Configuring Autonomic Networking

Autonomic Networking

Autonomic networking makes network devices intelligent by introducing self-management concepts that simplify network management for network operators.

Prerequisites for Autonomic Networking

- The Autonomic Networking Infrastructure feature supports only Ethernet ports and IPv6 addresses.
- All interfaces are up by default to exchange adjacency discovery messages if there is no startup configuration in the corresponding device.
- The Autonomic Control Plane is automatically built between two adjacent devices supporting the autonomic networking infrastructure. The Ethernet interfaces on both devices need to be up, and the device should either be unconfigured (greenfield rollout) or have autonomic networking configured explicitly.
- The Autonomic Control Plane can also be automatically built between two adjacent devices if there is an intervening nonautonomic layer 2 cloud such as a Metro ethernet service. This is achieved by the Channel Discovery protocol on the autonomic devices. This protocol probes for working VLAN encapsulations.
- To build the ACP across intervening nonautonomic L3 devices, you should explicitly configure a tunnel between the autonomic devices and enable autonomic adjacency discovery on this tunnel.
- Autonomic Registrar, commonly known as *registrar*, is required for the Autonomic Networking Infrastructure feature to work. At least one device in the network must be configured as a registrar to enroll new devices into the autonomic domain.
- In a network where all the required devices are already enrolled into the autonomic domain, a registrar is not required.
- Each registrar supports only one autonomic domain. The registrar is needed only when new autonomic devices join the domain.
- To contact the registrar for enrolment to the autonomic domain, all new devices must have L2 reachability to at least one device that is already enrolled to the domain. If there is no L2 reachability, user needs to configure the tunnel between the devices and configure autonomic adjacency discovery on them.
- A device can be enrolled only into one autonomic domain. Two devices enrolled into different domains will not build the autonomic control plane between each other.

- Autonomic intent can be configured only on the registrar and from there it is propagated to all the devices in the domain.
- For Zero Touch Bootstrap to take place, there must be no startup-config file present and the config-register must remain default which is 0x2102.

Restrictions for Autonomic Networking

- Autonomic networking supports only unique device identifier (UDI) -based devices.
- Autonomic networking and Zero Touch Provisioning (ZTP) are different zero touch solutions. We recommended that you do not test or use autonomic networking and ZTP at the same time.
- All the devices in an autonomic network should be contiguously autonomic. If there is no continuity, manual configuration is required to configure a tunnel through a nonautonomic network.
- In Cisco IOS XE Denali 16.3.1 Release, Cisco Catalyst 3850 and Cisco Catalyst 3650 switches support only untagged probes and channel.
- Devices running Cisco IOS XE Denali 16.3.x Release and later are not compatible with devices running releases earlier than IOS XE 3.18 or 15.6(01)T. To facilitate interwork between these devices, autonomic adjacency discovery should be configured on the interfaces.
- When autonomic networking is enabled, you must not disable IPv6 unicast routing manually.
- The autonomic Registrar functionality is not supported in Cisco Catalyst 3850 and Cisco Catalyst 3650 switches.

