Vendor: Cisco

Exam Code: 300-635

Exam Name: Automating Cisco Data Center Networking Solutions (DCNAUTO)

Version: DEMO

QUESTION 1
An engineer uses the `cisco.aci.aci_tenant` Ansible module to create an ACI tenant, but the playbook fails with an SSL certificate verification error (`CERTIFICATE_VERIFY_FAILED`). The test-lab APIC uses a self-signed certificate. Which parameter temporarily resolves this without changing the certificate infrastructure?

A. `validate_certs: false`
B. `use_ssl: false`
C. `use_proxy: false`
D. `timeout: 0`

Answer: A
Explanation:
Most Cisco Ansible collections (`cisco.aci.*` included) use `validate_certs` to control TLS certificate chain validation; setting it to `false` skips self-signed certificate validation (recommended only in lab/test environments). `use_ssl: false` switches to plaintext HTTP rather than skipping validation - a very different behavior and a common distractor. `use_proxy` relates to proxy servers; `timeout` is unrelated to TLS.

QUESTION 2
In a Git-based network configuration workflow, an engineer needs to "safely" undo a bad commit that has already been pushed to the remote and pulled by other collaborators, without disrupting their commit history. Which command should be used?

A. `git reset --hard HEAD~1`
B. `git revert <commit-hash>`
C. `git rebase -i HEAD~1` followed by a force push
D. `git commit --amend`

Answer: B
Explanation:
`git revert` creates a new commit that undoes the changes of the target commit, without rewriting history - safe for a public branch that others have already pulled. `git reset --hard` and rebase+force-push both rewrite history, causing divergence between collaborators' local history and the remote (a core Git collaboration principle: never rewrite history that has been pushed to a shared branch). `git commit --amend` only modifies the most recent, not-yet-pushed commit. This IaC/collaboration-workflow question is often paired with "public vs. private branch" scenarios.

QUESTION 3
Nexus Dashboard Insights (NDI) performs Compliance Analysis on an ACI Fabric and detects a discrepancy between design intent and the actual operational programming of a Contract, generating an Advisory. Which statement most accurately describes this capability?

A. NDI automatically remediates the discrepancy with no human involvement
B. NDI performs passive detection and analysis only - comparing expected vs. actual state and alerting; remediation still requires manual action or execution via APIC/NDO
C. This feature only applies to standalone NX-OS mode, not ACI
D. Compliance Analysis only detects hardware faults, not policy configuration

Answer: B
Explanation:
Nexus Dashboard Insights is an observability and analytics platform; its Compliance/Advisory

function detects and alerts but does not autonomously modify fabric configuration (consistent with separating analysis tooling from change-execution tooling, for operational safety). Option A is the most common misconception - mistaking it for fully automated "AIOps remediation" - but fixes still require human confirmation or a change pushed via APIC/NDO. The Operations domain frequently tests "which tool does what" boundaries.

QUESTION 4
In the following YANG model fragment, what is the key distinction between `container` and `list`?

```
yang
container interfaces {
list interface {
key "name";
leaf name { type string; }
leaf enabled { type boolean; }
}
}
```

A. A container can hold only one child node; a list can hold multiple
B. A container represents a single-instance grouping node; a list represents zero or more uniquely-keyed instances of the same type
C. Containers are used for RPC definitions; lists are used for configuration data
D. The two are semantically identical in YANG, differing only in naming convention

Answer: B
Explanation:
In YANG, `container` is a grouping node with exactly one instance (similar to a struct/namespace); `list` represents a collection of same-typed entries, each uniquely identified by a `key`, and can have zero to many instances (similar to a database table). This distinction is critical when constructing NETCONF/RESTCONF XML/JSON payloads: list entries appear as array elements in JSON, while a container is a single object. This is a high-frequency fundamental in the Network Element Programmability domain.

QUESTION 5
An engineer writes an EEM (Embedded Event Manager) applet that should automatically run a Python diagnostic script and log the result via syslog whenever interface Ethernet1/1 goes down. Which event detector best fits this scenario?

