ATMIC NETWORKS
YOUR GUIDE TO GLOBAL CERTIFIATION

"Please note that these files may not be up to date. However, the questions will help you understand the exam format and typical question patterns."

www.atmicnetworks .com

Warning: Keep connected with our support team for latest updates

Question: 1

Which process is primarily responsible for managing configuration and state in Junos OS?

A. rpd
B. mgd
C. chassisd
D. jsd

Answer: B
Explanation:

The mgd (management daemon) handles configuration and CLI interactions in Junos OS. It processes user input, applies configuration changes, and communicates with other daemons. Without mgd, configuration and operational commands cannot be executed. It is the foundation of MGD-based automation since automation tools rely on it to push changes. Thus, mgd is central to platform automation.

Question: 2

Which of the following are functions of mgd in automation? (Choose two)

A. Managing Junos routing protocols
B. Handling CLI requests
C. Serving NETCONF requests
D. Collecting interface statistics

Answer: B, C
Explanation:

MGD acts as the main automation entry point for Junos OS. It manages CLI parsing and execution, and it also provides programmatic access via NETCONF. While routing protocols are handled by rpd, and statistics by other daemons, mgd is the bridge between manual configuration and automated workflows. This makes it essential for automation frameworks.

Question: 3

Which transport protocol is typically used when Junos supports NETCONF sessions?

A. UDP
B. TCP over port 830

C. HTTP on port 8080
D. gRPC

Answer: B
Explanation:

NETCONF on Junos typically runs over SSH using TCP port 830. This ensures encrypted, secure communication between clients and Junos devices. Automation tools like Ansible, NAPALM, and PyEZ rely on this channel to push configuration and gather data. UDP and HTTP are not valid for NETCONF, and gRPC is a different telemetry protocol.

Question: 4

Which Junos automation feature allows structured streaming telemetry instead of polling SNMP data?

A. NETCONF
B. gNMI
C. Jinja2 templates
D. JSON-RPC

Answer: B
Explanation:

gNMI (gRPC Network Management Interface) is used for structured telemetry streaming in Junos. Unlike SNMP polling, telemetry sends continuous data streams, reducing overhead and improving real-time visibility. It is implemented over gRPC, providing high-performance monitoring. NETCONF is more suited to configuration management, not continuous telemetry.

Question: 5

Which two telemetry protocols are supported by Junos OS? (Choose two)

A. gRPC
B. SNMP v1
C. gNMI
D. SOAP

Answer: A, C
Explanation:

Junos OS provides telemetry using gRPC and gNMI, enabling efficient, structured streaming of operational data. These protocols improve monitoring and analytics by offering real-time feeds. SNMP v1 is outdated and polling-based, while SOAP is not used for Junos telemetry. Modern automation prefers

gRPC/gNMI for scale and performance.

Question: 6

What is the role of the jsd daemon in Junos OS?

A. Executes CLI commands
B. Provides REST API services
C. Manages routing protocols Handles chassis hardware
D.

Answer: B

Explanation:

The jsd (Junos service daemon) is responsible for providing RESTful APIs (such as REST API Explorer) in Junos OS. This enables developers and automation platforms to interact with Junos devices using HTTP/HTTPS requests and JSON/XML payloads. It extends automation beyond NETCONF by supporting modern web-based interfaces.

Question: 7

Which Junos automation method uses YANG models and operates primarily via gRPC/gNMI?

A. NETCONF
B. SNMP
C. Streaming Telemetry
D. CLI scripting

Answer: C

Explanation:

Streaming telemetry in Junos relies on YANG data models and leverages gRPC/gNMI to provide structured, real-time data. Unlike SNMP or CLI scripts, it does not require polling, reducing load on devices. NETCONF also uses YANG models but focuses on configuration, not telemetry. Thus, telemetry is key for continuous monitoring.

Question: 8

Which two methods are supported by Junos for remote automation access? (Choose two)

A. NETCONF over SSH
B. REST API via jsd
C. FTP CLI access
D. RADIUS

Answer: A, B

Explanation:

Automation in Junos relies on NETCONF (through mgd) and REST APIs (via jsd) for programmatic access. These allow tools and scripts to push configuration, retrieve state, and trigger automation tasks. FTP and RADIUS serve different roles (file transfer and authentication) and are not part of the automation control plane.

Question: 9

Which statement correctly describes MGD-based automation?

A. It is responsible only for hardware management.
B. It provides CLI, NETCONF, and XML/JSON interfaces.
C. It is based on gRPC/gNMI only.
D. It runs inside Routing Engine hardware only.

Answer: B

Explanation:

MGD-based automation allows multiple automation pathways, including CLI scripting, NETCONF RPCs, and JSON/XML APIs. It is the backbone of Junos configuration management. It does not rely solely on telemetry protocols like gRPC. While mgd runs on the routing engine, its role extends to configuration and operational automation.

Question: 10

Which two features differentiate streaming telemetry from SNMP polling? (Choose two)

A. Push-based data model
B. Lower bandwidth utilization
C. Encrypted data transfer using FTP
D. CLI command dependency

Answer: A, B

Explanation:

Telemetry uses a push model, where the device continuously streams updates to collectors, reducing the need for frequent polling. This leads to lower overhead and more real-time monitoring. FTP is unrelated, and CLI-based telemetry does not exist. SNMP polling, in contrast, is inefficient and less scalable.

Question: 11

What is the primary advantage of using gNMI for telemetry in Junos OS?

A. Supports FTP-based file exports
B. Enables real-time configuration rollbacks
C. Provides structured, model-driven data
D. Works without YANG models

Answer: C

Explanation:

gNMI relies on YANG-based models to deliver telemetry in a structured, consistent format. This allows network management tools to parse data reliably across multiple devices. Unlike ad-hoc CLI parsing, model-driven data ensures automation-friendly output. FTP and rollbacks are unrelated to gNMI's telemetry role.

Question: 12

Which Junos automation framework provides Python access for scripting?

A. mgd
B. Junos PyEZ
C. jsd
D. gRPC

Answer: B

Explanation:

Junos PyEZ is a Python automation framework that communicates with Junos devices over NETCONF. It provides developers with high-level APIs to interact with device configuration and operational data. While mgd powers the underlying NETCONF, PyEZ simplifies scripting. jsd provides REST APIs, not Python-native functions.

Question: 13

Which daemon would fail if a Junos device cannot respond to REST API automation requests?

A. mgd
B. rpd
C. jsd
D. dcd

Answer: C

Explanation:

The jsd daemon is responsible for Junos REST API. If jsd fails, automation frameworks that rely on REST calls (such as curl, Postman, or web service clients) cannot communicate with the device. MGD handles CLI/NETCONF, while rpd manages routing, and dcd manages DHCP. Therefore, REST issues point to jsd.

Question: 14

Which two automation frameworks rely on mgd for communication? (Choose two)

A. Ansible with netconf connection
B. Junos PyEZ
C. REST API Explorer
D. Telemetry over gRPC

Answer: A, B

Explanation:

Both Ansible (netconf modules) and PyEZ communicate with Junos devices using NETCONF, which is managed by mgd. REST API Explorer instead uses jsd, and telemetry gRPC does not depend on mgd. Thus, mgd remains the critical automation backend for NETCONF-based tools.

Question: 15

Which API in Junos OS is enabled by default and handled by mgd?

A. NETCONF
B. REST API
C. gRPC/gNMI
D. JSON-RPC

Answer: A

Explanation:

NETCONF is enabled by default on Junos and is handled by the mgd daemon. It provides structured XML/JSON-based automation capabilities. REST API via jsd must be explicitly enabled, and telemetry protocols like gRPC/gNMI require additional configuration. Therefore, NETCONF is the out-of-the-box automation interface.

Question: 16

Which Junos automation interface provides a browser-based interactive API for testing?

A. NETCONF over SSH
B. REST API Explorer
C. gRPC telemetry
D. CLI event-scripts

Answer: B

Explanation:

The REST API Explorer, powered by jsd, provides an interactive browser-based GUI for exploring Junos REST API calls. It allows users to test queries and view JSON/XML outputs. NETCONF and telemetry lack browser explorers, and event-scripts are CLI-based. This makes REST API Explorer very developer-friendly.

Question: 17

Which two are benefits of using model-driven telemetry in Junos? (Choose two)

A. Data consistency across vendors
B. Lower frequency of updates
C. Requires CLI parsing for data extraction
D. High scalability and real-time monitoring

Answer: A, D

Explanation:

Model-driven telemetry uses YANG data models, ensuring consistency across vendors. It is highly scalable and provides real-time monitoring, unlike traditional polling. It does not reduce frequency of updates; instead, it increases timeliness. CLI parsing is no longer required since structured data is used.

Question: 18

Which Junos feature enables automation tools to retrieve configuration and operational data in XML or JSON formats?

