ATEA TECH TOUR

Automated deployment of Cisco SD-Access fabric site

ATEA TECH TOUR

Jesper Munk

Senior Systems Engineer, Network & Security, Atea

ATEA TECH TOUR

Agenda

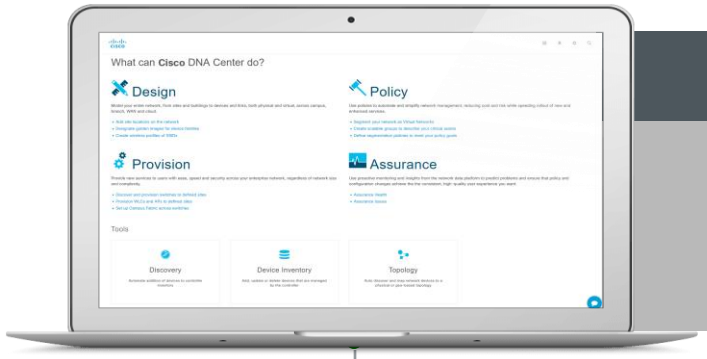
- **Introduction:** Cisco Software Defined Access (SDA), and why to automate it
- Prerequisites and Business Outcome
- Automation Functionality
- Automation Design
- **Python Script:** State Event Machine with fixed sequence of states
- Automation Methods
- **Trade-off:** script functionality vs development & maintenance costs, KEEP IT SIMPLE
- Possible Improvements

ATEA TECH TOUR

Agenda

- **Introduction:** Cisco Software Defined Access (SDA), and why to automate it
- Prerequisites and Business Outcome
- Automation Functionality
- Automation Design
- **Python Script:** State Event Machine with fixed sequence of states
- Automation Methods
- **Trade-off:** script functionality vs development & maintenance costs, KEEP IT SIMPLE
- Possible Improvements

Cisco Catalyst Center – controller for Campus traditional network and SD-Access Fabric



Cisco DNA Center

Policy | Provision | Design | Assurance

Cisco DNA Center Appliance

Physical and virtual infrastructure

Cisco and third party

Complete network management system

- Single pane of glass for all devices
- End-to-end health information in real time
- Granular visibility
- Simplified workflows

Automation for provisioning

- Zero-touch deployment
- Device lifecycle management
- Policy enforcement

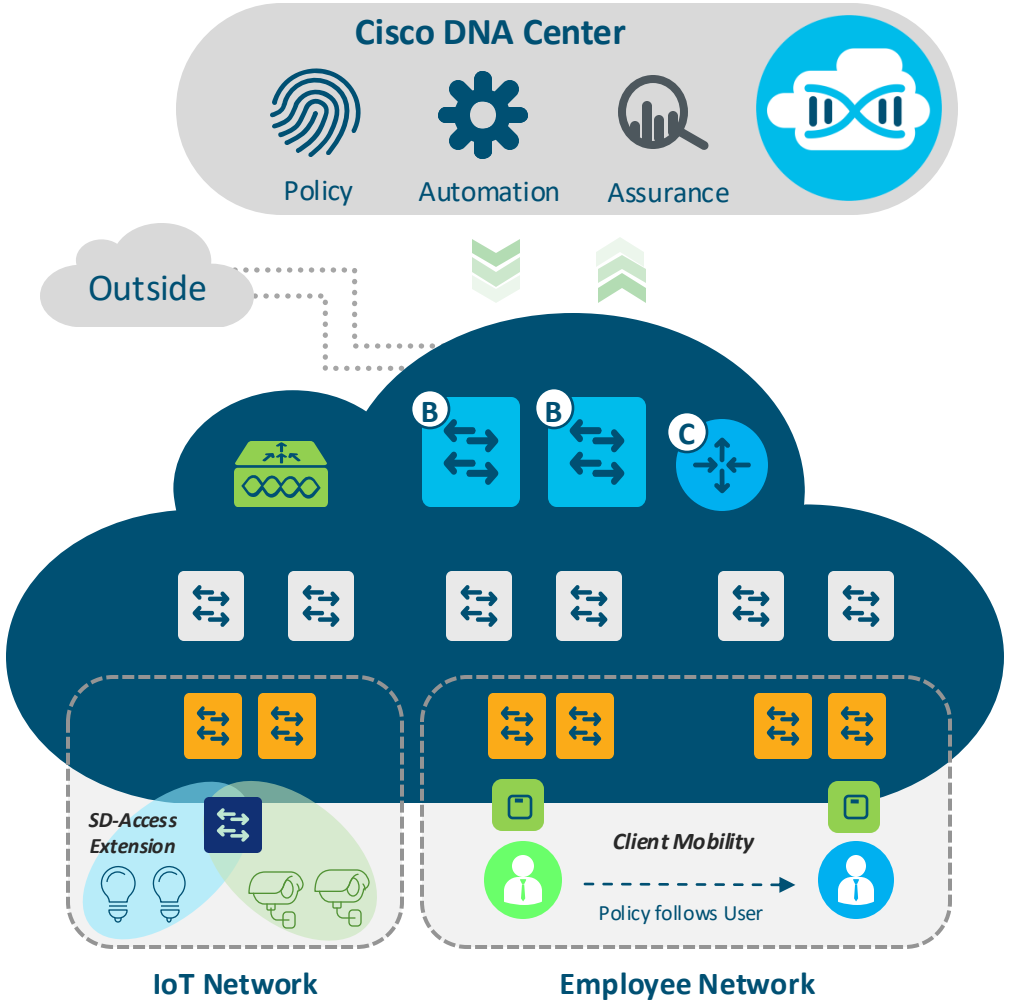
Analytics for assurance

- Verify intent of network settings
- Proactively resolve issues
- Reduce time spent troubleshooting