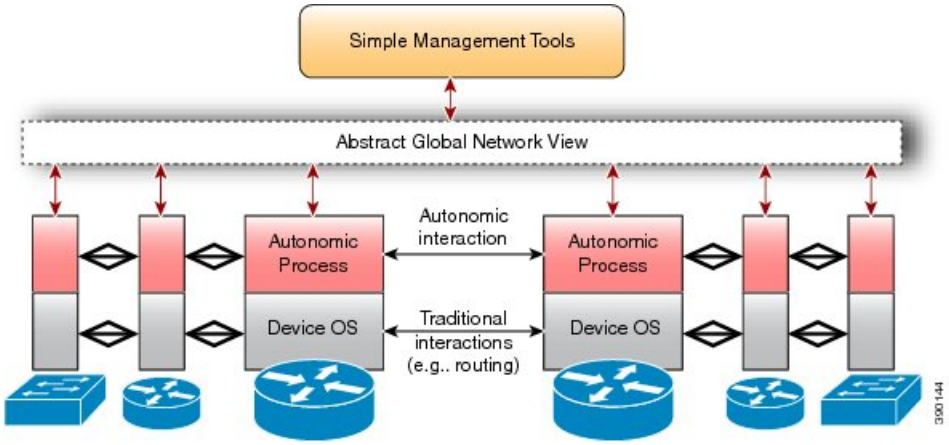
Information About Autonomic Networking

Overview of Autonomic Networking

The aim of autonomic networking is to create self-managing networks to overcome the rapidly growing complexity of the Internet and other networks to enable them to grow further. In a self-managing autonomic system, network management takes on a new role where, instead of controlling the network elements individually and directly, the administrator can define network-wide policies and rules to guide the self-management process.

The following figure provides a high-level architecture of an autonomic network.

Figure 1: High-Level Architecture of an Autonomic Network



Autonomic networking is controlled by a separate software entity running on top of traditional operating systems that include networking components, such as IP, Open Shortest Path First (OSPF), and so forth. Traditional networking components are unchanged and unaware of the presence of the autonomic process. The autonomic components use normal interfaces that are exposed by the traditional networking components and interact with different devices in the network. The autonomic components securely cooperate to add more intelligence to devices so that the devices in an autonomic network can autonomically configure, manage, protect, and heal themselves with minimal operator intervention. They can also securely consolidate their operations to present a simplified and abstracted view of the network to the operator.

Autonomic Networking Infrastructure

The Autonomic Networking Infrastructure feature simplifies the network bootstrap functionality by removing the need for any kind of prestaging, thereby allowing devices to join a domain securely, after which devices can be configured. The goal of the Autonomic Networking Infrastructure feature is to make new and unconfigured devices reachable by an operator or network management system, securely. This is carried as described here:

1. A device is defined and configured as the registrar. This registrar is the first autonomic domain device.
2. This step is optional. The network administrator collects a list of legitimate device identifiers of the devices to be added to the network. This list controls the devices that are added to the autonomic domain. Devices are identified by their unique device identifier (UDI). The list is compiled as a simple text file, one UDI per line. This step is optional because, in the absence of a whitelist, all the devices are allowed to join the domain. A whitelist is an approved list of entities that is provided a particular privilege, service, mobility, access, or recognition. Whitelisting means to grant access.
3. (Optional) The whitelist of known devices is uploaded to the registrar as part of its configuration.
4. Any new autonomic device that is directly connected to the registrar, or another enrolled domain device, will automatically receive a domain certificate from the registrar.
5. The autonomic control plane is automatically established across the autonomic domain to make new devices reachable.

The benefits of Autonomic Networking Infrastructure are as follows:

- Autonomic discovery of Layer 2 topology and connectivity by discovering how to reach autonomic neighbors.
- Secure and zero touch identity of new devices by using the device name and domain certificate.
- A virtual autonomic control plane that enables communications between autonomic nodes.

Autonomic behavior is enabled by default on new devices. To enable autonomic behavior on existing devices, use the **autonomic** command. To disable, use the **no** form of this command.

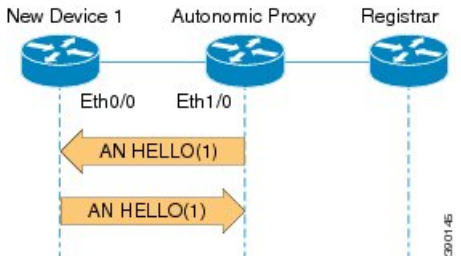
The components of autonomic networking are as follows:

- **Registrar**—A domain-specific registration authority in a given enterprise that validates new devices in the domain, provides them with domain-wide credentials, and makes policy decisions. Policy decisions can include a decision on whether a new device can join a given domain based on a preloaded whitelist. The registrar also has a database of devices that join a given domain and the device details.
- **Channel Discovery**—Used to discover reachability between autonomic nodes across nonautonomic Layer 2 networks.
- **Adjacency Discovery**—Used to discover autonomic neighbors. Adjacency discovery is performed on Layer 3. It is also possible to discover autonomic neighbors across pre-established Layer 3 Generic Routed Encapsulation (GRE) tunnels.

