Servicenow

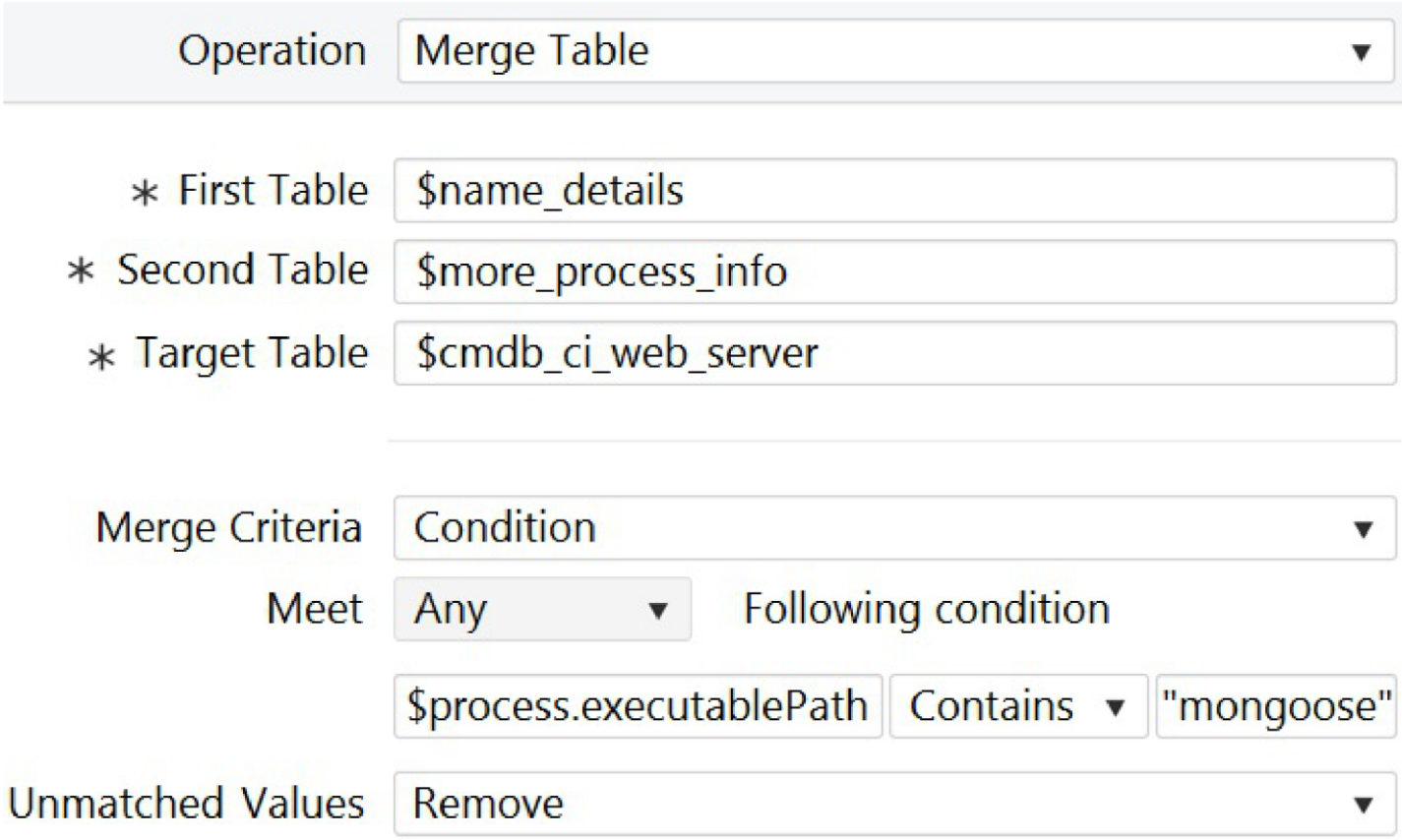
(CIS-Discovery)

Certified Implementation Specialist - Discovery

Total: **107 Questions**

Link:

Question: 1



Based on this image, which of the following statements are true? (Choose three.)

A.Attributes from two tables populate a table with the same name as a ServiceNow CMDB table. B.This operation is more than likely a part of a step on a pattern set to Application Pattern Type. C.If a value is unmatched, it is still merged into the Target Table.

D.For this operation to run, there must be some data in the process.executablePath variable.

E.This is a horizontal pattern of type "infrastructure."

Answer: ABD

Explanation:

A. Attributes from two tables populate a table with the same name as a ServiceNow CMDB table.

This describes how Union Table or Transform Table operations work: combining or transforming data from multiple sources and mapping it to a target table, often a CMDB CI class like cmdb_ci_appl or cmdb_ci_win_server.