Platform for extensibility

- Integrate APIs with third-party solutions
- Integrate and customize ServiceNow
- Evolve operational tools and processes

Cisco Software Defined Access: Architecture



One Automated Network Fabric

Single fabric for Wired and Wireless with full automation

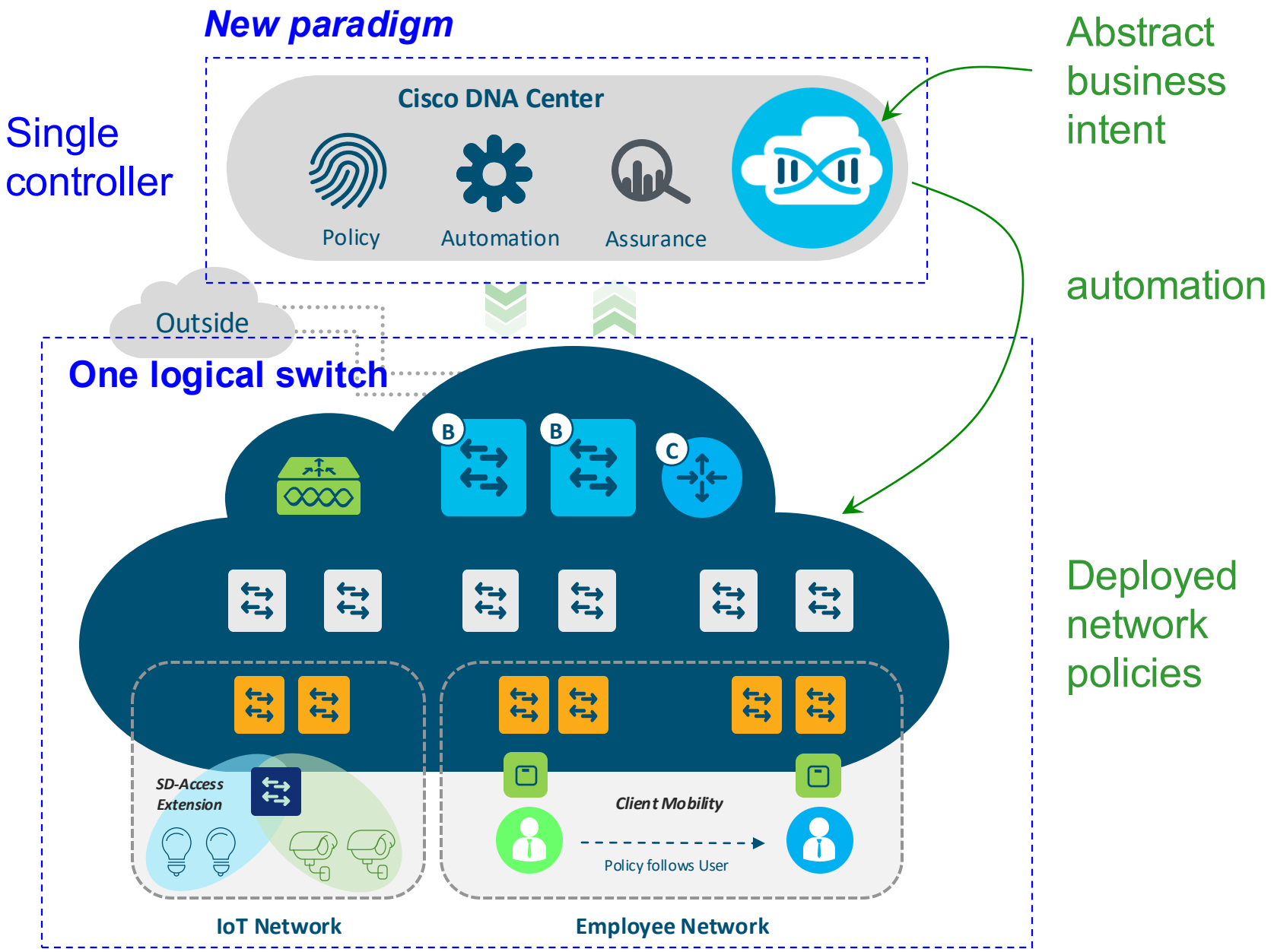
Identity-Based Policy and Segmentation

Policy definition decoupled from VLAN and IP address

AI-Driven Insights and Telemetry

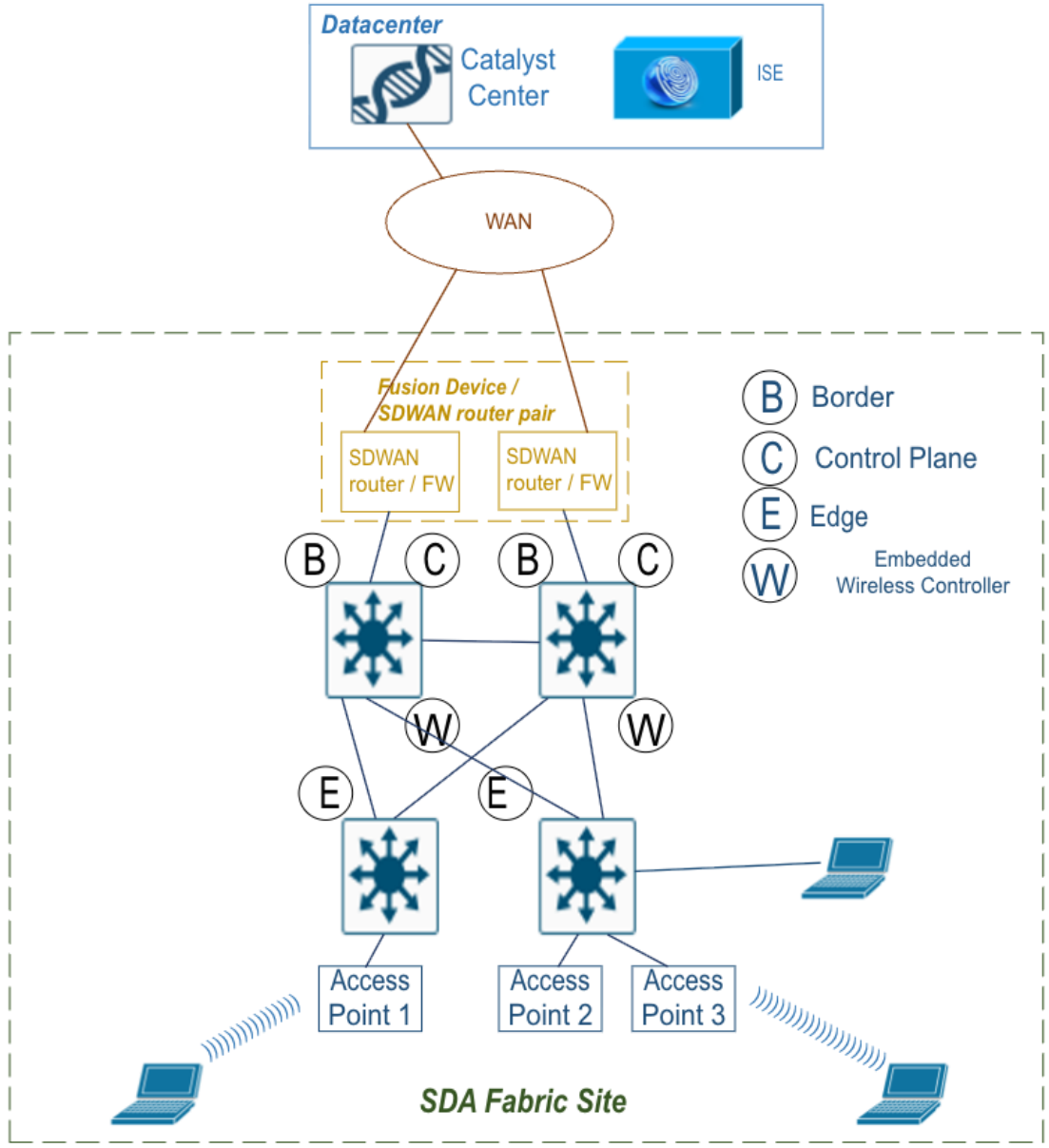