A. `event syslog pattern "LINK-3-UPDOWN.*Ethernet1/1.*down"`
B. `event snmp oid ... get-type exact entry-op eq entry-val 2`
C. `event timer countdown 60`
D. `event none`

Answer: A
Explanation:
While SNMP OID polling (Option B) could theoretically detect interface state, the most direct, low-latency, and idiomatic EEM approach is subscribing to syslog pattern matching (`event syslog pattern`), since interface up/down events naturally generate a standard `%LINK-3-UPDOWN` syslog message that EEM can capture in real time and trigger an action on (e.g., `action cli command` to invoke a Python or Bash-shell script). `event timer` is periodic, not event-driven; `event none` is only for manually-triggered applets. This tests practical judgment in selecting the correct EEM event detector.

QUESTION 6
In ACI, an engineer wants EPGs in two different tenants (Tenant A and Tenant B) to communicate, with the shared Contract provided from Tenant `common`. What is the key configuration required to enable cross-tenant Contract sharing?

A. Move both EPGs into the same VRF
B. Set the Contract's Scope to "Global," and configure the exported/imported Contract reference correctly on the consuming side (e.g., via a Contract Interface / exported Contract object) so provider/consumer relationships resolve across tenants
C. Establish an IPN (Inter-Pod Network) directly between the two tenants
D. Set both tenants' AAA domains to the same Security Domain

Answer: B
Explanation:
The core of cross-tenant Contract sharing in ACI is the Contract's **Scope** (Global, Tenant, or VRF), which determines its visibility, combined with an exported Contract reference object created in the consuming tenant that points back to the provider tenant's Contract - required for the provider/consumer relationship to resolve across tenant boundaries. Simply placing both EPGs in the same VRF (Option A) does not automatically allow cross-tenant policy; Security Domain (Option D) only governs RBAC, not the data-plane policy model. This tests a deeper part of the ACI policy model - Scope and cross-tenant object references.

QUESTION 7
Which of the following best describes the typical Kafka-based streaming telemetry integration architecture for Cisco Nexus Dashboard / ACI?

A. APIC itself acts as a Kafka Broker, and third-party systems connect directly to APIC to consume telemetry
B. ACI Fabric integrates via gRPC/Kafka, pushing events/health scores etc. as a stream to external big-data/SIEM platforms (e.g., via a Kafka Exporter feature in Nexus Dashboard or an external Kafka cluster) for consumption
C. Kafka can only collect syslog from standalone NX-OS devices, not ACI
D. Kafka integration requires configuring a separate Kafka Producer on every leaf switch

Answer: B
Explanation:
In practice, Nexus Dashboard (and services running on it, such as Insights) acts as the aggregation and analytics layer, and can push processed telemetry, anomaly events, and health scores to an external Kafka cluster via a built-in Kafka Exporter feature, for consumption by third-party SIEM/big-data platforms - rather than APIC itself acting as a Broker, and without needing per-leaf Kafka Producer configuration (telemetry collection is centralized, not distributed). This spans the Operations and AI in Automation domains, focusing on who generates data, who aggregates it, and who exports it externally.

QUESTION 8
When deploying ACI resources with Terraform, `terraform plan` shows an existing EPG resource will be "destroyed and re-created" instead of updated in place, even though the engineer only changed the EPG's description field. What is the most likely root cause?

A. The provider version is outdated and doesn't support that field
B. That attribute is marked as `ForceNew` in the Terraform resource schema, so any change to it

triggers resource recreation instead of an in-place update
C. The Terraform state file is corrupted
D. The ACI controller cluster is out of sync

Answer: B
Explanation:
In a Terraform provider's schema, certain attributes are marked `ForceNew: true`, meaning the provider cannot update them in place via the API - it must destroy and recreate the resource (typically for structural attributes involving naming or binding relationships). While it seems counterintuitive that changing a description would require recreation, if the provider implementation incorrectly (or intentionally) flags that field as ForceNew, the plan will show destroy+create. This is a harder question because it requires understanding Terraform's internal provider mechanics, not just ACI knowledge - easily mistaken for an environment/version issue.

QUESTION 9
When using the Cisco ACI Cobra SDK (Python ACI SDK) to manipulate managed objects, which step is the **necessary final action to actually commit changes to APIC**?

A. `moDir.login()`
B. Wrap the modified objects in a `cobra.mit.request.ConfigRequest()` and call `moDir.commit(config_req)`
C. `moDir.query()`
D. `mo.setAttr()`

Answer: B
Explanation:
The typical Cobra SDK workflow is: 1) log in (`login`), 2) build/modify the managed object (MO) tree locally (`setAttr`, creating child objects, etc.), 3) wrap the modified objects into a `ConfigRequest`, and 4) call `moDir.commit()` - only this actually pushes the configuration to APIC. Calling `setAttr()` alone only sets a local "dirty flag" on the in-memory object tree and produces no network traffic; `query()` reads rather than writes. This tests understanding of Cobra's local-object-tree-plus-explicit-commit programming model, a finer point of Network Element Programmability.