New Device Joining the Autonomic Network

The following figure illustrates how a new device joins an autonomic network.

Figure 2: New Device Joining the Autonomic Network



1. The new device sends out a hello message to the neighbor. In this case, the neighbor is part of an autonomic network domain.
2. The hello message includes the unique device identifier (UDI) of the new device.
3. The autonomic device acts as a proxy and allows the new device to join this autonomic network domain. The autonomic network device advertises itself with the domain information to its Layer 3 neighbors.
4. On receiving the autonomic network hello message from the neighbor and detecting the UDI information, the new device is validated with the autonomic registrar.
5. The new device advertises its domain certificate in its hello message with all neighbors. The neighbor information is exchanged every 10 seconds.

Note If the neighbor information changes, the entry is deleted and neighbor discovery is restarted. In the absence of a domain certificate and devices working with UDI, UDI is exchanged at a 10-second interval.

Channel Discovery in Autonomic Networking

Channel Discovery occurs automatically on all the interfaces when Autonomic Networking is enabled on the device. Note that autonomic Networking is enabled by default on devices with no configuration (greenfield devices, and assuming they have AN functionality), but will be passive. They will only be able to receive and answer CD probes, which are L2 frames. Only a device with domain certificate or one that is already enrolled to a domain can send out CD probes on all of its Ethernet interfaces that are up. As a result of this, neighbors

will be dynamically discovered. The probing will continue over time, so that newly added neighbors are discovered over time.

Adjacency Discovery in Autonomic Networking

After a channel is established, the proxy will send ND Hello messages to the new device, that is the one that is already enrolled in the domain and can act as a proxy for a new device joining the domain. The new device will send AN Hello messages in response back to the proxy. The Hello messages consist of an identification for the new device (UDI). On receiving AN Hello messages from the new device and detecting the UDI information, the AN proxy will send the details to the Autonomic Networking Registrar (ANR) for validating this new device.

Service Discovery in Autonomic Networking

Autonomic networking uses the multicast Domain Name System (mDNS) infrastructure to locate the various services required by the devices in the autonomic networking domain. A few of the services discovered by the network using the mDNS infrastructure are the AAA server, the configuration server, the syslog server, and the autonomic networking registrar. Autonomic networking listens to the mDNS advertisements on all the devices in the domain. From the devices hosting the services, autonomic networking initiates the mDNS advertisements.

Autonomic Control Plane

When a new device in the domain receives a domain certificate, it exchanges the domain certificate in the Hello messages with its neighbors. This creates an autonomic control plane between two autonomic devices of the same domain. There are different types of autonomic control planes that can be created based on the different capabilities of the devices. The autonomic control plane is established by using the following mechanisms:

- Configuring a loopback interface.
- Dynamically assigning an IPv6 address to the loopback interface.
- Configuring autonomic VPN routing and forwarding (VRF).

Autonomic Networking Intent

Overview of Autonomic Networking Intent

There is a need to allow network wide policies in certain autonomic deployments. Channel Discovery on active autonomic devices probe VLAN IDs on all interfaces. However, the possible VLAN ID range used across layer 2 clouds is significantly smaller than the 4095 possible VLAN IDs. Channel discovery probes are sent every 10 seconds, so probing all VLAN IDs can potentially take a long time. Limiting the VLAN ID range for channel discovery can decrease the bootstrap time for new devices.