Analytics and visibility into User and Application experience

Cisco Software Defined Access: Architecture



SDA Site

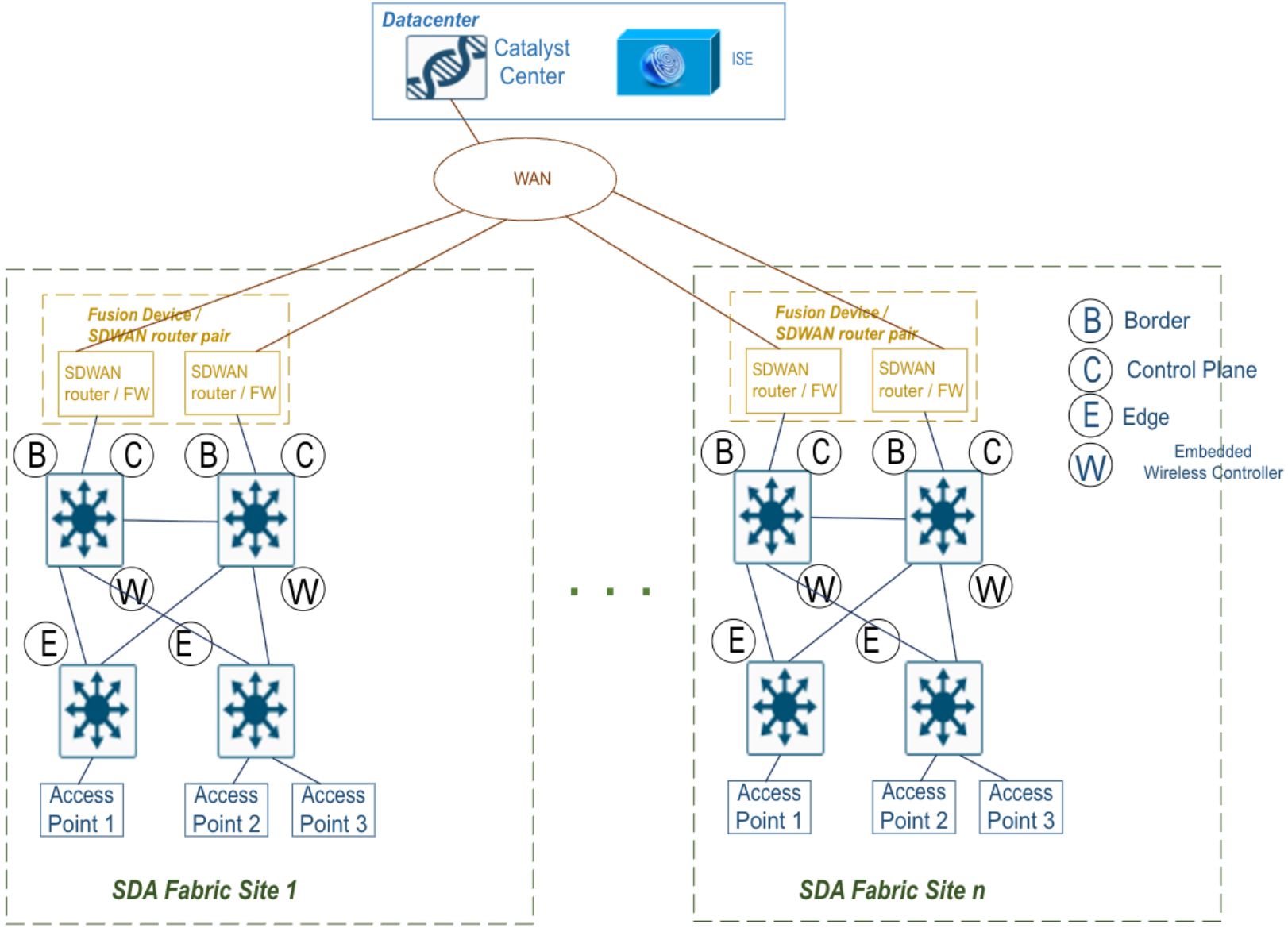
- The architecture of SDA means, that *automation of SDA will consist of API calls to Catalyst Center only*, not to the individual network devices.
- The exception being settings on the WLC, embedded in Catalyst 9000 switch, which are not controlled by Catalyst Center.
- In the present case this applies to renaming of Access Points.