A. NETCONF
B. gRPC
C. CLI scripts
D. Syslog

Answer: A

Explanation:

NETCONF in Junos supports both XML and JSON output formats, making it flexible for integration with automation frameworks. This enables structured, machine-readable communication instead of screen-scraping CLI outputs. gRPC is used for telemetry, CLI scripts are less structured, and syslog is logging-only.

Question: 19

When using mgd-based automation, which two formats can be returned by NETCONF RPC responses? (Choose two)

A. XML
B. YAML
C. JSON CSV
D.

Answer: A, C

Explanation:

MGD-based automation in Junos supports XML and JSON outputs when using NETCONF RPCs. These formats allow structured parsing by automation platforms. YAML and CSV are not supported by default in mgd/NETCONF. Having XML/JSON is crucial for interoperability with tools like PyEZ, Ansible, and NAPALM.

Question: 20

Which daemon in Junos is directly responsible for providing programmatic REST interfaces for configuration and operational tasks?

A. mgd
B. jsd
C. snmpd
D. rpd

Answer: B

Explanation:

The jsd daemon delivers REST API services in Junos, allowing interaction through HTTP/HTTPS with JSON or XML payloads. It complements mgd (NETCONF/CLI) and enables modern web-driven automation. snmpd handles SNMP and rpd manages routing protocols, making jsd the correct answer for REST API services.

Question: 21

Which transport protocol is most commonly used for NETCONF sessions on Junos devices?

A. UDP
B. TCP over port 22 (SSH)
C. HTTP over port 8080
D. gRPC

Answer: B

Explanation:

NETCONF sessions typically run over SSH on TCP port 22 or port 830, ensuring secure communication

between clients and Junos devices. This provides encryption and authentication for automation workflows. UDP and HTTP are not valid transports for NETCONF, while gRPC belongs to streaming telemetry, not NETCONF.

Question: 22

Which of the following is a benefit of NETCONF compared to CLI automation?

A. Requires CLI screen scraping
B. Model-driven configuration exchange
C. Uses only ASCII text outputs
D. Lacks structured data representation

Answer: B
Explanation:

NETCONF provides a model-driven approach for configuration and state management, using XML or JSON formats. Unlike CLI automation, which depends on screen scraping, NETCONF ensures structured and consistent data representation. This makes parsing easier for automation tools, improving reliability and efficiency.

Question: 23

Which format is the default encoding for Junos NETCONF RPCs?

A. YAML
B. XML
C. JSON
D. CSV

Answer: B
Explanation:

NETCONF in Junos traditionally uses XML encoding for its RPCs. While Junos also supports JSON as an alternative output, XML remains the standard default. YAML and CSV are not supported formats for NETCONF RPC exchanges. XML ensures hierarchical and structured communication aligned with YANG models.

Question: 24

Which two Junos automation access methods rely on the mgd daemon? (Choose two)

A. NETCONF

B. REST API
C. XML API

D. gRPC Telemetry

Answer: A, C

Explanation:

The mgd daemon powers NETCONF and XML API access methods in Junos. These allow structured automation and configuration changes. REST API relies on jsd, and gRPC telemetry is handled separately. Thus, mgd is the central process supporting traditional automation APIs.

Question: 25

What is a key difference between REST API and NETCONF in Junos automation?

A. REST API uses XML exclusively
B. NETCONF supports YANG-based models
C. REST API requires mgd daemon
D. NETCONF only works via HTTP

Answer: B

Explanation:

NETCONF is YANG-model based, providing structured, standardized configuration management. REST API, however, exchanges data through JSON or XML over HTTP/HTTPS. REST is designed for web developers, while NETCONF is designed for network automation. mgd supports NETCONF/XML, while REST requires jsd, not mgd.

Question: 26

Which of the following is true for XML API on Junos?

A. It is based on REST principles
B. It is a predecessor to NETCONF
C. It is used only for telemetry
D. It does not use RPCs

Answer: B

Explanation:

The XML API is a predecessor of NETCONF, providing access to configuration and operational data in XML format. NETCONF later standardized the use of XML/YANG with structured RPCs. XML API still exists but has been largely replaced by NETCONF and REST API for modern automation.

Question: 27

Which layer of the Junos OS handles NETCONF RPC parsing and execution?

A. jsd
B. rpd
C. mgd
D. chassisd

Answer: C
Explanation:

The mgd (management daemon) processes NETCONF RPCs, parses XML input, and applies configuration or retrieves operational data. rpd handles routing protocols, jsd handles REST API, and chassisd manages hardware. Therefore, mgd is the correct layer responsible for NETCONF automation.

Question: 28

Which two data formats are commonly supported by Junos REST API? (Choose two)

A. XML
B. JSON
C. YAML
D. INI

Answer: A, B
Explanation:

REST API on Junos supports XML and JSON, making it compatible with a wide variety of automation tools and developers' preferences. YAML and INI are not supported formats for REST API exchanges. REST API flexibility allows integration with modern DevOps tools and cloud systems.

Question: 29

Which Junos automation method is the best choice when integrating with cloud-native or web-based applications?

A. NETCONF
B. XML API
C. REST API
D. SNMP

Answer: C
Explanation:

The REST API is most suitable for cloud-native or web applications because it uses HTTP/HTTPS with JSON/XML payloads, making it easy to integrate with microservices and DevOps tools. NETCONF and XML API are more network-focused, while SNMP is limited to monitoring.

Question: 30

Which statement correctly describes the XML API in Junos OS?

A. It provides JSON output only
B. It allows structured requests using XML RPCs
C. It is based on gRPC protocol
D. It is deprecated and cannot be used

Answer: B

Explanation:

The XML API allows developers to interact with Junos using XML-based RPCs for both configuration and operational commands. While NETCONF later expanded on this, XML API is still functional. It supports XML, not JSON, and is not based on gRPC. Though older, it remains useful in some environments.

Question: 31

Which transport is most often used by Junos REST API?

A. SSH
B. HTTP/HTTPS
C. gRPC
D. Telnet

Answer: B

Explanation:

REST API is built on HTTP/HTTPS transport, making it highly compatible with modern web applications. Unlike NETCONF, which typically uses SSH, REST API aligns with standard web protocols. This ensures ease of use for developers familiar with HTTP clients like curl or Postman.

Question: 32

Which two features differentiate NETCONF from XML API? (Choose two)

A. NETCONF is standardized by IETF
B. XML API supports YANG models
C. NETCONF offers transaction support with locks
D. XML API supports gRPC streaming

Answer: A, C

Explanation:

NETCONF is an IETF-standardized protocol that provides additional features like configuration locks and transactions, ensuring consistency. XML API lacks these advanced mechanisms. XML API is not YANG-driven and does not support telemetry. NETCONF improves reliability and interoperability compared to XML API.

Question: 33

Which of the following is a benefit of using REST API for Junos automation?

A. Requires no authentication
B. Platform-neutral and easy for developers
C. Limited to XML-only format
D. Replaces gRPC telemetry

Answer: B
Explanation:

REST API is platform-neutral, using HTTP and JSON/XML, making it easier for developers familiar with web services. Authentication is required (username/password or token). REST API is flexible, but it does not replace telemetry protocols like gRPC. Its ease of integration is its key strength.

Question: 34

Which two automation methods provide programmatic configuration changes in Junos OS? (Choose two)

A. REST API
B. NETCONF
C. Syslog
D. SNMP

Answer: A, B
Explanation:

Both REST API and NETCONF allow structured configuration changes on Junos devices. Syslog and SNMP are primarily for monitoring and alerts, not for configuration. This makes REST and NETCONF essential for programmatic and automated workflows in DevOps environments.

Question: 35

Which Junos automation method is most widely used with Ansible and PyEZ?

A. REST API
B. gRPC Telemetry
C. NETCONF
D. XML API

Answer: C
Explanation:

Automation frameworks like Ansible (network modules) and PyEZ rely heavily on NETCONF for communication with Junos devices. REST API can also be used, but Ansible and PyEZ primarily depend on mgd/NETCONF. XML API is older and not the main choice for these frameworks.

Question: 36

Which two response formats can be requested via Junos REST API? (Choose two)

A. JSON
B. XML
C. YAML
D. HTML

Answer: A, B
Explanation:

Junos REST API supports both JSON and XML outputs, providing flexibility for developers and automation scripts. YAML and HTML are not directly supported for REST API calls. This makes REST API suitable for integration with CI/CD pipelines and analytics platforms.

Question: 37

Which Junos automation interface allows developers to test API calls through a browser-based GUI?

A. NETCONF
C. REST API Explorer
C. XML API
D. gRPC

Answer: B
Explanation:

The REST API Explorer, enabled through jsd, allows developers to test API calls interactively via a web browser. It displays responses in JSON/XML formats. NETCONF and XML API are CLI or programmatic only, while gRPC is for telemetry. REST Explorer makes it developer-friendly.

Question: 38

Which statement describes the role of YANG models in NETCONF?

A. They define telemetry streaming only B. They define device data structures C. They replace JSON schemas in REST D. They are optional and rarely used

Answer: B

Explanation:

YANG models define device configuration and operational data structures for NETCONF. They ensure data is machine-readable and consistent across vendors. While also used in telemetry (gNMI), they are not limited to streaming. YANG is central to structured automation workflows in Junos.