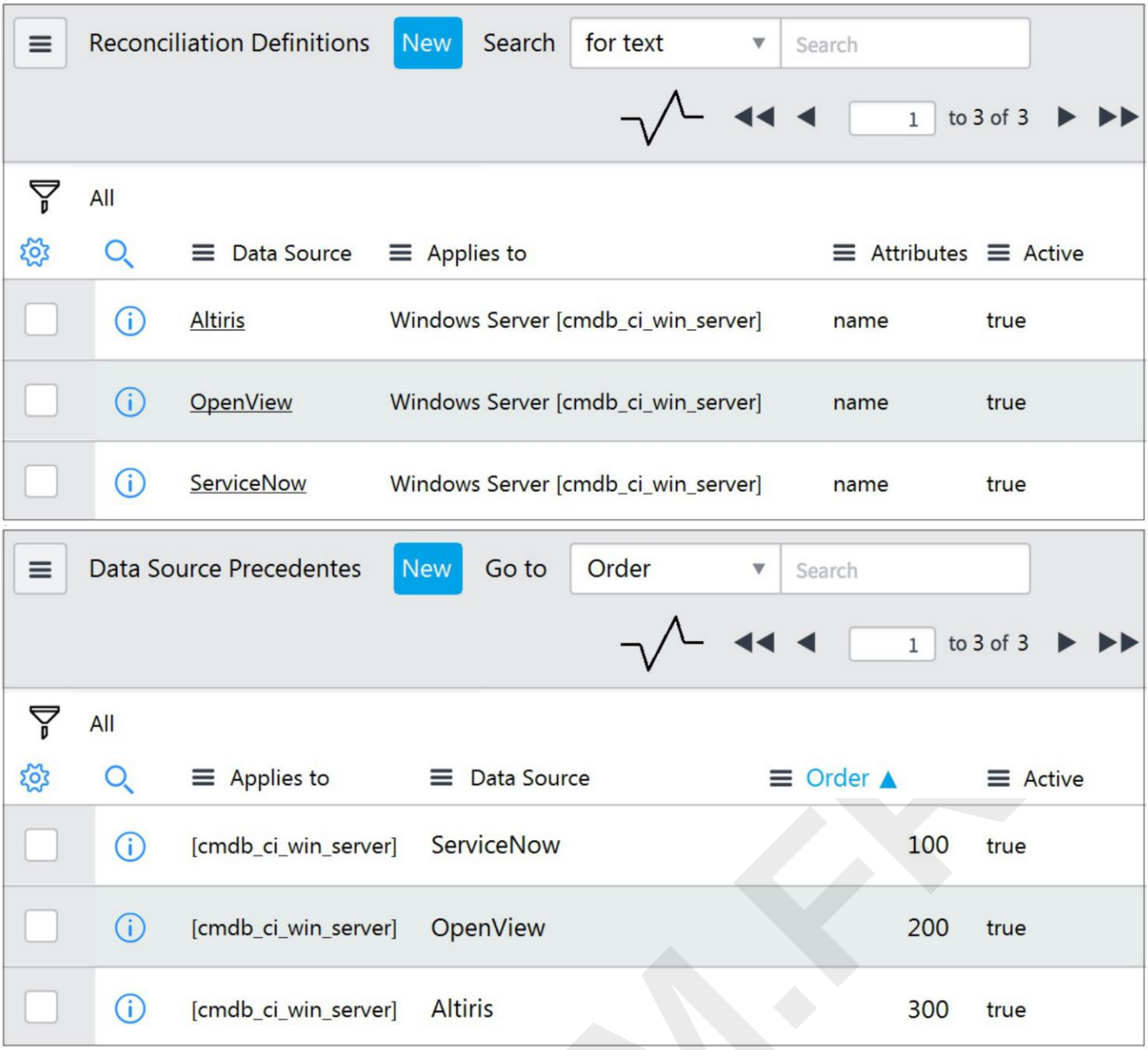
B. This operation is more than likely a part of a step on a pattern set to Application Pattern Type.

These types of operations (especially involving executable paths, processes, or combining attributes) are commonly found in Application-type patterns, which focus on discovering software and services running on top of OS layers.

D. For this operation to run, there must be some data in the process.executablePath variable.

If the operation involves parsing or transforming process information based on the process.executablePath, this variable must be populated for the logic to apply.

Question: 2



Based on the following images, which choice best describes what occurs if Discovery sets the name attribute of a discovered Windows Server CI to 'Windows1' and then Altiris discovery runs detecting 'Windows2' for the name attribute on the same CI?

A.The name of the CI stays 'Windows1'.
B.The name of the CI changes to 'Windows2'.
C.The name of the CI does not populate with either discovery.
D.The CI is not discovered because Discovery is not listed in either image.

Answer: A

Explanation:

A. The name of the CI stays 'Windows1'.

ServiceNow CMDB uses reconciliation rules to determine which discovery source wins when multiple tools populate the same attribute.

The reconciliation engine will retain the value from the highest-priority source (typically Discovery).

Question: 3

For the Parse Variable pattern operation, what is required to have two different parsing methods to populate variables?

A.Two different Debug Mode sessions.

B.A tabular and a scalar variable.

C.Two different steps.

D.Two different Define Parsing selections on the same step.

Answer: C

Explanation:

The correct answer is C: Two different steps.