ATEA TECH TOUR

SDA Solution with Many Sites

SDA site automation pays off when many SDA sites with similar design



ATEA TECH TOUR

Agenda

- **Introduction:** Cisco Software Defined Access (SDA), and why to automate it
- Prerequisites and Business Outcome
- Automation Functionality
- Automation Design
- **Python Script:** State Event Machine with fixed sequence of states
- Automation Methods
- **Trade-off:** script functionality vs development & maintenance costs, KEEP IT SIMPLE
- Possible Improvements

Prerequisites and Business Outcome

- **Prerequisites**
  - Many SDA sites with similar design
    - Physical cabling standard
    - Similar logical design
- **Business Outcome**
  - Effective deployment of SDA sites
  - No human errors

Note:

- If a company has other SDA site designs than described here, or varying SDA site designs, then additional script development costs may occur.

ATEA TECH TOUR

Agenda

- **Introduction:** Cisco Software Defined Access (SDA), and why to automate it
- Prerequisites and Business Outcome
- Automation Functionality
- Automation Design
- **Python Script:** State Event Machine with fixed sequence of states
- Automation Methods
- **Trade-off:** script functionality vs development & maintenance costs, KEEP IT SIMPLE
- Possible Improvements

Automation Functionality

Automatically deploys an entire SDA site

- **Prerequisite:** common central configurations in Catalyst Center are in place
- Deployment of Embedded WLC in Catalyst 9000 switch done manually (API missing)

ATEA TECH TOUR

Agenda

- **Introduction:** Cisco Software Defined Access (SDA), and why to automate it
- Prerequisites and Business Outcome
- Automation Functionality
- Automation Design
- **Python Script:** State Event Machine with fixed sequence of states
- Automation Methods
- **Trade-off:** script functionality vs development & maintenance costs, KEEP IT SIMPLE
- Possible Improvements

ATEA TECH TOUR

Automation Design

- **Python script**
- **Configuration data in YAML file**
  - YAML is “hierarchical database” in text file in fairly human readable format
  - YAML file must among other things contain
    - Number of downlinks from first SDA border node to Fabric Edge nodes and intermediate nodes
    - Serial number, placement in site hierarchy and host name for all switches and Access Points, and device usage for switches (border/Fabric Edge/Intermediate)

Automation Design:
YAML configuration data contents

IP Address Pools

```yaml
ip_address_pools:
- name: dkblp-border-handoff
  ipv4DhcpServers:
  - 172.19.40.10
  ipv4DnsServers:
  - 172.19.40.10
  ipv4GateWay:
  ipv4GlobalPool: 100.64.0.0/10
  ipv4Prefix: True
  ipv4PrefixLength: 24
  ipv4Subnet: 100.96.12.0
  type: Generic
- name: dkblp-corporate
  ipv4DhcpServers:
  - 172.19.40.10
  ipv4DnsServers:
  - 172.19.40.10
  ipv4GateWay: 100.112.0.1
  ipv4GlobalPool: 100.64.0.0/10
  ipv4Prefix: True
  ipv4PrefixLength: 23
  ipv4Subnet: 100.112.0.0
  type: Generic
```

