hypervisor is more useful.

Because vNICs have a variable max throughput (not defined as 1 Gbps or 10 Gbps for example), counters and bulkstats that export utilization as a percentage of throughput may have little value. Absolute values (bps) can be obtained from the VM, but where possible physical infrastructure utilization should be obtained from the hypervisor. This would not apply to pass-through PF NICs, as those have a fixed maximum throughput.

Cisco Prime Analytics

The Cisco Prime for Mobility suite of analytics provides scalable management of a VPC-DIVPC-SI instance.

Cisco Prime for Mobility supports the following:

- Integrated operator workflows across the Radio Access Network (RAN) backhaul and packet core
- Centralized network visibility and advanced troubleshooting and diagnostics
- Pre-integrated network management software components that reduce time and resources required for integration

For additional information, contact your Cisco account representative.

StarOS VPC-DIVPC-SI Build Components

The following StarOS build filename types are associated with VPC-DIVPC-SI:

- **.qvpc-di-<version>.iso** initial installation or startover ISO file.
- **.qvpc-di-<version>.bin** update, upgrade or recovery file for a system that is already running.
- **.qvpc-di-template-libvirt-kvm-<version>.tgz** KVM libvirt template plus ssi_install.sh.

- **.qvpc-di.qcow2.tgz** KVM QCOW2 disk template.
- **.qvpc-di-template-vmware.tgz** VMware files.
- **.qvpc-di-template-vmware-<version>.ova** VMware OVA template.

Software Installation and Network Deployment

This guide assumes that components of VPC-DIVPC-SI have been properly installed to run in virtual machines (VMs) on commercial off-the shelf (COTS) servers. For additional information, see Provisioning, on page 20.

The DI network must also be provisioned within the datacenter to meet the requirements specified in DI Network, on page 5 and Bandwidth Requirements, on page 21.

This guide assumes that VPC-SI has been properly installed to run in a virtual machine (VM) on a commercial off-the shelf (COTS) server.

For additional information on supported operating system and hypervisor packages, as well as platform configurations, please contact your Cisco representative. The Cisco Advanced Services (AS) group offer consultation, installation and network deployment services for the VPC-DIVPC-SI product.

Instructions for installing and initially configuring a VPC-DI instance are available in the form of a Method Of Procedure (MOP) specific to your deployment requirements, platforms, operating system and hypervisor. The MOP is developed by AS in response to customer inputs.