Auto-Configuration

Auto-Configuration

This chapter briefly describes about the following:

- Configuration Profile
- Universal Profile
- Profile Refresh
- Profile Migration

Information About Auto-Configuration in DFA

Configuration Profile

A configuration profile in DFA is a collection of commands used to instantiate a specific configuration. Based on appropriate end-host triggers (VDP or data plane trigger (any data frame)), the configuration profiles are grouped to allow flexible and extensible options to instantiate day-1 tenant-related configurations on a DFA leaf on need basis.

The commands are entered using variables for certain parameters instead of entering the actual value. The switch then fills the actual values to derive the completed command. When the required parameters for a particular configuration profile are available, the profile can be instantiated to create a configuration set. The switch then applies this configuration set to complete the command execution belonging to the configuration set.

The commands which are supported under a configuration profile are called config-profile-aware commands. Most of the commands in the switch can be entered into the configuration profile.

Note Various sets of configuration profiles can be created and stored in the network database, and each network can use a different configuration profile. The configuration profiles can be used from the network to set up on the leaf whenever required.

Universal Profile

Note The Universal Profiles are available since 7.1(0)N1(1) for Cisco Nexus 5000 and 6000 Series Switches and from 7.2(0)D1(1) for Cisco Nexus 7000 Series Switches and 7.1(1) on Cisco Prime DCNM.

Universal profiles are enhanced configuration profiles with the ability to support optional parameters and non-disruptive profile refresh. The support is also extended to VRF profiles by using the **include profile any** command in the network individual VRF profiles. VRF profiles (vrf-common) need not be statically defined using POAP; instead select any universal VRF profile in Cisco Prime DCNM while creating a partition (vrf-common-universal, vrf-common, and so on). During the instantiation of the network, the system downloads the selected VRF profile for the partition and performs the necessary VRF, VLAN/SVI or bridge-domain/BDI and BGP configurations.

Profile Refresh

Profile refresh involves updating and/or removing profile parameters (arguments or variables) without disrupting the traffic while using universal profiles. After the changes are done, Cisco Prime DCNM executes the **fabric database refresh vni/dot1q/vrf** command.

The enhanced syslogs are generated when profile apply, profile un-apply, and profile refresh is performed and it contains details about the host that triggers the profile events.

Given below is a sample syslog profile:

```
Profile Download and Apply:
2014 Dec 11 10:51:37 hmm-n6k-9 %HMM-6-AUTO_CONF_PROFILE_DOWNLOADED: hmm [4238] VDP Host:
Profile 'BariProfile-v4-include-any'
(instance_def_0_1) downloaded for vni 301005
2014 Dec 11 10:51:42 hmm-n6k-9 %HMM-6-AUTO_CONF_PROFILE_APPLIED: hmm [4238] VDP Host
(000000000000000000000000064000000): Profile
'BariProfile-v4-include-any' (instance_vni_301005_1) applied for vni 301005 and got encap
2

Profile Refresh:
2014 Dec 11 10:53:07 hmm-n6k-9 %HMM-6-AUTO_CONF_PROFILE_REFRESH: hmm [4238] Data Snooping
Host: Refresh success for encap 1000

Profile Deletion:
2014 Dec 11 10:53:20 hmm-n6k-9 %HMM-6-AUTO_CONF_PROFILE_UNAPPLIED: hmm [4238] Data Snooping
 Host: Profile 'BariProfile-v4' unapplied for
encap 1000 and for vni 301000
```

Note Profile refresh can be achieved with individual network profiles, but it is treated as re-applying the profile. Therefore, updating parameters is disruptive.

Profile Migration

Migration involves moving from an existing applied individual profile to a universal profile, and it is necessary if features such as optional parameter and refresh are required. Migration needs to be done for all the networks under a partition/VRF at the same time. Within the same VRF/partition, a combination of universal and individual network profiles are not supported. To migrate the networks under partition/VRF, change the network profile from defaultNetworkProfile to defaultNetworkUniversalProfile, which is a disruptive process.

LDAP Configuration

There are three different tables, which you can query from:

- Network Table
- Partition Table
- Profile Table

Network Table

All the parameters for a network host are stored in this table in the LDAP. ToR will query this LDAP table for network parameters based on the following configuration:

```
Switch# configure terminal
Switch(config)# fabric database type network
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=networks, dc=cisco, dc=com key-type 1
```

Partition Table

All the parameters that are required to provision a VRF on the ToR are stored in this table. When a network is using a Universal Profile, querying this table can identify corresponding VRF profile. ToR queries this table for every new network that is provisioned using Universal Profile. The following configuration helps ToR to query the partition table:

```
Switch# configure terminal
Switch(config)# fabric database type partition
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=partitions, dc=cisco, dc=com
```

Profile Table

Multi-tenancy lite version

This table is used to store configuration profiles, which are pre-packaged with DCNM and custom config profiles that are created. The Cisco Nexus 5000 Series Switches or Cisco Nexus 6000 Series Switches employ this tenancy model where a maximum of 4K VLANs are supported on the ToR. So, if a profile is not pre-configured on the system then ToR will query profile table to download profile contents and cache it locally.