Devices to be discovered during LAN Automation

```yaml
discovery_devices:
- device_serial_number: FCW2127G05W
  device_type: fabric_edge # Alternative: intermediate
  device_host_name: ntni-sde-001-dkblp-b00
  device_site_name_hierarchy: Lautrupvang 6/Lautrupvang 6A/B00
  device_management_ip_address: 100.96.0.70
- device_serial_number: FCW2127G05A
  device_type: fabric_edge
  device_host_name: ntni-sde-002-dkblp-b00
  device_site_name_hierarchy: Lautrupvang 6/Lautrupvang 6A/B00
  device_management_ip_address: 100.96.0.65
```

Python script: way of using

- All network devices including Access Points must be factory reset, cabled and powered up before start of automation
- Hereafter run script which will automatically provision everything
- Exception: embedded C9800 WLC in Catalyst 9000 switch must be manually provisioned (API missing)

Note: Based on the use of cabling standard, and the information in the YAML configuration data file about the number of downlinks from first SDA border node to Fabric Edge and intermediate nodes, and the serial number and site hierarchy of all network devices, *LAN Automation* will handle any physical network toplogy

ATEA TECH TOUR

Agenda

- **Introduction:** Cisco Software Defined Access (SDA), and why to automate it
- Prerequisites and Business Outcome
- Automation Functionality
- Automation Design
- **Python Script:** State Event Machine with fixed sequence of states
- Automation Methods
- **Trade-off:** script functionality vs development & maintenance costs, KEEP IT SIMPLE
- Possible Improvements

ATEA TECH TOUR

Python script:

State Event Machine with Fixed Sequence of States

- Python script structured as State Event Machine
- Script goes through fixed sequence of states
- In each state, the script executes one action

Same sequence of actions, as when you do it manually from the Catalyst Center GUI.

Python script:
Fixed Sequence of States

State 1: Create site area
State 2: Create site building area, if any
State 3: Create building
State 4: Create floor
State 5: Create Fabric Site
State 6: Create IP address pool
State 7: Add default authentication profile to fabric site
State 8: Add virtual network to fabric site
State 9: Add IP address pool to virtual network
State 10: Update Wireless Network Profile with new sites
State 11: Plug and Play deploy SDA borders
State 11AA: Sync SDA borders
State 11A: Change management IP address on SDA borders
State 12: Provision SDA borders
State 12A: Sync SDA borders
State 13: Add SDA borders to Fabric (L3 handoff to be added later)
State 13AA: Sync SDA borders
State 13A: Add SDA border L3 handoff
State 13B: Sync SDA borders

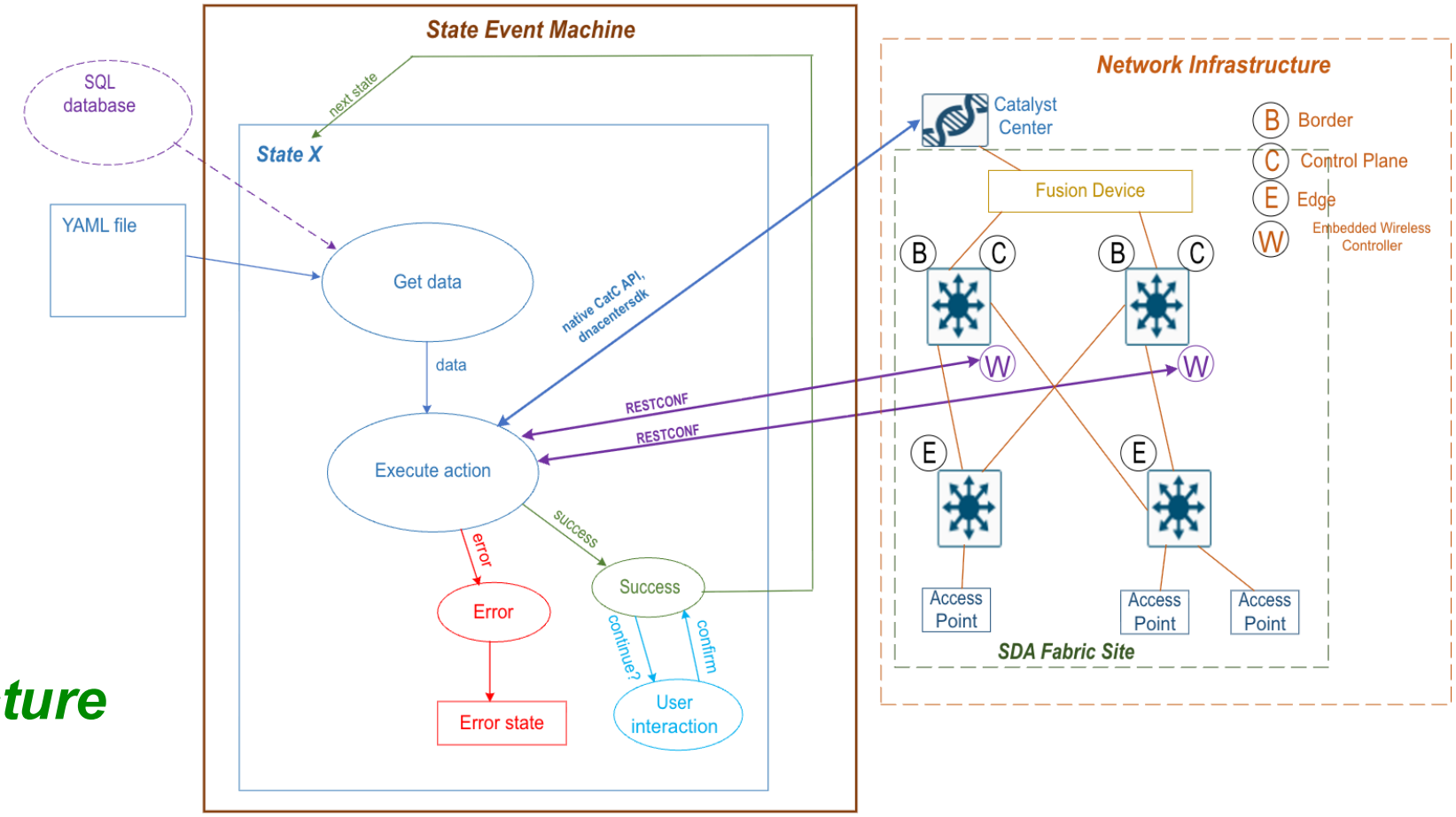
State 14: Map SSID to VLAN
State 14A: Add cross border links w LAN Automation
State 14B: Sync SDA borders
State 15: LAN Automation
State 15A: Sync all Fabric and Intermediate nodes
State 16: Provision IAN Automated Fabric Edge nodes and intermediate nodes (if any)
State 16A: Sync all Fabric and Intermediate nodes
State 17: Add fabric edge nodes to fabric
State 17A: Sync all Fabric Edge nodes
State 18: Configure fabric edge node ports for Access Points
State 18A: Sync all Fabric Edge nodes
State 19: Claim Accesss Points to site
State 20: Resync Embedded WLCs a number of times to speed upprovisioning of Access Points
State 21: Rename Provisioned Access Points and resync Embedded WLCs