Here's the justification: The Parse Variable pattern operation in ServiceNow Discovery allows extraction of specific data from a discovered device's output. To use two different parsing methods (e.g., using different regular expressions or text delimiters) to populate different variables, it necessitates creating two distinct steps within the pattern.

Each "Parse Variable" step is configured with a specific parsing method. If you try to apply two different parsing configurations within the same step, you'll likely encounter conflicts or only one configuration being applied. You can't, within a single "Parse Variable" step, logically parse the same source text using two fundamentally different approaches concurrently.

Option A (Two different Debug Mode sessions) is incorrect. Debug Mode is for testing and troubleshooting patterns, not for defining different parsing methods. Option B (A tabular and a scalar variable) is incorrect; the type of variable (tabular or scalar) doesn't dictate the need for separate parsing steps. You can have both tabular and scalar variables populated in a pattern, and the need for separate steps is dictated by different parsing needs, not variable type. Option D (Two different Define Parsing selections on the same step) is incorrect because the ServiceNow platform is built to handle one parsing definition per "Parse Variable" operation.

The ServiceNow documentation supports this approach. While it doesn't explicitly state "you need two steps for two different parsing methods" in those exact words, the structure and examples provided consistently illustrate patterns using a separate "Parse Variable" step for each distinct parsing logic required. By adding steps to discovery pattern, you essentially break down your logic in to units that can be individually modified and debugged.

In essence, each "Parse Variable" step encapsulates a specific parsing strategy. To employ multiple and distinct parsing strategies, one needs separate steps, each representing a unique extraction method and target variable(s). This enhances clarity, modularity, and maintainability of the discovery pattern.

Authoritative links:

ServiceNow Discovery Documentation: The best resource is ServiceNow's official documentation on Discovery patterns and related CMDB features. Unfortunately, direct links to specific sections are not stable as ServiceNow updates its documentation frequently. Search for "ServiceNow Discovery Patterns" in the ServiceNow documentation portal. Look for information on "Parse Variable" operations.

ServiceNow Community Forums: Check ServiceNow's community forums for discussions and examples related to Discovery patterns. This can provide practical insights into pattern creation and best practices.

Question: 4

Which best describes Discovery schedule of type Configuration Item?

A.Verifies Configuration Item data from the scanned IP ranges against the data in the CMDB.
B.Creates only a list of discovered IPs in both IPv4 and IPv6 formats.
C.Collects complete information from the scanned IP ranges and sends it to the CMDB.
D.Directly populates records in the assets table.

Answer: C

Explanation:

The correct answer, C, is justified because a Discovery schedule configured with the "Configuration Item" type focuses on gathering comprehensive data about assets within defined IP ranges and subsequently populating the Configuration Management Database (CMDB). This contrasts with simply listing IPs or verifying existing data; it aims to create and update CMDB records with detailed information.

Discovery uses probes and sensors to collect this information. Probes initiate the data collection process by targeting specific devices within the defined IP range. Once a probe identifies a device, it triggers sensors to gather detailed attributes like hardware specifications, software versions, and network configurations. The collected data is then processed and transformed into CMDB records, ensuring an accurate and up-to-date representation of the IT infrastructure.

Option A, verifying Configuration Item data, implies a comparison or reconciliation process, not the initial population of the CMDB. Option B, creating only a list of IPs, represents a far less comprehensive scan than the CMDB-focused Discovery process. Option D, directly populating the assets table, is incorrect as Discovery primarily targets the CMDB, using the discovered information to populate related asset tables, not directly writing to the Assets table initially. A complete discovery schedule updates the CMDB, which then reflects throughout other tables.

Therefore, the primary role of a "Configuration Item" Discovery schedule is to completely gather information and insert or update the related information into the CMDB, making option C the most accurate description.

For further research, refer to the official ServiceNow documentation on Discovery:

ServiceNow Docs: Discovery:https://docs.servicenow.com/bundle/sandiego-it-operations-management/page/product/discovery/concept/c_Discovery.html
ServiceNow Docs: Discovery Schedules:https://docs.servicenow.com/bundle/sandiego-it-operations-management/page/product/discovery/concept/c_DiscoverySchedules.html

Question: 5

When installing a MID Server on a Windows platform, which right must be associated when creating a Service Account?

A.Root
B.Domain Admin
C.MID Server User Role
D.Log on as service

Answer: D

Explanation:

The correct answer is **D. Log on as service**. Here's a detailed justification:

ServiceNow MID Servers, acting as a bridge between the ServiceNow instance and local networks, require specific permissions when installed on Windows. The "Log on as a service" right is crucial because it allows the Service Account under which the MID Server runs to register itself as a service with the Windows operating system. This registration is fundamental for the MID Server to start automatically upon system boot, run in the background, and perform its intended functions without direct user interaction. Without this right, the MID Server might fail to start or operate reliably as a background process.