The configuration of the VLAN ID range is done in a network wide fashion. The VLAN ID is configured on one device and the configuration is flooded through the entire network automatically. This is achieved using the Intent Distribution Protocol, which works as follows:

- Every autonomic device sends IDP messages with the current timestamp, the last time that intent has been ingested, to its neighbors. This floods the timestamp on the entire domain.
- A user ingests an intent at any ‘allowed’ place in the network.
- This increases the intent version number on that device, while it also locally interprets the intent.
- Neighbors notice the revision in the timestamp and pull in the newest revision of the intent from the neighbor that indicates the newer version number.

- This updates the intent in the network, and all nodes execute the intent locally.

Key Features of Intent

The key features of autonomic networking intent is as follows:

- Unambiguous configuration—An unambiguous representation of the intent is provided by the network. The **autonomic intent** command is converted and stored as an XML file in the flash. Thus, it is easy to parse and is unambiguous as each XML tag is unique.
- Persistency and reload—The intent is persistent across reloads. The intent configuration is not nvgen'ed on any device. The persistency is ensured via the XML file in flash, which will be parsed when autonomic is enabled after a reload, whether via manual autonomic configuration or via autonomic configuration in the startup configuration. After the intent file is parsed, channel discovery probes going out remain in the intent range, but, incoming channel discovery probes could be outside the intent range. This establishes a temporary channel discovery channel and an autonomic control plane with a neighboring device having the latest version of intent. After the new device receives a new intent from the neighboring device, the temporary channel becomes stale and is deleted so that a new channel is created conforming to the new intent range.
- Intent Security—The intent is created only on the registrar. The registrar adds the certificate and signs the contents before distributing the intent. Any changes in the intent configuration results in resigning the intent. The receiving devices use the public certificate received along with intent to verify the signature. On verification, the devices copy the intent and, if required, redistribute the new intent to its neighbors. Signature verification will fail if intent is modified.
- Default Intent—When the intent is removed from the registrar by using the **no** form of the command, a default intent will be generated and propagated. The default intent is just like a new intent but contains the keyword “default” in the *outer vlans* tag in the XML file.
- Multiple Registrars—There could be multiple registrars in the network, for redundancy purposes. Because all devices are in the same domain they must have the same intent version. Intent distribution protocol allows intent to be injected at any registrar and also propagate the intent to all neighbors in the given domain. This ensures that the latest intent version is maintained across all devices.

Intent Configuration

The place in the network to ingest that network wide configuration (or Autonomic Intent) is limited to the registrars. Use the **autonomic intent** command to configure intent followed by the **acp outer-vlans** command.

This configuration is not saved in the NVRAM because the intent pertains to network wide configuration and is, hence, stored ‘in the network.’ If any device, including the registrar, *forgets* about the intent, the IDP ensures that the device receives a copy of the intent immediately.

You cannot configure the intent on nonregistrar autonomic devices. However, on receiving the XML intent file, the nonregistrar autonomic devices parse the file and store the VLAN ID values.

When autonomic is disabled, the intent file is deleted from flash. If the intent file is deleted from flash, you must wait until the neighboring devices pushes the latest intent (which might happen when the autonomic control plane is and reestablished later) or when the neighboring devices receive a newer intent.

Working of Intent

When the outer VLAN range is configured on the registar, an intent XML file, an_intent_cp.xml, is created in the flash. If theXML file exists, it is deleted and a new file is created. A version number, which is the version number, is populated in the XML file with the domain ID.

In case of nonregistrar devices, a file with the name, an_intent_cp.xm, is created in flash to store the received intent. This ensures that a device, after a reboot, can execute the intent immediately, when autonomic is enabled. This file should not be deleted.