Same sequence of actions as when you do it manually

Python script:
Video excerpts of site automation

- **Add Layer 3 hand-offs to SDA borders**
  - File *SDA site automation add L3 handoffs.mov*
- **Rename Access Points to names according to naming standard**
  - File *SDA site automation rename Access Points.mov*

Python script: State Event Machine



Infrastructure as Code:

Configuration data + Code ======> Physical Network Infrastructure

ATEA TECH TOUR

Agenda

- **Introduction:** Cisco Software Defined Access (SDA), and why to automate it
- Prerequisites and Business Outcome
- Automation Functionality
- Automation Design
- **Python Script:** State Event Machine with fixed sequence of states
- Automation Methods
- **Trade-off:** script functionality vs development & maintenance costs, KEEP IT SIMPLE
- Possible Improvements

Automation Methods

- **Native API calls to Catalyst Center**
  - Full Cisco support
- **dnacentersdk API calls to Catalyst Center**
  - Community maintained
- **RESTCONF API calls to Catalyst 9000 IOS-XE switch** (Embedded WLC)
  - Full Cisco support
- **Catalyst Center basic automation: CLI template**
  - Full Cisco support

Automation Methods:
Native Catalyst Center API call

- Full control of functionality
- Full Cisco support
- You must handle connection to Catalyst Center and errors Yourself
- In some cases used because dnacentersdk did not have functionality or dnacentersdk did not work

Reference:
https://developer.cisco.com/docs/dna-center/

Automation Methods: RESTCONF

- Used for automation of Cisco IOS-XE network devices
- Based on YANG data model
- REST like “operational model”: Create, Read, Update, Delete (CRUD)
- Data in XML or JSON
- In present automation solution used for renaming of Access Points in embedded Catalyst 9800 WLC in Catalyst 9000 IOS-XE switch

Automation Methods: RESTCONF

Renaming of Access Point in present automation solution

```
# build RESTCONF URI
this_uri = "https://{}:443/restconf/operations/Cisco-IOS-XE-wireless-access-point-cfg-rpc:set-ap-name".
        format(device_ip)
logger.debug("The following RESTCONF URI will be used to rename the AP: {}\n".format(this_uri))

# build JSON payload
payload = '{"set-ap-name":[{"name":"' + new_ap_name + '","ap-name":"' + old_ap_name + '"}]}'
logger.debug("\nthe following payload will be sent: {}".format(payload))

try:
        response = requests.request("POST", this_uri, auth = HTTPBasicAuth(device_user_name,
        device_password), headers = {"accept": "application/yang-data+json",
        "content-type": "application/yang-data+json",}, verify=False, data=payload)
```

Automation Methods: dnacentersdk

- Handles connection to Catalyst Center and many errors
- Community maintained
- Used in present automation solution where dnacentersdk calls were available
- A few dnacentersdk calls did not work,
  then native Catalyst Center API calls were used instead