The other options are incorrect. Root and Domain Admin are overly broad privileges and grant far more access than necessary, violating the principle of least privilege, a fundamental security best practice in cloud computing and system administration. Giving the MID Server service account excessive rights increases the attack surface and the potential impact of a security breach. The "MID Server User Role" within ServiceNow itself manages access to ServiceNow resources, not Windows operating system privileges required for service execution.

The "Log on as a service" right specifically grants the ability to run a process in the security context of the service account. This is essential for the MID Server's function. It allows the MID Server to execute Discovery probes and other processes on the network, using the credentials of the specified account, while maintaining a proper security boundary. It allows the MID Server service to persistently run in the background.

For more information on MID Server installation and required permissions, refer to the official ServiceNow documentation:

ServiceNow Documentation: MID Server Installation
Microsoft Documentation: Configure the service accounts (This is a general Microsoft resource about service accounts and their rights.)

This explains why "Log on as service" is the necessary right for the Service Account when installing a MID Server on Windows.

Question: 6

Which of the below choices are needed for Quick Discovery? (Choose two.)

A.MID Server
B.Discovery Schedule
C.PID
D.Target IP

Answer: AD

Explanation:

The correct answer is **A. MID Server** and **D. Target IP**.

Here's why:

Quick Discovery is a simplified method within ServiceNow Discovery designed for fast identification of devices on a network. It prioritizes speed and ease of use over comprehensive data collection. Therefore, certain prerequisites and configurations are essential for its function.

A **MID Server (Management, Instrumentation, and Discovery Server)** is absolutely crucial. A MID Server is a lightweight Java application that resides within the customer's network, acting as a bridge between the ServiceNow instance (hosted in the cloud) and the devices to be discovered. Since ServiceNow's cloud

instance cannot directly access internal networks due to security constraints, the MID Server executes the discovery probes and sensors locally, retrieving information and relaying it back to ServiceNow. Without a MID Server deployed and configured, Quick Discovery cannot access and scan devices within the target network.

[https://docs.servicenow.com/bundle/utopia-servicenow-platform/page/product/mid-server/concept/c_MIDServer.html]

A **Target IP** (or a range of IPs) is equally indispensable. Quick Discovery must be told where to look. The target IP address provides the specific location of a device that Quick Discovery should attempt to identify. This is the starting point for the discovery process. Quick Discovery scans this IP address, attempting to determine the device type and basic information. Without a target IP, Quick Discovery has no starting point and cannot function.

Options B and C are incorrect because they represent elements typically related to a comprehensive Discovery setup, but they aren't strictly required for the basic function of Quick Discovery:

A **Discovery Schedule** is used to automate the Discovery process on a regular basis. While valuable for maintaining an up-to-date CMDB, it is not mandatory for Quick Discovery, which can be triggered manually on an ad-hoc basis.

PID (Pattern ID) refers to patterns used in Identification and Reconciliation Engine (IRE) to normalize data. While the IRE is used in Discovery, the patterns are more relevant for detailed, scheduled Discovery, and not strictly needed for the 'Quick' variant.

In summary, Quick Discovery needs a way to access the network (MID Server) and a target to discover (Target IP). These two elements form the bedrock upon which the brief process operates.

Question: 7

In order to use Debug from the Pattern Designer, you must have what?

A.a proxy server
B.a discoverable CI
C.the admin role
D.Service Mapping installed

Answer: B

Explanation:

The correct answer is B: a discoverable CI. To effectively use the Debug feature within ServiceNow's Pattern Designer during Discovery troubleshooting, you need a Configuration Item (CI) that the Discovery process can target and attempt to identify. The Debug function allows you to step through a pattern and observe how it interacts with a specific CI, showing you exactly which steps succeed or fail and why. This interactive debugging requires a tangible CI to operate on.

Option A, a proxy server, is relevant for Discovery configurations where devices are behind a proxy, but it's not a prerequisite for using the Pattern Designer's Debug function. You can debug a pattern even if the target CI is directly accessible. Option C, the admin role, is crucial for accessing and modifying patterns, including debugging them, but it's not the thing being acted on by the pattern. Admin rights are needed to use the debug feature, but it is not needed by the debug functionality itself. Option D, Service Mapping installed, is valuable because it leverages Discovery data to build visual representations of applications and their dependencies. While Service Mapping enhances the value of Discovery, it's not directly tied to the Debug functionality within Pattern Designer. The Debug function will work without Service Mapping, but Service Mapping benefits greatly from well-tuned discovery patterns.

Without a defined, discoverable CI (a device or application instance that the pattern is intended to identify and classify), the Debug feature in Pattern Designer has nothing to analyze and cannot provide meaningful insights into pattern behavior. The purpose of pattern designer debug is to simulate a pattern on a CI and see the results.

Question: 8

A discovery runs against a Windows Server returning the following attribute values for the first time: name = WindowsSN1 serial_number = 12321
A subsequent discovery runs against a different Windows Server returning the following attribute values: name = WindowsSN2 serial_number = 12321
With only base system CI Identifiers configured, which of the following is true?