The following configuration helps ToR to query the profile table:

```
Switch# configure terminal
Switch(config)# fabric database type profile
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=profiles, dc=cisco, dc=com
```

Multi-tenancy full version

The Cisco Nexus 7000 Series Switches employ this tenancy model where, upto 4K VLANs or dot1q tags can be supported on a per port basis.

The following configuration helps ToR to query the profile table:

```
Switch# configure terminal
Switch(config)# fabric database type profile
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=profilesBridgeDomain, dc=cisco, dc=com
```

Auto-Pull Configuration

In certain scenarios, it may be required to manually provision a network based on the LDAP database. You can perform this from DCNM via the **Deploy** option or directly from a ToR using the following:

```
Switch# fabric database auto-pull <dot1q/vni> interface ethernet <id>
```

The trigger is similar to other triggers done by VDP or Non-VDP packet.

Specifying profile Mapping for network instances

It is necessary to specify how a particular network instance will be provisioned. That is either using network database lookup or by using a static Profile on the switch. The following configuration describes how to configure ToR to fetch network parameters from Remote database (LDAP).

```
Device(config)# fabric database profile-map global
Device(config-profile-map-global)# ethernet-tag encapsulation vni default dynamic
Device(config-profile-map-global)# vdp vni default dynamic
```

Configuring Windows 2012 as DHCP Server

You can have common DHCP-Servers (for example, Microsoft Windows) for IP address assignments within DFA. The DHCP-Servers can assign IP addresses to a simple DHCP request. The common DHCP-Server support does not rely on specific DHCP scope option (for example, simple-mode) by accepting some limitations or additional configuration.

We support Windows 2012 DHCP server by utilizing the 'Super Scope' as well as the policy on option 82 for address range selection. The DHCP policy on scope reserves the address space exclusively for the request matching the policy.

Note We support both Windows DHCPv4 and DHCPv6 servers and the configurations are similar to regular networks.

Let us assume the switch is using the address from subnet B (it can be the backbone subnet, management subnet, or any customer designated subnet for this purpose) to communicate with the Windows DHCP server. In DFA we have subnets S1, S2, S3, …, Sn for segment s1, s2, s3, …, sn.

To configure DHCP on Windows server.

1. Create a super scope. Within the super scope, create scope B, S1, S2, S3, …, Sn for the subnet B and the subnets for each segment.
2. In scope B, specify the 'Exclusion Range' to be the entire address range (so that the offered address range must not be from this scope).
3. For every segment scope Si, specify a policy that matches on Agent Circuit ID with value of '0108000600XXXXXX', where '0108000600' is a fixed value for all segments, the 6 numbers "XXXXXX" is the segment ID value in hexadecimal. Also ensure to check the **Append wildcard(*)** check box.
4. Set the policy address range to the entire range of the scope.

Configuring Infoblox as DHCP Server

Uses the Link Selection sub-option for scope selection, as this is by default set as the client facing SVI address. For other DHCP servers such as DHCPd and CPNR, GIAddr based scope selection is used. If you are already using Infoblox, then you must upgrade the Cisco NX-OS Switch to version 7.1(1)N1(1) or later.

Note We support only DHCPv4 for Infoblox and the configurations are similar to regular networks. You can refer to Infoblox user manual for configuration.

Let us consider a case where, the DHCP clients are VM hosts connecting to Cisco switches in DFA. The switches are configured with SVI as gateway for the VM hosts. The IP address of the SVI may not be unique in the DFA system, as when VM host moves to server connecting to another switch, then another SVI will be brought up on that switch and configured with the same gateway IP so that the VM does not need to change its gateway IP.

Configuring DHCPd as DHCP Server

The system has a centralized DHCP server that serves all VM hosts. Every switch has a DHCP relay agent running to forward the DHCP requests from VM hosts to the DHCP server. Because the SVI IP address is not unique, hence not reachable from the DHCP server, the relay agent on switch cannot use it as the GIAddr in the request. Instead, it uses another routing interface which has unique IP address as GIAddr. In order for the DHCP server to select the correct subnet for each host, the relay agent also put an identifier in the Circuit ID field in the Relay Agent Information option. The identifier uniquely identifies the subnet that a host connects to. However the identifier is only a portion of the Circuit ID.