Reference:
https://pypi.org/project/dnacentersdk

Automation Methods: dnacentersdk

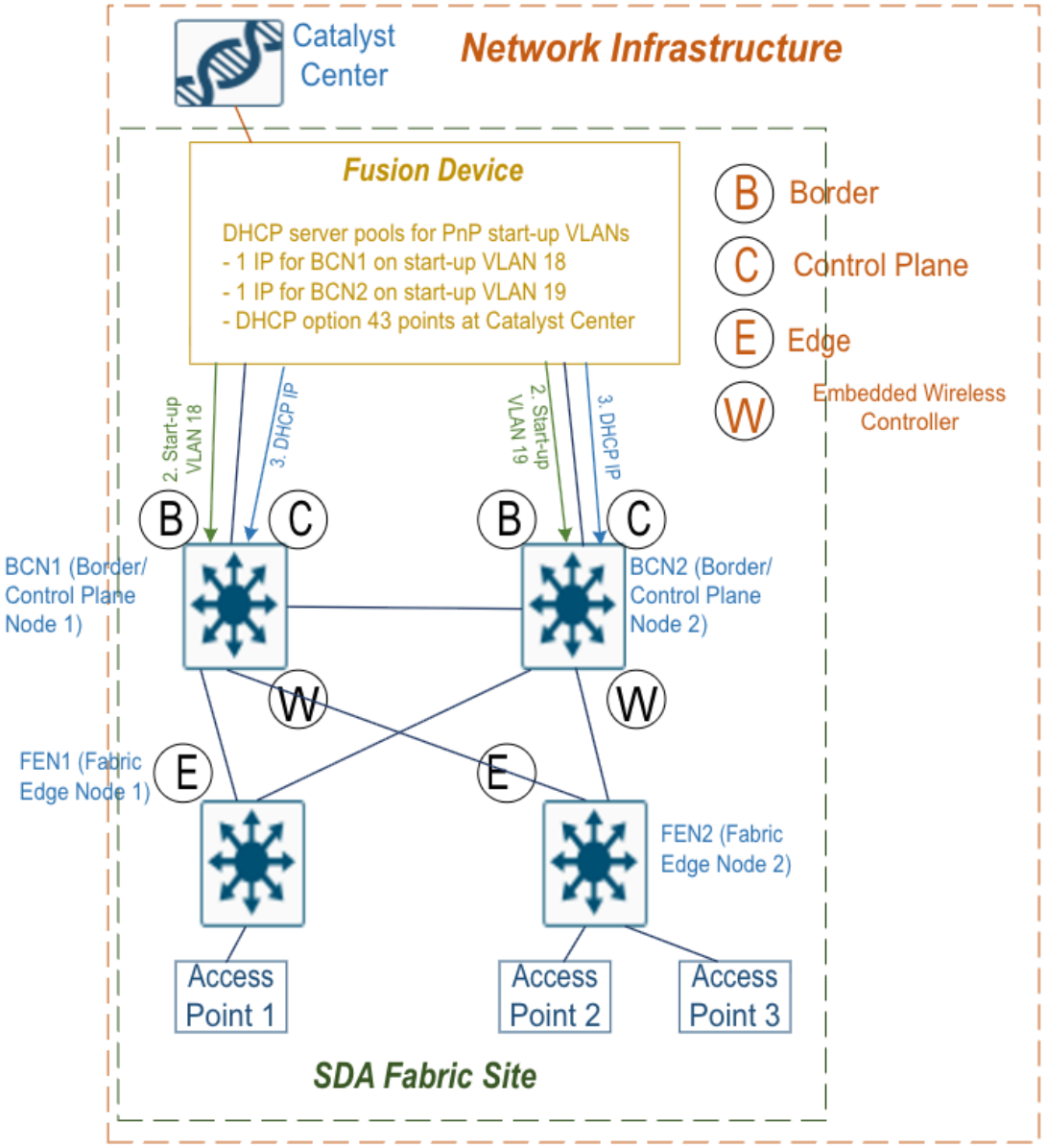
Example: get device list from Catalyst Center
and then close connection to Catalyst Center
(from: https://pypi.org/project/dnacentersdk/)

```
from dnacentersdk import DNACenterAPI

dnac = DNACenterAPI()
try:
    devices = dnac.devices.get_device_list()
finally:
    dnac.close()
```

Catalyst Center Basic Automation (CLI Template) (1/2)