A.A Windows Server CI is created, then updated with WindowsSN2 as the name.
B.Two Windows Sewer CIs are created, with WindowsSN1 AND WindowsSN2 for names.
C.Two Windows Server CIs are created, without serial_number values.
D.A Windows Server CI is created, then updated with WindowsSN1 as the name.

Answer: A

Explanation:

Here's a detailed justification for why option A is the correct answer:

Discovery in ServiceNow relies heavily on CI Identifiers to determine if a discovered device already exists in the CMDB or if a new CI needs to be created. Base system CI Identifiers typically include attributes like name and serial_number. The key here is understanding how ServiceNow prioritizes these identifiers during reconciliation.

In the first Discovery run, a new CI (Configuration Item) is created for a Windows Server because no existing CI matches the discovered attributes (name = WindowsSN1, serial_number = 12321A). The system creates a Windows Server CI and populates it with these values.

The crucial point comes with the second Discovery run. A server is discovered with name = WindowsSN2 and serial_number = 12321A. The system starts comparing the attributes. Because only base system CI Identifiers are configured, ServiceNow compares using only these attributes. In this scenario, serial_number 12321A matches an existing CI.

Because the serial_number matched, ServiceNow recognizes that the newly discovered server is actually the same server as the one discovered earlier. It does not create a new CI. Instead, it updates the existing CI. In this scenario, because serial_number is configured as an identifier, the name field is updated from WindowsSN1 to WindowsSN2.

Option B is incorrect because matching serial_number identifier prevents creation of a second CI. Option C is wrong as serial_number values are part of the identification process, not omitted. Option D is wrong as the name is updated to reflect the value of the second discovery, not the first.

In conclusion, the first server becomes a CI (WindowsSN1 and serial 12321A), and the second discovery updates the existing CI with new name but same serial.

Further reading:

ServiceNow Documentation on Identification and Reconciliation:https://docs.servicenow.com/bundle/utah-platform-cmdb/page/product/configuration-management/concept/c_IdentificationReconciliation.html

ServiceNow KB on CI Identifiers:https://support.servicenow.com/kb?id=kb_article_view&sysparm_article=KB0715850

Question: 9

Which choice represents the three best ways of extending Discovery?

A.Orchestration, Classifiers, Discovery Patterns
B.Fingerprinting, Classifiers, Discovery Patterns
C.Orchestration, Classifiers, Probes & Sensors
D.Classifiers, Probes & Sensors, Discovery Patterns
E.Classifiers, Fingerprinting, Probes & Sensors

Answer: D

Explanation:

The correct answer, D (Classifiers, Probes & Sensors, Discovery Patterns), represents the three best ways to extend ServiceNow Discovery. Let's break down why:

Classifiers: Classifiers are critical for identifying the type of CI (Configuration Item) being discovered. Extending them allows Discovery to recognize new or custom hardware and software. This ensures the discovered item is correctly categorized and managed within the CMDB (Configuration Management Database). ServiceNow Docs: Classifiers

Probes & Sensors: Probes are responsible for gathering information from target devices. Sensors then process this raw data into meaningful CI attributes. Extending probes and sensors enables Discovery to collect specialized information beyond the standard attributes, like custom application details or specific performance metrics. This allows for granular insights into the environment. ServiceNow Docs: Discovery Probes and Sensors

Discovery Patterns: Discovery Patterns, formerly called Identification and Exploration (IRE) patterns, provide a powerful way to define complex discovery logic. They allow you to automate the identification and relationships of complex applications and infrastructure. Extending patterns enables Discovery to handle intricate environments and custom applications which don't fit standard discovery profiles, therefore are vital to extending discovery's functionality. ServiceNow Docs: Discovery Patterns

Options A and C include Orchestration. While Orchestration can utilize discovered information, it's not fundamentally a method for extending the discovery process itself. It uses the data gathered by Discovery.

Option B and E include Fingerprinting. While fingerprinting (service signatures) can aid in discovery, it's a function within Classifiers and/or Patterns, not a primary method for extending the overall discovery capabilities in the same way as the chosen answer. Fingerprinting is a component used by Classifiers or Patterns.

Therefore, Classifiers, Probes & Sensors, and Discovery Patterns offer the most direct and comprehensive approaches to extending Discovery's capabilities, allowing it to identify, gather data from, and model a wider range of CIs and applications within the environment. They ensure more complete and accurate CMDB data, which is essential for effective IT service management.

Question: 10

SNMP Credentials require which of the following?

A.write community strings
B.usernames
C.read community strings
D.port 135 access

Answer: C

Explanation:

The correct answer is C: read community strings. SNMP (Simple Network Management Protocol) fundamentally operates based on community strings for authentication. These strings act like passwords, allowing access to managed devices. SNMP read community strings are specifically required for a Discovery schedule to successfully retrieve information from devices.

Here's why the other options are incorrect:

A. write community strings: While write community strings exist in SNMP, they are used for modifying device configurations. Discovery primarily reads configuration and status data, not writes it. Using write community strings for Discovery could pose a significant security risk.

B. usernames: SNMP versions 1 and 2c (which are the versions that use community strings) do not use usernames for authentication. Usernames are used in SNMPv3, a more secure version, but it does not utilize community strings.