Question: 39

Which two automation access methods in Junos provide transaction-based configuration management? (Choose two)

A. NETCONF
B. REST API
C. XML API
D. CLI

Answer: A, D

Explanation:

Both NETCONF and CLI support transaction-based configuration with commit and rollback options. REST API lacks full transaction locking, and XML API is more limited. NETCONF enhances this by providing edit-config, lock, and rollback RPCs, ensuring atomic and safe configuration.

Question: 40

Which API is recommended when building DevOps pipelines that integrate Junos devices with cloud CI/CD systems?

A. XML API
B. NETCONF
C. REST API
D. SNMP

Answer: C

Explanation:

For DevOps pipelines, the REST API is recommended because it integrates easily with CI/CD tools and cloud-native applications. It uses familiar web technologies (HTTP, JSON, XML), unlike NETCONF/XML API, which are more specialized. SNMP is limited to monitoring, not CI/CD integration.

Question: 41

Which of the following is a key characteristic of YAML?

A. Uses braces and brackets like JSON
B. Uses indentation to represent hierarchy
C. Supports only XML-based structures
D. Requires schema definitions

Answer: B
Explanation:

YAML represents data hierarchically using indentation instead of brackets or braces. This makes it human-readable and widely used in automation tools like Ansible. Unlike XML or JSON, YAML avoids heavy syntax but still maintains structure. Schemas are not required by default, making it lightweight.

Question: 42

Which two serialization formats are most commonly used in Junos automation? (Choose two)

A. XML
B. YAML
C. INI
D. JSON

Answer: A, D
Explanation:

Junos automation primarily relies on XML (for NETCONF) and JSON (for REST API and telemetry). YAML is widely used in tools like Ansible, but Junos native APIs output mostly XML/JSON. INI files are outdated and rarely used in automation pipelines. XML and JSON remain central in Junos integration.

Question: 43

Which serialization format is most compact and web-friendly for API responses?

A. YAML
B. XML

C. JSON
D. CSV

Answer: C
Explanation:

JSON is lightweight, compact, and directly supported in most programming languages and web APIs. It is more concise than XML and more structured than CSV. YAML is readable but less efficient for APIs. Therefore, JSON is the preferred choice for Junos REST API and telemetry responses.

Question: 44

Which two are advantages of YAML in automation? (Choose two)

A. More human-readable than JSON
B. Requires strict indentation rules
C. Supports comments in configuration files
D. Is the default format for NETCONF

Answer: A, C

Explanation:

YAML is human-friendly and allows comments, which JSON does not natively support. Its indentation structure makes automation playbooks (like Ansible) easy to maintain. However, it is not the default format for Junos APIs (which use XML/JSON). YAML is best suited for configuration templating and orchestration.

Question: 45

Which templating engine is commonly used with Python and Ansible to generate configuration files dynamically?

A. Jinja2
B. JSON
C. XML API
D. YAML

Answer: A

Explanation:

Jinja2 is a powerful templating engine widely used in Python and automation tools like Ansible. It allows inserting variables, loops, and conditionals into configuration templates. YAML, JSON, and XML are data serialization formats, not templating engines. Jinja2 bridges the gap between data and templates.

Question: 46

Which data format is self-descriptive and schema-based?

A. JSON
B. YAML
C. XML
D. CSV

Answer: C

Explanation:

XML is self-descriptive and supports schemas (XSD, DTD), making it suitable for structured and validated data exchange. JSON is lightweight but lacks built-in schema enforcement. YAML is more human-friendly but less rigid. CSV is flat and unstructured. XML's extensibility makes it powerful for Junos automation history.

Question: 47

Which two serialization formats are typically used for REST API communication in Junos? (Choose two)

A. JSON
B. XML
C. YAML
D. INI

Answer: A, B

Explanation:

Junos REST APIs support JSON and XML outputs. These formats are machine-readable and integrate easily with automation tools. YAML is not used in REST APIs but is used in playbooks. INI is legacy and not supported in Junos APIs. JSON is preferred for speed, XML for structure.

Question: 48

Which of the following is a benefit of JSON over XML?

A. Supports comments
B. Less verbose and faster to parse
C. Requires schemas for parsing
D. Only supported in Junos PyEZ

Answer: B

Explanation:

JSON is less verbose than XML and much faster to parse, making it ideal for APIs and telemetry. It does not require schemas to function, though optional schemas exist (JSON Schema). JSON does not natively support comments. Junos PyEZ supports both JSON and XML, not exclusively JSON.

Question: 49

Which two features are supported by Jinja2 templates? (Choose two)

A. Variable substitution
B. Loops and conditionals
C. Hierarchical indentation rules
D. Data serialization in XML

Answer: A, B

Explanation:

Jinja2 templates allow variable substitution and control structures like loops and conditionals. This makes it ideal for dynamic configuration file generation. YAML's indentation rules and XML serialization are unrelated to Jinja2. Instead, Jinja2 works with data sources (YAML, JSON) to build templates.

Question: 50

Which format is most suited for human readability in configuration playbooks like Ansible?

A. JSON
B. YAML
C. XML
D. CSV

Answer: B

Explanation:

YAML is considered the most human-readable serialization format due to its indentation-based structure and minimal syntax. JSON is machine-friendly but harder for humans to scan in large playbooks. XML is verbose, and CSV lacks hierarchy. YAML's clarity makes it the default choice for Ansible.

Question: 51

Which serialization format is typically used in NETCONF RPCs?

A. JSON
B. YAML
C. XML
D. CSV

Answer: C

Explanation:

NETCONF uses XML encoding for its RPC requests and responses. This allows hierarchical representation of configuration and state. JSON is supported in REST APIs and telemetry, YAML in orchestration tools, and CSV in tabular exports. XML remains the foundation of NETCONF in Junos automation.

Question: 52

Which two formats are commonly used with Jinja2 templates in automation pipelines? (Choose two)

A. YAML
B. JSON

C. XML
D. INI

Answer: A, B

Explanation:

YAML and JSON are commonly used as input data sources for Jinja2 templates. These provide structured variables that Jinja2 injects into dynamic templates. XML can also be transformed but is less common. INI is not widely used in modern templating. YAML and JSON are best aligned with DevOps tools.

Question: 53

Which statement correctly describes the role of Jinja2 in automation?

A. It is a data serialization format like YAML
B. It generates configuration templates dynamically
C. It streams telemetry to collectors
D. It replaces NETCONF in Junos

Answer: B

Explanation:

Jinja2 is a templating engine that dynamically generates configuration files by combining static templates with variable data. It does not serialize data like YAML or JSON, nor replace NETCONF. Instead, it complements automation by creating reusable, parameterized templates for Junos and multi-vendor environments.

Question: 54

Which two serialization formats allow hierarchical data representation? (Choose two)

A. YAML
B. JSON
C. CSV
D. INI

Answer: A, B

Explanation:

Both YAML and JSON support hierarchical (nested) structures. This makes them ideal for representing complex configurations in automation. CSV and INI are flat, lacking nested capabilities. This hierarchy is crucial for cloud, networking, and DevOps workflows that require detailed multi-level data representation.

Question: 55

Which serialization format is the most verbose and schema-driven?

A. XML
B. JSON
C. YAML
D. CSV

Answer: A
Explanation:

XML is highly verbose due to its tag-based syntax but provides schema validation, namespaces, and extensibility. JSON is lighter, YAML is more human-friendly, and CSV is flat. XML's verbosity makes it less efficient but ensures strong structure and validation when required.

Question: 56

Which two formats are widely supported in Junos telemetry outputs? (Choose two)

A. JSON
B. XML
C. YAML
D. CSV

Answer: A, B
Explanation:

Junos telemetry supports JSON and XML outputs, aligning with industry-standard collectors and analytics platforms. YAML is used in orchestration, not telemetry, while CSV is unsuitable for structured streaming. JSON is usually preferred for its performance, while XML is used for compatibility.

Question: 57

Which Jinja2 feature allows the use of conditions to build different parts of a configuration?

A. Filters
B. Control structures
C. Indentation
D. Serialization

Answer: B
Explanation:

Jinja2 supports control structures such as if, else, and loops, enabling conditional logic in templates. This allows flexible generation of configurations depending on variables. Filters are also supported but serve to modify values, not structure. Indentation and serialization belong to YAML/XML.

Question: 58

Which two serialization formats are most interoperable across vendors in network automation? (Choose two)

A. JSON
B. YAML
C. XML
D. INI

Answer: A, C
Explanation:

JSON and XML are widely used and supported across vendors and tools, making them highly interoperable in multi-vendor automation. YAML is common in orchestration (e.g., Ansible) but less standardized across vendors. INI is outdated and rarely used in networking automation.

Question: 59

Which serialization format does not natively support comments?