Now on the DHCP server, you must configure it to fetch the identifier out of the Circuit ID and use the identifier to choose the right subnet. We are able to do this with DHCPd in the following way: we define classes matching on substring of the Circuit ID. All the host subnets are in a shared-network. The shared-network also contains the subnet for the routing interfaces on the switch, so that the shared-network will be picked when the request comes. The subnet for the routing interfaces does not have address pool, so it will not assign addresses. The address allocation is from the host subnets in the shared-network. Each host subnet only allows its own class members. Hence the server can correctly choose a subnet for address allocation based on the identifier carried in the request.

An example of the DHCPd configuration is given below. Here '59.2.8.0/24' and '99.1.3.0/24' are the host subnets, with identifier '01:5f:91' and '01:5f:92' respectively. Subnet '43.2.0.0/24' is the subnet of the routing interfaces. It is used to select the shared-network, but not used for address allocation.

```
# Start Segment 90001
class "15f91" {
match if substring (option agent.circuit-id, 5, 3) =01:5f:91;
}
# End Segment 90001

# Start Segment 90002
class "15f92" {
match if substring (option agent.circuit-id, 5, 3) =01:5f:92;
}
# End Segment 90002

shared-network "dfa-network" {

# Start Segment primarySubnet
subnet 43.2.0.0 netmask 255.255.255.0 {
}
# End Segment primarySubnet

# Start Segment 90001
subnet 59.2.8.0 netmask 255.255.255.0 {
```

```
option routers 59.2.8.1;
option vlan-id 90001;
 }
 pool {
allow members of "15f91";
range 59.2.8.2 59.2.8.254;
}
# End Segment 90001

# Start Segment 90002
subnet 99.1.3.0 netmask 255.255.255.0 {
option routers 99.1.3.1;
option vlan-id 90002;
 }
 pool {
allow members of "15f92";
range 99.1.3.2 99.1.3.254;
}
# End Segment 90002

}
```

Configuring a Profile

SUMMARY STEPS

1. **configure profile** *profile-name*
2. **interface vlan** *vlan-id* or **interface bdi** *bd-id*
3. **no shutdown**
4. **no ip redirects**
5. **include profile** *profile-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure profile** *profile-name* **Example:** `Switch(config)# configure profile profile1`	Creates a configuration profile.
Step 2	**interface vlan** *vlan-id* or **interface bdi** *bd-id* **Example:** `Switch(config)# interface vlan 50`	Defines a VLAN or BDI and enters interface configuration mode.
Step 3	**no shutdown** **Example:** `Switch(config-if)# no shutdown`	Administratively enables the interface.
Step 4	**no ip redirects** **Example:** `Switch(config-if)# no ip redirects`	Disables sending ICMP redirect messages.

	Command or Action	Purpose
Step 5	**include profile** *profile-name* **Example:** `Switch(config-if)# include profile profile2`	Includes a particular profile in another configuration profile, which will be uniquely instantiated for a set of parent profiles belonging to the same configuration set.

Example for Configuring an Individual Profile

Multi-tenancy lite version

The following is an example of an individual network configuration profile:

```
configure profile defaultNetworkIpv4EfProfile
 vlan $vlanId
 vn-segment $segmentId
 mode fabricpath
interface vlan $vlanId
 vrf member $vrfName
 ip address $gatewayIpAddress/$netMaskLength tag 12345
 ip dhcp relay address $dhcpServerAddr use-vrf default
 fabric forwarding mode proxy-gateway
 no ip redirects
 no shutdown
include profile vrf-common
```

Multi-tenancy full version

```
config profile defaultNetworkIpv4EfProfile
 vni $segmentId
 bridge-domain $bridgeDomainId
  member vni $segmentId
interface bdi $bridgeDomainId
 vrf member $vrfName
 ip address $gatewayIpAddress/$netMaskLength tag 12345
 ip dhcp relay address $dhcpServerAddr use-vrf management
 fabric forwarding mode proxy-gateway
 no ip redirects
 no shutdown
include profile vrf-common
```

Example for Configuring a Universal Profile

Multi-tenancy lite version

The following is an example for a universal configuration profile:

```
configure profile defaultNetworkUniversalEfProfile
 vlan $vlanId
 vn-segment $segmentId
 mode fabricpath
interface vlan $vlanId
 vrf member $vrfName
 ip address $gatewayIpAddress/$netMaskLength tag 12345
 ip dhcp relay address $dhcpServerAddr use-vrf $vrfDhcp
 ipv6 address $gatewayIpv6Address/$prefixLength tag 12345
 fabric forwarding mode proxy-gateway
 no ip redirects
 no ipv6 redirects
 no shutdown
```

```
include profile any
```