D. port 135 access: Port 135 is typically associated with Microsoft's Endpoint Mapper service and DCOM (Distributed Component Object Model). This port is relevant for protocols like WMI, which is another discovery method, but is not directly related to SNMP. SNMP primarily uses UDP ports 161 and 162.

Discovery leverages SNMP to query network devices for their configuration, status, and other relevant attributes. For this querying process, the Discovery process needs to provide the correct read community string to authenticate with the device and gain access to its data. If the read community string is incorrect, the device will reject the Discovery probe, and the data collection will fail. Therefore, specifying the correct read community string is a critical requirement for successful SNMP-based Discovery.The ServiceNow documentation emphasizes the usage of SNMP and the necessity of having the appropriate read community string for discovery purposes.

Refer to the ServiceNow documentation for more information on SNMP Discovery:

ServiceNow Discovery Documentation
SNMP Protocol in Discovery

Question: 11

Which choice will populate the Location field for a discovered CI?

A.Location field for a Discovery Schedule
B.Location field for a parent CI Type
C.Location field for a Port Probe
D.Location report from the Discovery Dashboard

Answer: A

Explanation:

The answer is **A. Location field for a Discovery Schedule.**

Discovery schedules can be configured with specific locations. When a Discovery schedule runs, it uses this specified location to populate the Location field of the CIs it discovers. This allows you to associate discovered devices with a particular physical place, which is important for accurate asset management and incident routing.

A CI might not always have a direct parent CI type that inherently provides location information, so using a CI type's Location field (option B) is less reliable as a source. Port Probes (option C) themselves don't inherently define location; they are tools used during discovery. The Discovery Dashboard (option D) presents reports, but doesn't directly set the location of discovered CIs. The Discovery Schedule location offers a centralised and consistent method of associating devices with physical spaces, ensuring organizational consistency and improved operational efficiency.Reference:

ServiceNow Discovery Documentation (Specifically, review sections related to Discovery Schedules and how they are configured.)

Question: 12

What role is needed by the MID Server's user account to interact with a ServiceNow instance?

A.mid_server
B.discovery_admin
C.sm_mid
D.mid_discovery

Answer: A

Explanation:

The correct answer is **A. mid_server**. Here's a detailed justification:

The MID Server (Management, Instrumentation, and Discovery) acts as a bridge between the ServiceNow instance and external resources, such as servers, network devices, and cloud services, that reside within the on-premise network or cloud infrastructure. To securely communicate with the ServiceNow instance and execute Discovery operations, the MID Server uses a dedicated user account.

This user account requires specific permissions to interact with the ServiceNow instance without granting excessive or unnecessary privileges. The mid_server role is the minimum required role for the MID Server's user to authenticate and perform its core functions. This role grants the necessary permissions to connect to the instance, receive instructions, and send results back.

Option B, discovery_admin, grants administrative privileges related to the Discovery application, which are not required for the basic operation of the MID Server itself. While a discovery_admin user can be used for the MID Server user, it's generally best practice to follow the principle of least privilege and grant only the necessary permissions, which the mid_server role provides. The roles sm_mid and mid_discovery are not standard roles within the ServiceNow platform and would not enable MID server connectivity. Choosing the most appropriate role limits the potential blast radius in case of a security breach or misconfiguration. The mid_server role directly facilitates the communication and operation of the MID Server agent. The MID Server user does not need to directly administer discovery to simply report the discoveries it finds.

Authoritative links for further research:

ServiceNow Docs - MID Server roles: Search ServiceNow official documentation for "MID Server roles". They

contain detailed information about the roles required for various MID Server operations, including the fundamental mid_server role.

ServiceNow Community Forums: Search within the ServiceNow community forums for discussions about MID Server user roles and best practices. These discussions often provide practical insights from experienced ServiceNow administrators and developers.

Question: 13

Which operation is used to change from the default credentials to any other appropriate credentials in a horizontal pattern?

A.Change credentials
B.Change user
C.Alternate credentials
D.Alternate user

Answer: B

Explanation:

The correct answer is **C. Alternate credentials**. Let's break down why.

Horizontal Discovery patterns aim to identify and classify Configuration Items (CIs) across your network by querying devices using various protocols like SSH, WMI, or SNMP. During the execution of a pattern, Discovery often encounters scenarios where the default or initial credentials don't grant the necessary access to retrieve the required information from a target device. This necessitates a switch to different credentials.

The term "Alternate credentials" specifically refers to this dynamic credential switching process within a Discovery pattern. It allows the pattern to cycle through a list of defined credentials until it finds a set that provides the appropriate access level. This is a key part of ensuring comprehensive and accurate CI discovery.

"Change user" or "Change credentials" are not standard operations within Discovery patterns. While you might conceptually be changing the user or credentials being used, the operation is recognized and configured within ServiceNow as utilizing "Alternate credentials" defined in the pattern or through credential alias definitions.

"Alternate user" isn't a recognized term within ServiceNow Discovery's context for handling credentials.