A. YAML
B. JSON
C. XML
D. INI

Answer: B
Explanation:

JSON does not natively support comments, which limits its readability for humans. YAML and XML both allow comments, making them easier to document. INI also allows inline comments. JSON remains machine-friendly but less human-friendly for long configuration files.

Question: 60

Which of the following best describes templating with Jinja2 in Junos automation?

A. Data is serialized into JSON before sending to the device
B. Templates are created once and cannot be parameterized
C. Templates use variables and logic to create device configs dynamically
D. Jinja2 directly replaces YAML in automation pipelines

Answer: C
Explanation:

With Jinja2 templating, templates can contain variables, loops, and conditionals, enabling dynamic generation of device configurations. This allows reusable, parameterized templates for different devices or environments. Jinja2 does not replace YAML but works alongside it. This combination makes automation scalable and flexible.

Question: 61

Which Python library is designed by Juniper for automating Junos devices?

A. Paramiko
B. Netmiko
C. PyEZ
D. NAPALM

Answer: C
Explanation:

Junos PyEZ is Juniper's native Python library for automating configuration and operational tasks. It connects to devices using NETCONF and provides easy-to-use APIs. Paramiko and Netmiko are generic SSH-based libraries, and NAPALM is a multi-vendor abstraction. PyEZ is most deeply integrated with Junos.

Question: 62

Which two transport methods are supported by PyEZ for device connectivity? (Choose two)

A. NETCONF over SSH
B. Telnet
C. Serial Console
D. Local shell (for on-box scripts)

Answer: A, D
Explanation:

PyEZ primarily uses NETCONF over SSH for remote connectivity. However, when running directly on a Junos device (on-box scripting), it can operate using the local shell transport. Telnet is not supported due to security concerns, and serial console is not typical in PyEZ workflows.

Question: 63

Which Python object in PyEZ is used to establish a connection to a Junos device?

A. Device()
B. Connect()
C. Session()

D. Client()

Answer: A
Explanation:

The Device() object from the jnpr.junos module is used to open and manage a session to a Junos device. It handles authentication, connectivity, and context for all further operations. Without initializing Device(), no PyEZ RPCs or configuration tasks can be executed.

Question: 64

Which Python exception is raised when a PyEZ session fails due to authentication issues?

A. ConnectTimeoutError
C. LoginError
C. ConnectAuthError
D. ValueError

Answer: C
Explanation:

If PyEZ fails authentication (e.g., wrong username or password), it raises a ConnectAuthError. This allows developers to handle failed logins gracefully in scripts. ConnectTimeoutError occurs if the device is unreachable, while ValueError is unrelated. Proper exception handling ensures robust automation scripts.

Question: 65

Which two features of PyEZ are used for retrieving operational data? (Choose two)

A. Tables and Views
B. Config.load()
C. RPC calls
D. CLI scripting

Answer: A, C
Explanation:

PyEZ allows retrieving operational data via Tables/Views abstraction and direct RPC calls. Config.load() is for configuration management, not operational data. CLI scripting is possible, but PyEZ prefers structured APIs over screen-scraping. Tables and RPCs are the recommended approach for automation workflows.

Question: 66

Which method in PyEZ is used to apply configuration changes to a Junos device?

A. Config.push()
B. Config.commit()
C. Config.load()
D. Config.apply()

Answer: C
Explanation:

The Config.load() method is used to load configuration changes into the candidate database. Developers can then use Config.commit() to apply them. This mirrors the Junos CLI commit model. Config.push() and Config.apply() do not exist in PyEZ, making Config.load() the correct choice.

Question: 67

Which two are benefits of using RPCs in PyEZ? (Choose two)

A. Direct access to Junos internal APIs
B. Uses YANG-modeled structured output
C. Always requires CLI screen scraping
D. Avoids dependency on NETCONF

Answer: A, B
Explanation:

PyEZ RPCs provide direct access to Junos operational and configuration APIs, returning structured data based on YANG/XML models. This eliminates the need for fragile screen scraping. Since RPCs are executed over NETCONF, option D is incorrect. RPCs provide reliable, model-driven automation.

Question: 68

Which Python module in PyEZ handles rollback of configurations?

A. jnpr.junos.utils.sw
B. jnpr.junos.utils.fs
C. jnpr.junos.utils.config
D. jnpr.junos.rpc

Answer: C
Explanation:

The Config utility in PyEZ (jnpr.junos.utils.config) allows for rollback, commit, and configuration management. The sw utility handles software upgrades, while fs handles file system operations. The rpc module handles raw RPC calls. Config utility is central for Junos configuration automation.

Question: 69

What is the default output format for PyEZ RPC responses?

A. YAML
B. XML
C. JSON
D. Plain text

Answer: B

Explanation:

By default, PyEZ RPC responses are returned in XML format. However, developers can convert them into JSON or Python dicts for easier handling. YAML and plain text are not default formats. XML provides the full structured representation from the Junos device.

Question: 70

Which two steps are necessary before committing configuration via PyEZ? (Choose two)

A. Open a Device() connection
B. Use Config.load() to stage changes
C. Execute rpc.get-chassis-inventory()
D. Start Jinja2 templating

Answer: A, B

Explanation:

To commit changes in PyEZ, a developer must first open a device session and then load changes into the candidate config using Config.load(). RPC calls like get-chassis-inventory() are unrelated, and Jinja2 is optional for templating but not required before commit.

Question: 71

Which PyEZ utility is used to handle software upgrades on Junos devices?

A. Config
B. Sw
C. Rpc
D. Tables

Answer: B

Explanation:

The Sw utility (jnpr.junos.utils.sw) is used for software upgrades in PyEZ. It supports installing, verifying,

and rebooting after upgrades. Config handles configuration, Rpc retrieves operational data, and Tables/Views abstract structured outputs. Sw is dedicated to upgrade automation.

Question: 72

Which two exception classes are important in PyEZ error handling? (Choose two)

A. ConnectTimeoutError
B. ConnectAuthError
C. RPCError
D. JSONDecodeError

Answer: A, B

Explanation:

ConnectTimeoutError and ConnectAuthError are critical for handling PyEZ connection issues. RPCError also exists but is specific to failed RPCs, not connectivity. JSONDecodeError belongs to Python's standard library, not PyEZ. Exception handling ensures robust scripts that fail gracefully.

Question: 73

Which PyEZ call retrieves the routing table information?

A. dev.rpc.get_chassis_inventory()
B. dev.rpc.get_route_information()
C. dev.rpc.get_interface_information()

D. dev.rpc.get_config()

Answer: B

Explanation:

The get-route-information RPC retrieves the device's routing table. RPCs map directly to Junos operational commands. Chassis inventory provides hardware data, interface-information shows interface status, and get_config retrieves configurations. RPC mapping is one of PyEZ's most powerful features.

Question: 74

Which two programming features are supported by PyEZ for automation flexibility? (Choose two)

A. Tables/Views abstraction
B. CLI screen scraping
C. Direct XML RPC calls
D. Proprietary binary data formats

Answer: A, C

Explanation:

PyEZ supports Tables/Views abstraction for structured data access and direct XML RPC calls for advanced operations. CLI screen scraping is avoided due to fragility, and proprietary formats are not used. These two features provide balance between simplicity and full device control.

Question: 75

Which Python command ensures a PyEZ device connection is properly closed?

A. dev.shutdown()
B. dev.disconnect()
C. dev.close()
D. dev.exit()

Answer: C

Explanation:

The dev.close() method properly terminates a PyEZ session with a Junos device. This ensures that NETCONF sessions are cleaned up and resources released. Shutdown() and exit() do not exist in PyEZ, while disconnect() is not valid. Closing sessions is important in large-scale automation.

Question: 76

Which two PyEZ utilities are directly responsible for configuration changes? (Choose two)

A. Config
B. Sw
C. Rpc
D. Tables

Answer: A, C

Explanation:

The Config utility handles configuration staging, commits, and rollbacks, while Rpc can also be used to make direct configuration-related RPC calls. Sw handles software upgrades, and Tables are primarily for operational data abstraction. Config and Rpc are central to config automation.

Question: 77

Which PyEZ feature allows mapping Junos XML to Python classes for structured access?

A. Sw
B. Tables and Views
C. Jinja2

D. Config

Answer: B
Explanation:

Tables and Views allow developers to map Junos XML output into structured Python classes. This simplifies accessing nested data without manually parsing XML. Sw and Config are utilities for upgrades and configs, while Jinja2 is unrelated. Tables/Views improve data handling efficiency.

Question: 78

Which two benefits are achieved by PyEZ exception handling? (Choose two)

A. Prevents automation script crashes
B. Allows retry logic on errors
C. Eliminates need for RPC calls
D. Removes mgd dependencies

Answer: A, B
Explanation:

Exception handling in PyEZ ensures automation scripts don't crash unexpectedly and supports retry mechanisms for resilience. It does not eliminate RPC usage or mgd dependencies. Proper error handling improves reliability, especially in large-scale or CI/CD environments.

Question: 79

Which PyEZ RPC retrieves interface statistics?