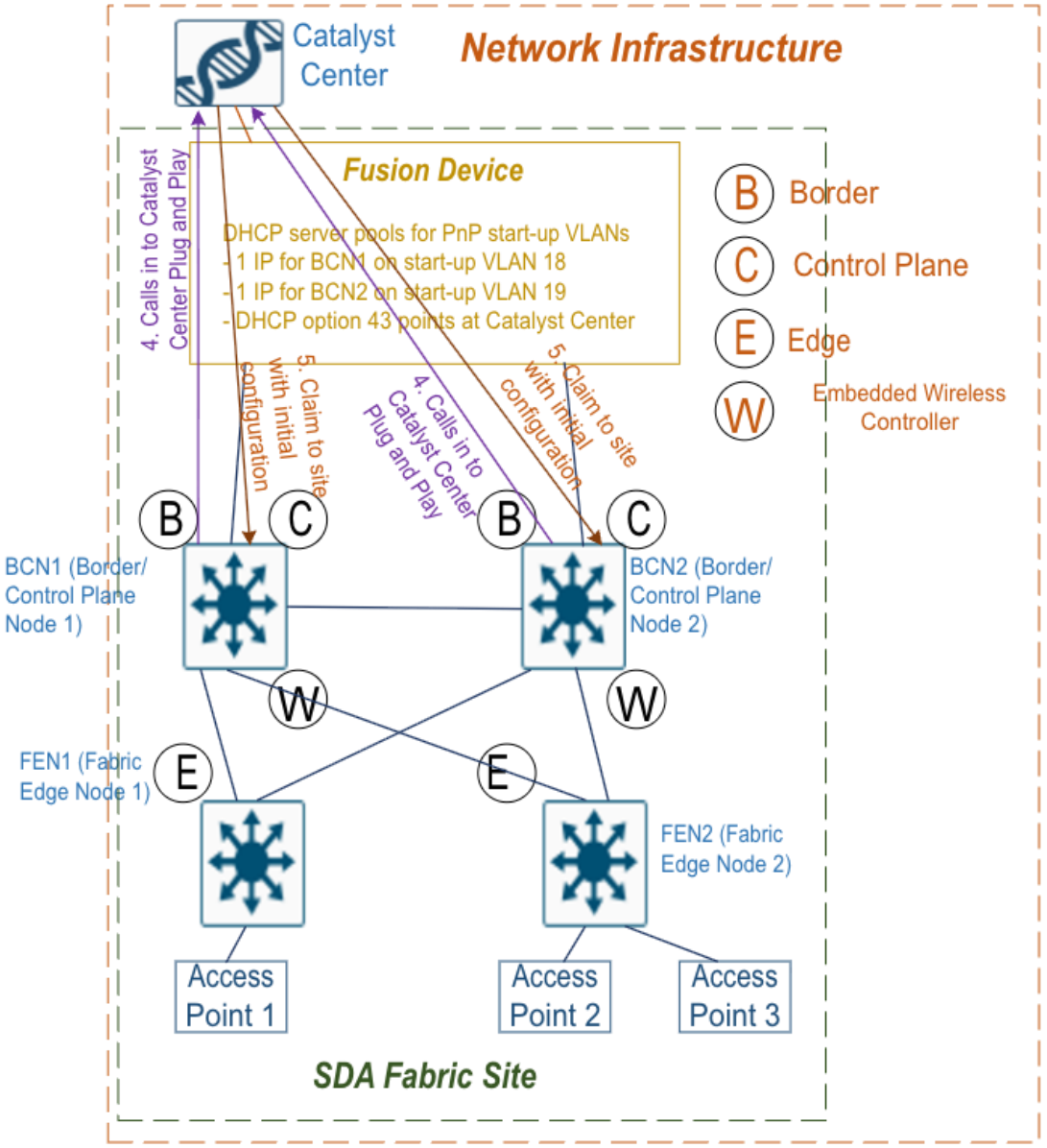
The CLI template is parameterized and used as on on-boarding template. With API call the CLI template needs not be attached to a switching network profile.



Steps in Catalyst Center Basic Automation of the two SDA Border / Control Plane Nodes

1. All Border / Control Plane Nodes and Fabric Edge Nodes and Access Points factory reset
2. Fusion device provides PnP start-up VLAN 18 to BCN1 and PnP start-up VLAN 19 to BCN2
3. BCN1 and BCN2 receives reserved DHCP IP address on their startup VLAN
4. BCN1 and BCN2 call in to Catalyst Center Plug and Play
5. BCN1 and BCN2 get claimed to site with initial configuration from Catalyst Center CLI template

Catalyst Center Basic Automation (CLI Template) (2/2)



Steps in Catalyst Center Basic Automation of the two SDA Border / Control Plane Nodes

1. All Border / Control Plane Nodes and Fabric Edge Nodes and Access Points factory reset
2. Fusion device provides PnP start-up VLAN 18 to BCN1 and PnP start-up VLAN 19 to BCN2
3. BCN1 and BCN2 receives reserved DHCP IP address on their startup VLAN
4. BCN1 and BCN2 call in to Catalyst Center Plug and Play
5. BCN1 and BCN2 get claimed to site with initial configuration from Catalyst Center CLI template

ATEA TECH TOUR

Agenda

- **Introduction:** Cisco Software Defined Access (SDA), and why to automate it
- Prerequisites and Business Outcome
- Automation Functionality
- Automation Design
- **Python Script:** State Event Machine with fixed sequence of states
- Automation Methods
- **Trade-off:** script functionality vs development & maintenance costs, KEEP IT SIMPLE
- Possible Improvements

Trade-off:
Script functionality vs development & maintenance costs

- Script goes through fixed sequence of states, each state performs one action
- Option to verify result of state in real world infrastructure after each state
- If error in a state, you must fix the errors manually, then restart script to re-perform the state in question
  - Fix error in configuration data
  - Fix error in real world data network infrastructure
- Same sequence of states (actions) as when deploying SDA site manually
- Above hits a reasonable balance between script functionality and development and maintenance costs
- KEEP IT SIMPLE
- ***Example***: In Atea lab total time to deploy SDA site with two SDA border / control plane nodes, two Fabric Edge nodes and three Access Points is *1 hour 55 minutes*. Of these *47 minutes* are ”waiting” time in the format of synchronization of switches, and some times giving Catalyst Center time to ”digest” the changes. By adding more API calls to check completion of tasks, and implement parallel synchronization of nodes, this ”waiting time” could be reduced. However it has been chosen not to do this, to keep script simple.

ATEA TECH TOUR

Agenda

- **Introduction:** Cisco Software Defined Access (SDA), and why to automate it
- Prerequisites and Business Outcome
- Automation Functionality
- Automation Design
- **Python Script:** State Event Machine with fixed sequence of states
- Automation Methods
- **Trade-off:** script functionality vs development & maintenance costs, KEEP IT SIMPLE
- Possible Improvements

ATEA TECH TOUR

Possible Improvements

- Implement Web application to enter configuration data per site, and get them pre-validated. Configuration data kept in database, e.g. SQL database.
- Execute script from that Web application, which stores the last state, the script successfully completed for a given site
- Automatically generate host names according to naming standard (today host names are entered manually in YAML file)
- Performance improvements, e.g. by performing synchronization of multiple nodes in parallel

ATEA TECH TOUR

Thank you