QUESTION 10
A data-center team wants to add a "pre-change validation" stage in their CI/CD pipeline for Ansible playbooks that modify network configuration - validating playbook syntax and module parameters without actually pushing configuration. Which command/option should be used?

A. `ansible-playbook site.yml --check`
B. `ansible-playbook site.yml --syntax-check`
C. `ansible-lint site.yml`
D. All of the above are valid at different levels of validation depth and are typically combined

Answer: D
Explanation:
This is a comparison question: `--syntax-check` only parses playbook YAML/syntax and never connects to target hosts; `--check` (Check/Dry-Run Mode) connects to targets and simulates "what would change if this actually ran," though not all network modules fully support check mode; `ansible-lint` is a static code-quality/best-practices linter, catching style issues rather than actual syntax errors or device connectivity problems. Production CI/CD pipelines typically layer all three (Lint → Syntax Check → Check Mode → Apply). DCNAUTO's IaC domain favors questions

about combining similar-sounding options rather than picking one.

QUESTION 11
In an ACI Layer 4-7 Service Graph, to insert a third-party firewall transparently (Go-To/Go-Through mode) into the traffic path between two EPGs, with APIC automatically handling policy-based traffic redirection, which component is responsible for the actual redirect configuration?

A. The Bridge Domain's Unicast Routing setting
B. Policy-Based Redirect (PBR), configured via a Redirect Policy in the Service Graph Template, bound to the Device Package/Device Cluster
C. The Contract's QoS Priority setting
D. The VRF's Route Leaking configuration

Answer: B
Explanation:
The core mechanism for transparently inserting a third-party device into an ACI traffic path with automatic redirection is **Policy-Based Redirect (PBR)**: a Redirect Policy is defined within the Service Graph and bound to specific Device Cluster/Concrete Device interfaces; APIC then programs the corresponding redirect rules on the leaf switches so traffic matching the Contract is steered through the firewall before reaching its destination. Bridge Domain routing, Contract QoS, and VRF route leaking are related but do not perform the redirection function itself. This is a high-difficulty question combining Service Graph and PBR knowledge - an advanced ACI topic.

QUESTION 12
While debugging the Nexus Dashboard Orchestrator (NDO) REST API with Postman or a Python script, an engineer sends a PUT request to update a Schema Template's configuration and receives a 200 OK, but the policy is never actually deployed to the site. What step was most likely missed?

A. Forgetting to set the `Content-Type: application/json` header
B. Forgetting to separately call the "Deploy" API (e.g., `POST /mso/api/v1/execute/schema/{id}`), since a PUT on the Schema only updates the orchestrator's desired-state definition and does not automatically trigger deployment to the site's APIC
C. Forgetting to refresh the API authentication token
D. A heartbeat timeout between NDO and APIC

Answer: B
Explanation:
This is a classic pitfall in NDO API usage: updating a Schema (PUT /schemas/{id}) only modifies the "desired state" definition stored in the orchestrator's database. Actually pushing that state to a specific site's APIC requires a separate call to the **Deploy/Execute** API endpoint (corresponding to clicking "Deploy to Sites" in the GUI). Beginners often assume "update = effective immediately," leading them to chase red herrings like auth or connectivity issues (Options A/C/D). Understanding the separation of "definition (desired state)" from "execution (deployment)" is a general orchestrator design pattern and an important IaC concept.

Thank You for Trying Our Product

Lead2pass Certification Exam Features:

★ More than **99,900** Satisfied Customers Worldwide.

★ Average **99.9%** Success Rate.

★ **Free Update** to match latest and real exam scenarios.

★ **Instant Download** Access! No Setup required.

★ Questions & Answers are downloadable in **PDF** format and **VCE** test engine format.

★ Multi-Platform capabilities - **Windows, Laptop, Mac, Android, iPhone, iPod, iPad**.

★ **100%** Guaranteed Success or **100%** Money Back Guarantee.

★ **Fast**, helpful support **24x7**.

View list of all certification exams: http://www.lead2pass.com/all-products.html

10% Discount Coupon Code: **ASTR14**