A. dev.rpc.get_interface_information()
B. dev.rpc.get_config()
C. dev.rpc.get_chassis_inventory()
D. dev.rpc.get_route_information()

Answer: A
Explanation:

The get-interface-information RPC retrieves operational interface statistics such as bandwidth, errors, and status. get_config retrieves configuration, chassis-inventory shows hardware, and route-information displays routing details. RPC mapping enables structured and programmatic data retrieval.

Question: 80

Which two PyEZ features simplify writing human-readable code for operational data retrieval? (Choose two)

A. Tables/Views
B. Filters in Jinja2
C. Direct NETCONF edits
D. JSON exports

Answer: A, D

Explanation:

Tables/Views provide abstraction for human-readable code, while exporting RPC results as JSON makes parsing easy in Python. Jinja2 is for templating, and NETCONF edits are config-focused. These two features improve developer experience when retrieving and processing operational data.

Question: 81

Which daemon powers the REST API on Junos devices?

A. mgd
B. rpd
C. jsd
D. snmpd

Answer: C

Explanation:

The jsd daemon provides REST API services in Junos, enabling HTTP/HTTPS communication with JSON/XML payloads. mgd handles NETCONF/XML API, rpd manages routing, and snmpd manages SNMP. REST automation workflows depend on jsd for modern API interactions.

Question: 82

Which two data formats are supported in Junos REST API responses? (Choose two)

A. JSON
B. XML
C. YAML
D. INI

Answer: A, B

Explanation:

REST API in Junos supports JSON and XML, making it flexible for integration with cloud-native and DevOps platforms. YAML and INI are not supported in REST responses. JSON is often preferred for its simplicity, while XML maintains compatibility with older tools.

Question: 83

Which Python-based tool validates Junos configurations and states using predefined test cases?

A. PyEZ
B. JSNAPy
C. Ansible
D. Jinja2

Answer: B
Explanation:

JSNAPy is a Python testing framework developed by Juniper to verify Junos device configuration and operational states. It compares snapshots against predefined test cases to ensure compliance and detect issues. Unlike PyEZ, JSNAPy is focused on validation and verification, not device configuration.

Question: 84

Which two tasks are commonly performed with JSNAPy? (Choose two)

A. Snapshot device states
B. Validate operational outputs
C. Commit device configurations
D. Manage Python exceptions

Answer: A, B
Explanation:

JSNAPy can take snapshots of configuration/operational data and validate outputs against test cases. It does not handle committing configurations (PyEZ does that) or Python exceptions. Its strength lies in ensuring consistency and correctness across deployments.

Question: 85

Which serialization formats are most often used in JSNAPy test files?

A. YAML and JSON
B. XML and CSV
C. JSON and INI
D. XML and YAML

Answer: A
Explanation:

JSNAPy test cases are defined in YAML or JSON, providing structured and human-readable definitions. XML and CSV are not used for test definitions, though XML may appear in device responses. YAML's

readability makes it a popular choice for writing test playbooks in JSNAPy.

Question: 86

Which two benefits does REST API provide over NETCONF in Junos? (Choose two)

A. Easier integration with web/cloud apps
B. Human-friendly YAML outputs
C. Works natively with HTTP/HTTPS
D. Supports rollback transactions

Answer: A, C
Explanation:

REST API works with HTTP/HTTPS and is easier to integrate with web/cloud applications than NETCONF. However, rollback transactions are a NETCONF feature, not REST. REST supports JSON/XML but not YAML natively. REST excels in cloud-native DevOps environments.

Question: 87

Which Jinja2 feature enables dynamic looping through variables in templates?

A. Control structures
B. Filters
C. Indentation
D. Serializers

Answer: A
Explanation:

Jinja2 supports control structures like for loops and if conditionals, allowing templates to dynamically generate configurations. Filters modify variable values, indentation belongs to YAML, and serializers refer to data conversion. Control structures enable flexible template logic.

Question: 88

Which two use cases best fit JSNAPy? (Choose two)

A. Pre- and post-change validation
B. Regression testing in CI/CD pipelines
C. Automated configuration deployment
D. Python templating for configs

Answer: A, B
Explanation:

JSNAPy excels in validation use cases such as pre/post-change checks and regression testing in CI/CD. It does not deploy configs (that's PyEZ/Ansible) or template configs (that's Jinja2). Its focus is on detecting deviations from expected device states.

Question: 89

Which REST API interface allows interactive testing of Junos API calls via a browser?

A. Swagger UI
B. REST API Explorer
C. Ansible UI
D. PyEZ Views

Answer: B

Explanation:

The REST API Explorer allows testing Junos REST calls interactively via a browser. It is built into Junos and powered by jsd. Swagger UI is a generic tool but not Junos-specific. Ansible UI and PyEZ Views are unrelated. REST API Explorer improves developer productivity.

Question: 90

Which two Jinja2 features enable customization of templates? (Choose two)

A. Variable substitution
B. Filters
C. Schema validation
D. RPC abstraction

Answer: A, B

Explanation:

Jinja2 templates support variable substitution and filters for transforming data. Schema validation is related to XML/JSON, and RPC abstraction belongs to PyEZ. Jinja2 provides flexibility in generating device configs dynamically from YAML/JSON variables.

Question: 91

Which JSNAPy feature allows defining pass/fail conditions in test cases?

A. Validators
B. Filters
C. Serializers
D. Control structures

Answer: A
Explanation:

In JSNAPy, validators define the pass/fail logic for test cases. They specify conditions like “interface must be up” or “routes must exist.” Filters and control structures are Jinja2 concepts, while serializers handle data formats. Validators enforce compliance checks.

Question: 92

Which two data formats can be used as input to Jinja2 templates? (Choose two)

A. JSON
B. YAML
C. XML
D. CSV

Answer: A, B
Explanation:

Jinja2 templates often consume JSON and YAML data as variable sources. These formats are structured and human-readable. XML is less common, though it can be transformed, and CSV lacks hierarchy. JSON and YAML are best for variable-driven templating in Junos automation.

Question: 93

Which two Junos automation methods rely on the jsd daemon? (Choose two)

A. REST API
B. REST API Explorer
C. PyEZ RPCs
D. JSNAPy

Answer: A, B
Explanation:

The jsd daemon provides REST API services, including REST API Explorer. PyEZ RPCs rely on mgd, and JSNAPy is Python-based, not directly tied to daemons. Therefore, jsd specifically supports REST interfaces. Without jsd, REST automation cannot function.

Question: 94

Which two tasks are commonly automated using Jinja2 templates? (Choose two)

A. Generating device configurations
B. Building Ansible playbooks
C. Performing software upgrades

D. Taking device snapshots

Answer: A, B

Explanation:

Jinja2 templates are used to generate configs and build playbooks dynamically. Software upgrades are handled by PyEZ Sw utility, and snapshots are handled by JSNAPy. Jinja2 is designed for templating, not direct device management, making it complementary to other tools.

Question: 95

Which Python tool helps ensure network state validation during CI/CD pipelines?

A. Jinja2
B. PyEZ
C. JSNAPy
D. REST API

Answer: C

Explanation:

JSNAPy is used for validating network state before/after changes and in CI/CD pipelines. It helps enforce compliance and detect unintended changes. Jinja2 handles templating, PyEZ manages configs, and REST API provides programmatic access. JSNAPy ensures state correctness in DevOps workflows.

Question: 96

Which two authentication methods are supported by Junos REST API? (Choose two)

A. Username/password
B. OAuth tokens
C. LDAP bind
D. SNMPv3

Answer: A, B

Explanation:

REST API supports username/password and token-based authentication (OAuth/JWT). LDAP can be integrated indirectly through Junos AAA, but it is not direct REST auth. SNMPv3 is unrelated to REST API. These methods ensure secure access for automation platforms.

Question: 97

Which Jinja2 feature can transform variable values before rendering them into templates?

A. Filters

B. Validators
C. Control structures
D. Parsers

Answer: A
Explanation:

Filters in Jinja2 allow modifying variables (e.g., uppercasing, trimming whitespace) before rendering them into templates. Validators belong to JSNAPy, control structures handle loops, and parsers are unrelated. Filters make templates more flexible and adaptable to data.

Question: 98

Which two outputs can JSNAPy produce after validation runs? (Choose two)

A. Pass/Fail results
B. Detailed logs
C. REST API schema
D. Jinja2 filters

Answer: A, B
Explanation:

JSNAPy produces pass/fail validation results and detailed execution logs. REST API schemas and Jinja2 filters are unrelated. These outputs are critical for auditing automation workflows and integrating with CI/CD tools for compliance checks.

Question: 99

Which REST API advantage makes it preferred for cloud-native DevOps?

A. Supports NETCONF transactions
B. Uses HTTP/HTTPS with JSON payloads
C. Works only with XML schemas
D. Requires mgd daemon

Answer: B
Explanation:

REST API is cloud-native friendly because it uses HTTP/HTTPS with JSON payloads, allowing integration with CI/CD pipelines and microservices. NETCONF transactions belong to mgd, not REST. REST supports XML as well but is mostly JSON-focused. This makes it ideal for DevOps.