The following is a sample of the XML file:

```
<autonomic intent>
<acp>
<outer vlans>
30,35,50-60,25,80
</outer vlans>
</acp>
</autonomic intent>
</autonomic domain cisco.com>
<ANR signature>
SAFKJNKG3242107FEGEGEF3234
</ANR signature>
<ANR public cert>
NHSKLFF901IF112E13R3938RY093
FREIFJFIGIO3FOIHF03F3JFNII3FJG
RGFRGFNKRNGR4R980-921D23-R
JFDPOIEJF02IGFR3209RU309UF3IF
3KG3GF03IGF3-IF3KGFK3[P[KGF3
</ANR public cert>
```

The following is a high level overview of how intent works for a new device and an existing device:

- New Device—When the autonomic control plane of a new device is up, IDP message is triggered from a neighbor device. The neighbor device fetches the version number from the XML file and send a message to the new device. Sometimes, a neighbor might request for intent (via another IDP message) if its version number is lower or if does not havea version at all. In such cased, the sending device parses the contents of XML file and constructs an IDP message whose payload includes the XML file, the signature, and the public certificate. This IDP message is unicast over the ACP of the device whose ACP just came up.
- Existing Device—A device recieves a new intent when the device has no intent or the existing intent version number, on the device, is lower than the recieved intent. When a device, having the XML file exists on the device recieves a new intent, the device verifies the signature of the intent and overwrites the existing intent, along with new signature and new certificate. After the intent is applied, all existing channel discovery channels are verified for stale channels. Stale channels are channels with outer encapsulationthat fall outside the range of the new intent. Stale channels are removed to create new channels based on new intent. The outgoing channel discovery probes must be sent for VLANs only in the range specified by the intent.

Note Irrespective of intent availabilty, all incoming channel discovery packets are allowed whether they fall in the intent VLAN range. This is to ensure that there is no deadlock when a new intent is propogated in the network.

How to Configure Autonomic Networking

Configuring the Registrar

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **autonomic**
4. **autonomic registrar**
5. **domain-id** *domain-name*
6. **device-accept** *udi*
7. **whitelist** *filename*
8. **no shut**
9. **exit**
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**autonomic** **Example:** `Device# autonomic`	Enables autonomic networking.
Step 4	**autonomic registrar** **Example:** `Device(config)# autonomic registrar`	Enables a device as a registrar and enters registrar configuration mode.
Step 5	**domain-id** *domain-name* **Example:** `Device(config-registrar)# domain-id abc.com`	Represents a common group of all devices registered with the registrar.
Step 6	**device-accept** *udi* **Example:** `Device(config-registrar)# device-accept PID:A901-12C-FT-D SN:CAT1902U88Y`	(Optional) Specifies the UDI of a quarantined device to be accepted in the autonomic domain. **Note** This command is not required when configuring the registrar. It is required only after the registrar is enabled to accept previously quarantined devices.
Step 7	**whitelist** *filename* **Example:** `Device(config-registrar)# whitelist flash:whitelist.txt`	(Optional) Allows loading a file on the local device that contains a list of devices to be accepted in a given domain. The file must contain one UDI entry per line. **Note** If this command is not configured, all the devices are accepted into the domain.

	Command or Action	Purpose
Step 8	**no shut** **Example:** `Device(config-registrar)# no shut`	Enables the autonomic registrar.
Step 9	**exit** **Example:** `Device(config-registrar)# exit`	Exits registrar configuration mode and returns to global configuration mode.
Step 10	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

How to Configure Autonomic Networking

Configuring the Intent

Before you begin

A registrar must be configured with a domain ID and must be up by executing the **no shutdown** command.

Note The intent can only be configured on the registrar.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **autonomic intent**
4. **acp outer-vlans** *vlan-id-range*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**autonomic intent** **Example:** `Device(config)# autonomic intent`	Configures intent on the registrar and enters intent configuration mode.
Step 4	**acp outer-vlans** *vlan-id-range* **Example:** `Device(config-intent)# acp outer-vlans 30,35,50-60,25,80`	Configures the VLAN range.
Step 5	**end** **Example:** `Device(config-intent)# end`	Exits intent configuration mode and returns to privileged EXEC mode.