Therefore, the choice offering the most accurate and ServiceNow-centric terminology for this process is "Alternate credentials". It's directly tied to the functionality that enables the use of different credentials within a horizontal pattern when the default ones are insufficient.

Refer to the official ServiceNow documentation on Discovery and Credential Management for further details on credential aliases and pattern design.

Question: 14

After navigating to an Automaton Error Messages list from Discovery > Home, how are the options on the right navigation pane categorized? (Choose two.)

A.SELECT ALL
B.SELECT ONE
C.ACTION ON SELECTED

D.ACTION ON ALL

Answer: CD

Explanation:

The correct answer is **C. ACTION ON SELECTED** and **D. ACTION ON ALL**.

After navigating to the Automation Error Messages list in ServiceNow Discovery, the right navigation pane primarily offers actions related to the selected error messages or actions that apply to all error messages listed. This categorization reflects two distinct operational paradigms.

"ACTION ON SELECTED" refers to actions that are performed only on the error messages that the user has explicitly chosen (e.g., marking a specific error message as resolved, assigning it to a specific group, or deleting a selected few). This allows for targeted management of individual or grouped errors based on their specific characteristics or criticality.

"ACTION ON ALL" refers to operations that are executed across all error messages currently displayed in the list. This would include actions like resolving all errors, re-running discovery for all failed devices, or exporting all error data to a report. This approach is useful for bulk operations, especially when addressing widespread issues or generating comprehensive reports.

The underlying design principle aligns with providing efficient error handling and automation within Discovery. By organizing actions into those applied to selected items and those applied universally, ServiceNow enables administrators to quickly triage and remediate problems encountered during discovery processes. This segregation of functions supports both granular control and efficient bulk management, improving overall system administration. The interface is designed to be user-friendly and intuitive, presenting options in a logical manner that facilitates both immediate actions and long-term error trend analysis. By efficiently organizing actions, ServiceNow empowers administrators to identify, investigate, and resolve Discovery-related issues promptly.

Authoritative Links:

While specific documentation directly addressing the categorization within the Automation Error Messages list is sometimes difficult to find granularly, these links provide a broader understanding of Discovery and Error Handling:

ServiceNow Discovery:https://www.servicenow.com/products/discovery.html (General overview)
ServiceNow Documentation:https://docs.servicenow.com/ (Search for "Discovery" and "Error Handling" within the official documentation for more detailed information.)

Question: 15

Which of the following can be used in the Debug Identification Section in Debug Mode for an infrastructure pattern? (Choose two.)

A.IP
B.AWS Endpoint
C.PID
D.Host Name

Answer: AD

Explanation:

The correct answer identifies the IP address (A) and the Host Name (D) as elements usable within the Debug Identification section of Debug Mode for an infrastructure pattern in ServiceNow Discovery.

Let's delve into why:

ServiceNow Discovery utilizes patterns to identify and classify configuration items (CIs) within an IT infrastructure. When troubleshooting issues with pattern execution or identification accuracy, Debug Mode becomes invaluable. The Debug Identification section allows you to specify criteria to focus the debugging efforts on a particular CI.

An IP address is a fundamental network identifier for a device. Specifying an IP in the Debug Identification section narrows down the debug scope to that specific device during pattern execution. Discovery attempts to identify a CI based on this targeted IP.

Similarly, a Host Name serves as a logical name for a device on the network. Using a Host Name in the Debug Identification section allows for targeting a specific device during the debug process.

AWS Endpoint (B) refers to a specific service endpoint in Amazon Web Services (AWS). While Discovery integrates with AWS, specifying a generic AWS Endpoint isn't useful for debugging a specific infrastructure pattern execution related to a CI. Discovery needs a more granular identifier.

PID (C), Process ID, is a dynamic identifier for a process running on a device. While valuable for troubleshooting running processes, it's not a stable identifier for an infrastructure device itself and therefore less useful than a static IP or hostname in identifying a CI during discovery. PID is transient. The Discovery probes identify CIs not running processes.

In summary, IP addresses and Host Names are stable and readily available attributes that directly identify a specific network-connected device within an infrastructure. They provide the necessary targeting capabilities for debugging infrastructure patterns in ServiceNow Discovery.

Further reading:

ServiceNow Discovery documentation: https://docs.servicenow.com/bundle/utopia-it-operations-management/page/product/discovery/concept/c_Discovery.html
ServiceNow Pattern Debugger: https://docs.servicenow.com/bundle/utopia-it-operations-management/page/product/discovery/task/t_RunAPatternDebug.html

Question: 16

With multiple CI data sources, which choice is the best for determining which source can update a CI attribute?

A.Business Rules
B.Data Certification
C.Transform Maps
D.Reconciliation Rules

Answer: D

Explanation:

The correct answer is **D. Reconciliation Rules** because they are specifically designed to manage CI data updates from multiple sources in ServiceNow Discovery. When you have various data sources feeding information about the same Configuration Item (CI), Reconciliation Rules determine which source is authoritative for specific attributes. They allow you to prioritize sources and define criteria for how updates are applied, ensuring data consistency and accuracy in your CMDB.