Question: 100

Which two tools often work together in Junos automation pipelines? (Choose two)

A. Jinja2
B. JSNAPy
C. Telnet
D. Excel macros

Answer: A, B

Explanation:

Jinja2 and JSNAPy often complement each other: Jinja2 generates configs, and JSNAPy validates states pre- and post-deployment. Telnet is obsolete in automation, and Excel macros are unrelated. These two tools align with PyEZ and REST APIs in Junos automation pipelines.

Question: 101

What is the primary purpose of an Ansible playbook in Junos automation?

A. To define device inventory

B. To execute ad-hoc CLI commands
C. To describe automation workflows in YAML
D. To replace NETCONF RPCs

Answer: C

Explanation:

Ansible playbooks describe automation workflows in YAML format. They define tasks, roles, and configurations that Ansible executes against Junos devices. Inventories list devices but are separate, while CLI ad-hoc commands are one-off. Playbooks provide structured, reusable workflows for configuration and state management.

Question: 102

Which two automation components are typically included in an Ansible playbook? (Choose two)

A. Hosts
B. Tasks
C. Python exceptions
D. RPC filters

Answer: A, B

Explanation:

Playbooks define hosts (devices or groups to automate) and tasks (actions executed on them). Python exceptions and RPC filters are not part of YAML playbook structure. Tasks often call Junos modules or templates. This structure ensures modular and readable automation workflows.

Question: 103

What is the role of Ansible Tower in Junos automation?

A. Provides a web-based GUI for automation management
B. Replaces YAML playbooks with XML scripts
C. Directly executes NETCONF RPCs without Ansible core
D. Manages REST API calls only

Answer: A
Explanation:

Ansible Tower is a GUI and enterprise framework that centralizes automation, job scheduling, RBAC, and reporting. It complements Ansible CLI by providing governance and visibility. Tower does not replace playbooks or limit automation to RPCs/REST. It makes large-scale Junos automation more manageable.

Question: 104

Which templating engine is most commonly used with Ansible playbooks for Junos configs?

A. Velocity
B. Jinja2
C. Mako
D. JSONnet

Answer: B
Explanation:

Jinja2 is the standard templating engine used in Ansible for dynamic configuration generation. It allows variables, loops, and conditionals to customize configs per device. Other engines exist but are not used with Ansible. Jinja2 enables modular and scalable Junos automation playbooks.

Question: 105

Which two advantages does Ansible offer when automating Junos devices? (Choose two)

A. Agentless operation over SSH/NETCONF
B. Built-in rollback like Junos CLI
C. Idempotency to ensure desired state
D. Requires Python installed on devices

Answer: A, C
Explanation:

Ansible is agentless, requiring no software on devices, and is idempotent, ensuring configs match the

desired state regardless of initial state. Rollback is a Junos feature, not an Ansible one. Devices don't need Python since Ansible connects via NETCONF/SSH. This simplicity makes Ansible powerful for Junos.

Question: 106

Which Ansible module is specifically designed for Junos configuration changes?

A. junos_config
B. junos_rpc
C. junos_facts
D. junos_commit

Answer: A
Explanation:

The junos_config module allows pushing configuration changes to Junos devices. It supports load, merge, replace, and commit operations. junos_rpc executes RPCs, junos_facts retrieves device facts, and junos_commit is not an actual module. junos_config is central for configuration management.

Question: 107

Which two outputs can be generated by Ansible playbooks for verification? (Choose two)

A. Device facts (hostname, version, interfaces)
B. Pre/post configuration diffs
C. NETCONF schemas
D. RPC timeouts

Answer: A, B
Explanation:

Ansible playbooks can collect device facts and show configuration diffs before and after commits. NETCONF schemas and RPC timeouts are backend details, not standard outputs of playbooks. This verification ensures automation is safe, predictable, and auditable.

Question: 108

Which statement correctly describes idempotency in Ansible playbooks?

A. Each playbook always re-applies configs regardless of state
B. Playbooks ensure the device matches the desired state
C. Ansible randomizes task execution order
D. It requires manual rollback after failure

Answer: B
Explanation:

Idempotency means Ansible ensures the device ends up in the desired state, applying changes only if necessary. This avoids unnecessary updates. It does not reapply blindly, randomize tasks, or require rollback. Idempotency is critical for reliable automation in Junos.

Question: 109

Which feature in Ansible Tower allows automation jobs to run on a schedule?

A. Templates
B. Roles
C. Schedules
D. Inventory

Answer: C

Explanation:

Schedules in Tower allow automation jobs to run at predefined times. Templates define job workflows, roles group tasks, and inventories list devices. Scheduling ensures regular configuration checks, backups, or compliance enforcement in Junos environments.

Question: 110

Which two formats can Ansible playbooks output results in? (Choose two)

A. JSON
B. YAML
C. HTML
D. Plain text

Answer: A, D

Explanation:

Ansible playbook outputs are usually JSON (machine-parsable for pipelines) or plain text (human-readable). YAML is input format, not output. HTML is possible only with external plugins, not by default. JSON/plain text allow integration with CI/CD and monitoring systems.

Question: 111

What is the function of Ansible inventory files?

A. Define automation workflows
B. Specify devices or groups to target
C. Provide rollback logic for configs
D. Contain Jinja2 templates

Answer: B
Explanation:

Inventories define devices or groups that Ansible playbooks will target. They may include IPs, hostnames, and connection parameters. Playbooks define workflows, rollback is handled by Junos, and Jinja2 templates are separate. Inventory files are essential to scale automation across multiple Junos devices.

Question: 112

Which two benefits does Jinja2 templating bring to Ansible playbooks? (Choose two)

A. Reusability of templates across devices
B. Dynamic generation of configs using variables
C. Automatic rollback of failed commits
D. Predefined REST schema validation

Answer: A, B
Explanation:

Jinja2 templates provide reusability and dynamic variable substitution, enabling flexible configuration generation. Rollback is handled by Junos CLI, not Jinja2, and REST schema validation is unrelated. Templates ensure consistency and reduce errors in Junos automation workflows.

Question: 113

Which Ansible module would you use to execute a Junos RPC directly?

A. junos_config
B. junos_rpc
C. junos_facts
D. junos_command

Answer: B
Explanation:

The junos_rpc module executes Junos RPCs and retrieves structured outputs. junos_config manages configs, junos_facts retrieves device details, and junos_command executes CLI commands. junos_rpc is key when you need to automate Junos operational checks.

Question: 114

Which two features are provided by Ansible Tower for enterprise environments? (Choose two)

A. Role-based access control (RBAC) B. Centralized logging and reporting C. Native NETCONF commit/rollback D. Jinja2 loop execution

Answer: A, B
Explanation:

Tower provides enterprise governance features like RBAC and centralized reporting. NETCONF commit/rollback is done by devices, not Tower. Jinja2 loop execution is part of Ansible itself. Tower makes automation more controlled and auditable in Junos networks.

Question: 115

Which file format is used to write Ansible playbooks?

A. XML
B. JSON
C. YAML
D. INI

Answer: C
Explanation:

Ansible playbooks are written in YAML, chosen for its readability and structured format. YAML is also used in other DevOps tools like Kubernetes and JSNAPy. XML and JSON are supported in Junos APIs, but not for Ansible playbooks. INI is used for inventories, not playbooks.

Question: 116

Which two ways can Ansible variables be provided for Jinja2 templates? (Choose two)

A. From inventory files
B. From extra-vars in command line
C. From PyEZ device sessions
D. From syslog messages

Answer: A, B
Explanation:

Variables for Jinja2 templates can be supplied through inventory files or extra-vars on the CLI. PyEZ device sessions are separate, and syslog is irrelevant. This flexibility allows dynamic parameterization in Ansible workflows, making playbooks adaptable across devices.

Question: 117

Which two outputs can be captured from junos_facts module in Ansible? (Choose two)

A. Hostname and software version
B. Interface list
C. RPC timeout settings

D. REST API schema

Answer: A, B

Explanation:

The junos_facts module gathers hostname, version, and interface details among other facts. RPC timeout and REST schema are unrelated to Ansible facts. These outputs are useful for conditional logic in playbooks and inventory management.

Question: 118

Which Ansible component provides reusable automation code units?

A. Inventories
B. Tasks
C. Roles
D. Templates

Answer: C

Explanation:

Roles allow organizing reusable units of automation, combining tasks, variables, and templates into a modular package. Inventories define targets, tasks execute operations, and templates generate configs. Roles improve scalability and reusability across Junos automation projects.

Question: 119

Which two integrations make Ansible Tower valuable for Junos automation pipelines? (Choose two)

A. CI/CD tools (Jenkins, GitLab)
B. ChatOps (Slack, Teams)
C. SNMP trap forwarding
D. CLI rollback

Answer: A, B

Explanation:

Tower integrates with CI/CD platforms and ChatOps tools, allowing automation jobs to be triggered and monitored collaboratively. SNMP traps and CLI rollback are unrelated to Tower. This integration aligns Junos automation with modern DevOps workflows.