Verifying and Monitoring Autonomic Networking Configuration

SUMMARY STEPS

1. **enable**
2. **show autonomic device**
3. **show autonomic neighbors** [**detail**]
4. **show autonomic control-plane** [**detail**]
5. **show autonomic l2-channels** [**detail**]
6. **show autonomic interfaces**
7. **debug autonomic** {**Bootstrap** | **Channel-Discovery** | **Infra** | **Intent** | **Neighbor-Discovery** | **Registrar** | **Services** } {**aaa** | **all** | **ntp** | **events** | **packets**} {**info** | **moderate** | **severe**}
8. **clear autonomic** {**device** | **neighbor** *UDI* | **registrar accepted-device** *device UDI*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show autonomic device** **Example:** `Device# show autonomic device`	Displays the current state of an autonomic device including the global details.
Step 3	**show autonomic neighbors** [**detail**] **Example:** `Device# show autonomic neighbors detail`	Displays information about the discovered neighbors.
Step 4	**show autonomic control-plane** [**detail**] **Example:**	Displays information about the autonomic control plane.

	Command or Action	Purpose
	`Device# show autonomic control-plane detail`	
Step 5	**show autonomic l2-channels** [**detail**] **Example:** `Device# show autonomic l2-channels`	Displays the results of Channel Discovery.
Step 6	**show autonomic interfaces** **Example:** `Device# show autonomic interfaces`	Displays information about the interfaces in the autonomic domain.
Step 7	**debug autonomic** {**Bootstrap** | **Channel-Discovery** | **Infra** | **Intent** | **Neighbor-Discovery** | **Registrar** | **Services** } {**aaa** | **all** | **ntp** | **events** | **packets**} {**info** | **moderate** | **severe**}	Enables debugging of the autonomic network.
Step 8	**clear autonomic** {**device** | **neighbor** *UDI* | **registrar accepted-device** *device UDI*}	Clears or resets autonomic information. • The **clear autonomic device** command clears or resets all the device-specific autonomic networking information, including the information obtained during the bootstrapping process. • The **clear autonomic neighbor** command clears the neighbor-related information obtained during the neighbor discovery process. If no neighbor is specified, it clears the entire neighbor database. • The **clear autonomic registrar accepted-device** command clears the public key stored for each device enrolled by the registrar.

Verifying Autonomic Intent Configuration

SUMMARY STEPS

1. **enable**
2. **show autonomic intent**
3. **show autonomic intent neighbors**

DETAILED STEPS

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show autonomic intent**

Example:

```
Device# show autonomic intent

Intent File  : Available
Version Num  : 1395908791 (Parsed)
Version Time : 2014-03-27 13:56:31 IST
Outer Vlans  : 30,35,50-60,25,80
DB count     : 15
Vlans in DB  : 25, 30, 35, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 80,
```

Displays the status and details of the intent of the device.

Step 3 **show autonomic intent neighbors**

Example:

```
Device# show autonomic intent neighbors

Neighbor                                                Intent version
-------------------------------------------------------------------------------
PID:Unix SN:1652527195                                  1395908791
```

Displays the status and details of the intent of the neighboring devices.

Configuration Examples for Autonomic Networking

Example: Configuring the Registrar

```
Device> enable
Device# configure terminal
Device(config)# autonomic registrar
Device(config-registrar)# domain-id abc.com
Device(config-registrar)# whitelist flash:anra-domain.txt
Device(config-registrar)# end
```

Example: Configuring and Verifying Autonomic Networking Intent

```
Device> enable
Device# configure terminal
Device(config)# autonomic
Device(config)# autonomic intent
Device(config-intent)# acp outer-vlans 30,35,50-60,25,80
Device(config-intent)# end
Device# show autonomic intent

Intent File  : Available
Version Num  : 1395908791 (Parsed)
Version Time : 2014-03-27 13:56:31 IST
Outer Vlans  : 30,35,50-60,25,80
DB count     : 15
Vlans in DB  :
25, 30, 35, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 80,

Device# show autonomic intent neighbors

Neighbor                                                Intent version
```

```
--------------------------------------------------------------------------------
PID:Unix SN:1652527195                              1395908791
```