Multi-tenancy full version

The following is an example for a universal configuration profile:

```
configure profile defaultNetworkUniversalEfProfile
 vni $segmentId
 bridge-domain $bridgeDomainId
 member vni $segmentId
interface bdi $bridgeDomainId
 vrf member $vrfName
 ip address $gatewayIpAddress/$netMaskLength tag 12345
 ipv6 address $gatewayIpv6Address/$prefixLength tag 12345
 fabric forwarding mode proxy-gateway
 no ip redirects
 no ipv6 redirects
 no shutdown
include profile any
```

The following is an example for vrf-common-universal configuration profile:

```
configure profile vrf-common-universal
 vrf context $vrfName
 vni $include_vrfSegmentId
 rd auto
 ip route 0.0.0.0/0 $include_serviceNodeIpAddress
  address-family ipv4 unicast
    route-target import $include_borderLeafRt
    route-target both auto
  address-family ipv6 unicast
    route-target import $include_borderLeafRt
    route-target both auto
router bgp $asn
 vrf $vrfName
  address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
    maximum-paths ibgp 2
  address-family ipv6 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-V6HOST
    redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
    maximum-paths ibgp 2
```

The VRF profile is updated on the leaf resulting in the loopback routable IP address being auto-configured under that VRF as well as advertised via MP-BGP to all leaf nodes.

The following is an example of a profile that auto-configures a routable loopback interface per ToR per VRF. This profile is pre-packaged in DCNM and looks as follows. Any parameter prefixed with the keynote as 'system_auto_' indicates that the corresponding value will be auto-generated by the ToR.

```
configure profile vrf-common-loopback-universal
interface loopback $system_auto_loopbackId
  vrf member $vrfName
 ip address $system_auto_backboneIpAddress/32 tag 12345
  vrf context $vrfName
  vni $include_vrfSegmentId
 rd auto
 ip route 0.0.0.0/0 $include_serviceNodeIpAddress
  address-family ipv4 unicast
    route-target both auto
  address-family ipv6 unicast
    route-target both auto
router bgp $asn
 vrf $vrfName
  address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
    maximum-paths ibgp 2
```

```
address-family ipv6 unicast
redistribute hmm route-map FABRIC-RMAP-REDIST-V6HOST
redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
maximum-paths ibgp 2
```

Dynamic Provisioning

Note This section is applicable only for multi-tenancy full version.

Dynamic provisioning simplifies the management of the VRF and VLAN/BD configurations. Dynamic provisioning can be triggered by:

- Any data frame Frame snooping
- VDP signaling from the server

Per-port configuration can be used to enable or disable dynamic provisioning. When the VM comes up in a leaf switch for the first time, the port receives an ARP or ND packet and for cases where VDP is enabled, the hypervisor may issue a VDP packet. The platform is expected to punt these packets to the HMM component, which will then install a tenant profile based on the information in this packet. In the simplest case, a (Port, VLAN) information from the incoming packet will be used to derive the tenant profile that needs to be installed. The following needs to be implemented when the Vinci functionality is enabled on the switch to support this feature:

- The server-facing port needs to be configured with 'default VSI' and 'auto-config' must be enabled. The Ingress CBL state must be set to allow packets in all the VLANs. Note that for DHCP based snooping, 'feature dhcp' should be enabled.
- Global ACLs must be set up to punt the packets of interest such as ARP and ND to the HMM component. All the remaining packets must be dropped to retain the CBL behavior. The global ACLs must be used only for the (Port, VLAN) that has not been configured on the system.

Once the tenant profile is installed, the VLAN/BD configuration along with the associated port membership will be configured.

The following is an example for VN-Segment configuration:

```
system bridge-domain 2-3967

feature vni
 vni 5000, 10010-10011, 20000

 system fabric bridge-domain 3001-3967
 bridge-domain 2,10-11

vrf context management

 encapsulation profile vni all_tenants
 dot1q 6-204 vni 7002-7200
 encapsulation profile vni cisco_all
 dot1q 10-1010 vni 10010-11010
 bridge-domain 2,10-11
 member vni 5000,10010-10011

interface Ethernet3/6
 no shutdown
 service instance 3 vni
 no shutdown
```

```
 encapsulation profile cisco_all default
```

The following is an example for VN-Segment dynamic auto-configuration:

```
interface ethernet 2/2
  service instance vni default
      encapsulation dynamic vdp


interface ethernet 2/2
  service instance vni default
      encapsulation dynamic frame-snoop profile cisco
```