Question: 120

Which Jinja2 feature allows templates to include conditional logic?

A. Filters

B. Control structures
C. Variables
D. Playbooks

Answer: B
Explanation:

Control structures (if, else, for) in Jinja2 allow dynamic logic inside templates. This enables per-device customization without duplicating configs. Filters modify values, variables hold data, and playbooks orchestrate tasks. Control structures bring intelligence to Jinja2 templates in Junos automation.

Question: 121

Which two scripting languages are supported natively for on-box Junos automation? (Choose two)

A. Python
B. Perl
C. SLAX
D. Ruby

Answer: A, C
Explanation:

Junos supports Python and SLAX for on-box scripting. Python offers flexibility with rich libraries, while SLAX is Juniper's XML-based scripting language optimized for Junos. Perl and Ruby are not natively supported. These languages form the foundation of Junos automation scripting.

Question: 122

Which type of Junos automation script executes immediately after a configuration is committed?

A. Event script
B. Commit script
C. Op script
D. Translation script

Answer: B
Explanation:

Commit scripts run automatically after a configuration is committed. They can validate, modify, or reject the configuration. Event scripts are triggered by system events, op scripts are run manually, and translation scripts convert SNMP OIDs. Commit scripts ensure configuration integrity.

Question: 123

Which two benefits are provided by Junos event scripts? (Choose two)

A. Trigger automated actions on alarms

B. Validate configuration before commit
C. Execute based on syslog messages
D. Automatically translate SNMP OIDs

Answer: A, C

Explanation:

Event scripts allow Junos to react to system events like syslog messages, traps, or alarms, triggering corrective actions. Commit scripts validate configuration, and SNMP translation is separate. Event scripts are critical for building self-healing automation workflows.

Question: 124

Which type of Junos script is executed by the user manually from the CLI?

A. Commit script
B. Translation script
C. Op script
D. Event script

Answer: C

Explanation:

Op scripts (operational scripts) are run manually by users from the Junos CLI, often to automate repetitive operational tasks. Commit scripts run after configuration commits, event scripts run automatically on events, and translation scripts handle SNMP OID conversions.

Question: 125

Which two Junos script types can modify configuration data? (Choose two)

A. Commit scripts
B. Event scripts
C. Op scripts
D. SNMP scripts

Answer: A, B

Explanation:

Commit scripts can alter configuration during commit, and event scripts can make configuration changes in response to events. Op scripts typically perform show-like tasks, and SNMP scripts only translate data. Commit and event scripts directly influence configuration states.

Question: 126

Which Junos automation script type is primarily used for validating or rejecting configurations?

A. Commit script
B. Event script
C. Op script
D. Python RPC script

Answer: A
Explanation:

Commit scripts can enforce compliance by validating configuration syntax and semantics before they are applied. They may reject incorrect or unauthorized configs. Event scripts trigger on conditions, op scripts are user-driven, and Python RPCs retrieve data. Commit scripts enforce governance.

Question: 127

Which scripting language was developed by Juniper as a more readable alternative to XSLT for automation?

A. SLAX Python YAML JSON
B.
C. **Answer: A**
D. **Explanation:**

SLAX (Style Sheet Language Alternative for XML) is Juniper's scripting language, designed as a more readable alternative to XSLT. It is tailored for XML-based Junos automation. Python, YAML, and JSON serve different purposes. SLAX scripts are heavily used for commit and op scripts.

Question: 128

Which two Junos automation script types use SLAX or Python? (Choose two)

A. Commit scripts
B. Translation scripts
C. Op scripts
D. SNMP scripts

Answer: A, C
Explanation:

Commit scripts and op scripts can be written in SLAX or Python. Translation and SNMP scripts have their own formats and are not typically implemented in SLAX/Python. These script types allow flexibility in enforcing config policies and simplifying operational workflows.

Question: 129

Which Junos script type is used to translate SNMP OIDs into human-readable values?

A. Commit script
B. Event script
C. Translation script
D. Op script

Answer: C

Explanation:

Translation scripts handle the mapping of SNMP OIDs to human-readable values in Junos. This allows SNMP managers to interpret vendor-specific or extended OIDs. Commit scripts validate configs, event scripts respond to triggers, and op scripts automate show commands.

Question: 130

Which automation scripting option in Junos supports interaction with external libraries and APIs?

A. Python
B. SLAX
C. Translation scripts
D. SNMP scripts

Answer: A

Explanation:

Python provides access to external libraries and APIs, making it highly extensible for Junos automation. SLAX is Junos-specific, and translation/SNMP scripts are limited in scope. Python's flexibility allows integration with cloud APIs, databases, and external systems.

Question: 131

Which two contexts can Python scripts run in Junos? (Choose two)

A. On-box (within Junos device)
B. Off-box (remote automation server)
C. SNMP MIBs
D. Kernel processes

Answer: A, B

Explanation:

Python scripts can run on-box directly in Junos OS or off-box on automation servers. On-box execution is

suitable for event/commit scripts, while off-box uses PyEZ or REST APIs. SNMP MIBs and kernel processes are unrelated. This dual flexibility makes Python very powerful.

Question: 132

Which two script types support rollback functionality when used in Junos? (Choose two)

A. Commit scripts
B. Event scripts
C. Op scripts
D. Translation scripts

Answer: A, B

Explanation:

Commit and event scripts can both modify configuration and trigger rollback when errors occur. Op scripts do not alter configurations, and translation scripts are unrelated. Rollback ensures system stability and compliance in automation workflows.

Question: 133

Which Junos scripting language uses curly-brace and C-like syntax for readability?

A. SLAX
B. XSLT
C. Python
D. YAML

Answer: A

Explanation:

SLAX improves on XSLT by introducing C-like curly-brace syntax, making it easier for developers. Python uses indentation, YAML uses whitespace, and XSLT is verbose XML. SLAX's syntax simplifies Junos automation scripts, making them more maintainable.

Question: 134

Which two Junos script types are designed for proactive automation (self-healing)? (Choose two)

A. Commit scripts
B. Event scripts
C. Op scripts
D. SNMP scripts

Answer: A, B
Explanation:

Commit scripts ensure configurations meet standards proactively, while event scripts respond to alarms, syslogs, or failures automatically. Together they enable self-healing automation. Op scripts require manual execution, and SNMP scripts just provide translations, not corrective actions.

Question: 135

Which scripting type in Junos is most suitable for automating routine operational tasks (like show commands)?

A. Commit script
B. Event script
C. Op script
D. Translation script

Answer: C
Explanation:

Op scripts automate operational tasks such as running show commands, formatting outputs, or simplifying troubleshooting. They do not enforce configs like commit scripts or handle triggers like event scripts. Translation scripts are only for SNMP data. Op scripts save operator time.

Question: 136

Which Junos script type is most suited for real-time reaction to alarms?

A. Commit script
B. Event script
C. Op script
D. Translation script

Answer: B
Explanation:

Event scripts are triggered by system events such as alarms, syslog messages, or SNMP traps. They can initiate actions like reconfigurations or notifications. Commit scripts work only during config changes, op scripts are user-driven, and translation scripts handle SNMP OIDs.

Question: 137

Which two benefits of SLAX scripts differentiate them from XSLT? (Choose two)

A. C-like syntax for readability
B. Easier debugging and maintenance

C. Built-in access to Python libraries
D. Direct REST API integration

Answer: A, B

Explanation:

SLAX provides a C-like syntax and improved readability over XSLT, making it easier to write, debug, and maintain. Python libraries and REST integration are unrelated to SLAX. SLAX improves XML automation specifically for Junos scripting use cases.

Question: 138

Which script type allows Junos to proactively reject invalid configuration changes?

A. Event script
B. Commit script
C. Op script
D. Translation script

Answer: B

Explanation:

Commit scripts can validate and reject invalid configuration changes before they are applied. This ensures compliance with policies and prevents misconfiguration. Event scripts trigger after events, op scripts are manual, and translation scripts are for SNMP. Commit scripts enforce governance.

Question: 139

Which two scripting frameworks in Junos can be used for on-box real-time execution? (Choose two)

A. Python
B. SLAX
C. YAML
D. JSON

Answer: A, B

Explanation:

Both Python and SLAX can run on-box in Junos, handling commit, op, and event scripts. YAML and JSON are data formats, not execution frameworks. On-box scripting allows automation to run directly within the device, reducing external dependencies.

Question: 140

Which two script types can interact with syslog and SNMP traps in Junos? (Choose two)

A. Event scripts
B. Commit scripts
C. Translation scripts
D. Op scripts

Answer: A, C

Explanation:

Event scripts can react to syslog messages and traps to trigger automation. Translation scripts help by converting SNMP OIDs to human-readable outputs, aiding monitoring integration. Commit scripts validate configs, and op scripts are operator-driven. Event + Translation scripts tie closely to monitoring.

Question: 141

What is the primary purpose of a YANG model in network automation?

A. Define CLI syntax for Junos
B. Describe structured data for device configuration and state
C. Generate SNMP traps automatically
D. Execute RPC calls directly

Answer: B

Explanation:

YANG models define structured data for configuration, operational state, and management. They serve as schemas for APIs like NETCONF and RESTCONF. They do not define CLI syntax or generate SNMP traps. RPCs are based on YANG definitions but are not executed by the model itself.

Question: 142

Which two APIs in Junos rely on YANG models? (Choose two)

A. NETCONF
B. RESTCONF
C. SNMP
D. CLI

Answer: A, B

Explanation:

Both NETCONF and RESTCONF rely on YANG models to structure their data. SNMP relies on MIBs, and CLI is human-oriented, not model-driven. YANG provides consistency across APIs, enabling automation frameworks to interoperate using standard schemas.

Question: 143

Which statement best describes OpenConfig in Junos automation?

A. Juniper-specific extension to NETCONF
B. A vendor-neutral, community-driven YANG model
C. A proprietary Junos RPC interface
D. Replacement for Junos PyEZ

Answer: B
Explanation:

OpenConfig is a vendor-neutral, open-source initiative defining common YANG models for interoperability across vendors. It is community-driven, not Juniper-only. It does not replace PyEZ or act as RPC. Junos supports OpenConfig models alongside native YANG modules.

Question: 144

Which two benefits do YANG models provide in automation? (Choose two)

A. Data validation using schemas
B. Consistency across vendors
C. Reduction of event-script triggers
D. CLI syntax enforcement

Answer: A, B
Explanation:

YANG models provide schema validation for configuration and ensure consistency across vendors when using standard models like OpenConfig. They don't enforce CLI syntax or replace event scripts. Their structured nature makes automation predictable and scalable.

Question: 145

Which Junos daemon is responsible for handling YANG-based NETCONF requests?

A. jsd
B. rpd
C. mgd
D. snmpd

Answer: C
Explanation:

The mgd daemon manages NETCONF, which relies on YANG models for structure. jsd supports REST API, rpd manages routing protocols, and snmpd handles SNMP MIBs. mgd is central for model-driven

automation using YANG in Junos.

Question: 146

Which two YANG-based standards are supported by Junos for telemetry? (Choose two)

A. gNMI
B. gRPC
C. OpenConfig
D. JSON-RPC

Answer: A, C
Explanation:

Junos supports gNMI (as a transport) and OpenConfig YANG models for telemetry. gRPC provides the transport mechanism but is not a YANG model. JSON-RPC is unrelated. YANG + OpenConfig ensures structured telemetry streaming across vendors.

Question: 147

Which feature of YANG makes it suitable for data modeling in automation?

A. Binary encoding for telemetry
B. Hierarchical tree-based structure
C. Embedded rollback transactions
D. Dependency on CLI parsing

Answer: B
Explanation:

YANG uses a tree-based hierarchical structure, which mirrors how configurations and states are represented on devices. This makes it intuitive for network automation. Rollback is handled by Junos, not YANG, and it avoids CLI parsing entirely. Binary encoding is unrelated.

Question: 148

Which two YANG-based operations are supported by Junos via NETCONF? (Choose two)

A. <get-config>

B. <edit-config>
C. <send-trap>
D. <show-command>

Answer: A, B
Explanation:

NETCONF operations like <get-config> and <edit-config> rely on YANG schemas to retrieve or modify structured data. <send-trap> is SNMP, and <show-command> is CLI-based. YANG ensures consistent schema definitions for configuration and operational RPCs.

Question: 149

What is the role of OpenConfig in multi-vendor networks?

A. Provide Juniper-specific YANG models
B. Enable uniform configuration across vendors
C. Replace native Junos YANG modules
D. Enforce CLI-only automation

Answer: B
Explanation:

OpenConfig provides uniform YANG models to enable consistent automation across different vendors. It doesn't replace Junos native models but complements them. CLI-only automation is not OpenConfig's scope. OpenConfig ensures interoperability and simplifies large-scale deployments.

Question: 150

Which two output encodings can YANG-modeled data be represented in Junos APIs? (Choose two)

A. XML JSON YAML CSV
B.
C. **Answer: A, B**
D. **Explanation:**

Junos supports XML and JSON encodings for YANG-modeled data. XML is standard for NETCONF, while JSON is widely used in RESTCONF and telemetry. YAML and CSV are not supported encodings for YANG in Junos. These encodings make YANG consumable by automation tools.

Question: 151

Which statement best describes the relationship between YANG and NETCONF?

A. YANG defines the transport; NETCONF defines the schema
B. YANG defines data models; NETCONF transports and manipulates them

C. NETCONF is vendor-neutral; YANG is Juniper-specific
D. YANG replaces NETCONF in Junos

Answer: B
Explanation:

YANG defines the data models, while NETCONF provides the transport and operations (like <get>, <edit>, <commit>) for interacting with those models. They complement each other: YANG ensures structure, and NETCONF ensures secure delivery. YANG does not replace NETCONF.

Question: 152

Which two components are included in a YANG model definition? (Choose two)

A. Containers
B. Leaves
C. CLI commands
D. Syslog facilities

Answer: A, B
Explanation:

YANG models use containers (grouping elements) and leaves (individual data elements) to define hierarchical data. CLI commands and syslog are unrelated. These components provide structure, ensuring Junos APIs deliver consistent configuration and state data.

Question: 153

Which OpenConfig model is commonly used to configure interfaces?

A. openconfig-telemetry
B. openconfig-interfaces
C. openconfig-bgp
D. openconfig-platform

Answer: B
Explanation:

openconfig-interfaces provides a standardized model for configuring and monitoring network interfaces across vendors. Other models handle BGP, telemetry, and hardware. By using openconfig-interfaces, automation ensures consistent workflows regardless of vendor CLI differences.

Question: 154

Which two YANG concepts represent reusable building blocks for modeling? (Choose two)

A. Groupings
B. Typedefs
C. SNMP MIBs
D. Commit scripts

Answer: A, B

Explanation:

Groupings (reusable structures) and typedefs (custom data types) in YANG allow modular and reusable modeling. SNMP MIBs are separate from YANG, and commit scripts are Junos automation tools, not YANG features. These concepts improve maintainability of YANG models.

Question: 155

Which statement describes the native Junos YANG modules?

A. They are vendor-neutral standards
B. They represent all CLI commands directly
C. They model Junos-specific configuration and operational data
D. They replace OpenConfig models

Answer: C

Explanation:

Native Junos YANG modules describe Junos-specific configuration and state data not covered by standard OpenConfig models. They complement OpenConfig but do not replace it. They don’t model CLI commands; instead, they define structured data consumed via APIs.

Question: 156

Which two transports can be used with YANG-modeled data in Junos? (Choose two)

A. NETCONF
B. RESTCONF
C. Syslog
D. FTP

Answer: A, B

Explanation:

Junos supports NETCONF and RESTCONF as transports for YANG data. Syslog and FTP are not YANG-related transports. NETCONF uses XML encoding, while RESTCONF often uses JSON. These transports ensure YANG-modeled automation is flexible and API-driven.

Question: 157

Which OpenConfig model would you use for BGP configuration?

A. openconfig-interfaces
B. openconfig-bgp
C. openconfig-platform
D. openconfig-system

Answer: B
Explanation:

openconfig-bgp defines vendor-neutral configuration and state models for BGP. Interfaces are handled by openconfig-interfaces, hardware by openconfig-platform, and system functions by openconfig-system. OpenConfig BGP ensures cross-vendor consistency in routing automation.

Question: 158

Which two benefits do OpenConfig models bring to service providers? (Choose two)

A. Multi-vendor interoperability
B. Faster adoption of vendor CLI features
C. Standardized telemetry data
D. Automatic rollback of configurations

Answer: A, C
Explanation:

OpenConfig ensures multi-vendor interoperability and standardized telemetry outputs, enabling consistent automation across diverse networks. It doesn't accelerate CLI feature adoption or provide rollback. OpenConfig reduces fragmentation and aligns with DevOps and cloud practices.

Question: 159

Which Junos feature allows operators to browse YANG models available on a device?

A. show configuration scripts
B. show system commit
C. file show /opt/models/yang
D. show system schema

Answer: D
Explanation:

The show system schema command displays the YANG models implemented on a Junos device. This includes both native and OpenConfig models. Other options show configuration, commit logs, or file data,

not YANG models. This feature helps operators discover model support.

Question: 160

Which two statements describe the implementation of YANG in Junos? (Choose two)

A. Both native and OpenConfig models are supported
B. Only Junos-specific models exist
C. Models can be used with NETCONF and RESTCONF
D. YANG models must replace CLI entirely

Answer: A, C

Explanation:

Junos supports both native YANG modules and OpenConfig. These can be consumed via NETCONF or RESTCONF APIs. CLI still coexists, and YANG does not replace it. This hybrid support ensures flexibility for both legacy CLI users and modern API-driven automation.