Reconciliation Rules evaluate attributes coming from different data sources. They apply logic to decide which source wins, based on factors such as source priority, freshness of data, or specific attribute mappings.

Business Rules (A) are more general-purpose automation scripts, not specifically targeted at source prioritization for CMDB updates. Data Certification (B) focuses on validating the accuracy and completeness of CMDB data but doesn't define the update precedence across sources. Transform Maps (C) map incoming data from external sources to CMDB fields, but they don't handle the conflict resolution when multiple sources provide competing information. Reconciliation Rules are thus the most suitable mechanism for defining data source precedence and managing updates in a multi-source CMDB environment, a crucial aspect of maintaining a healthy and reliable CMDB. This is important for dependency mapping, impact analysis, and overall IT service management efficiency. In essence, Reconciliation Rules act as the gatekeepers, determining the definitive source of truth for each attribute of a CI in a multi-source environment.

Further research:

ServiceNow Docs on Reconciliation Rules: https://docs.servicenow.com/bundle/sandiego-cmdb/page/product/configuration-management/concept/c_ReconciliationRules.html

Question: 17

One method for deleting specific CIs not discovered in 30 days is:

A.Scheduled Job
B.UI Policy
C.Service Mapping
D.Data Policy

Answer: A

Explanation:

The correct answer is **A. Scheduled Job**. Here's why:

A **Scheduled Job** in ServiceNow allows you to automate tasks on a recurring basis. To delete CIs not discovered in 30 days, a Scheduled Job can be configured to run periodically, query the CMDB for CIs with a last discovered date older than 30 days, and then delete those records. This provides a robust and automated mechanism for CMDB maintenance.

Here's why the other options are less suitable:

UI Policy: UI Policies affect the behavior and appearance of forms within the user interface. They do not execute background processes to delete data based on age.

Service Mapping: Service Mapping discovers application services and their underlying infrastructure. While related to CI discovery and relationships, it's not directly used for bulk CI deletion based on discovery age.

Data Policy: Data Policies enforce data consistency by setting mandatory fields, read-only states, and other rules when data is created or updated. While they can prevent certain data inconsistencies, they are not designed for deleting records based on age.

Using a Scheduled Job offers several benefits:

Automation: It eliminates manual effort, ensuring consistent CMDB maintenance.
Scalability: It can handle a large number of CIs efficiently.
Customization: The script within the scheduled job can be customized to handle specific CI types or deletion

criteria.
Auditing: Provides a log of when and how CIs were deleted.

Deleting CIs without proper safeguards can have unintended consequences. It's crucial to thoroughly test the scheduled job in a non-production environment and implement appropriate logging and error handling to avoid accidental data loss. Always consider the impact on dependent services and applications before deleting CIs.

For further reading:

ServiceNow Docs - Scheduled Jobs:https://docs.servicenow.com/en-US/bundle/quebec-platform-administration/page/administer/general_administration/concept/c_ScheduledJobs.html
ServiceNow Docs - CMDB:https://docs.servicenow.com/en-US/bundle/quebec-platform-administration/page/product/configuration-management/concept/c_ConfigurationManagementDatabaseCMDB.html

Question: 18

Which of the following choices must be installed on a MID Server to run Credential-less Discovery?

A.Credential-less Extension
B.Nmap
C.Advanced IP Scanner
D.Defender

Answer: B

Explanation:

The correct answer is **B. Nmap**. Here's a detailed justification:

Credential-less Discovery leverages the MID Server's ability to perform network scans and OS fingerprinting without needing specific login credentials for each device. This is achieved through tools like Nmap, which can probe devices and gather information about their operating system, open ports, and running services by analyzing network responses. Nmap is a powerful network scanning tool specifically designed for service discovery and security auditing. It's a core component when employing credential-less discovery methods because it actively probes the network to identify devices and their attributes.

Option A, "Credential-less Extension," is vague and not a standard ServiceNow term related to required MID Server components. Option C, "Advanced IP Scanner," while capable of basic network scanning, is not the primary tool utilized by ServiceNow's Credential-less Discovery. Option D, "Defender," is a security product, and not a requirement to be installed on MID Server.

Nmap relies on packet analysis and OS fingerprinting techniques to gather information, eliminating the need for explicit credentials. For example, it sends carefully crafted packets to target devices and analyzes the responses to determine the OS version or open ports. The MID Server uses the data collected by Nmap to build a comprehensive inventory of network devices and their configurations.

The MID Server then utilizes this information to populate the CMDB (Configuration Management Database) with details about the discovered devices, making it an essential tool for creating and maintaining an accurate and up-to-date IT asset inventory. Therefore, Nmap installation is a crucial prerequisite for successfully executing Credential-less Discovery within a ServiceNow environment.

Further information can be found in ServiceNow documentation on Discovery and MID Servers:

ServiceNow Discovery Documentation
ServiceNow MID Server Documentation

Question: 19

A network device has both an SSH port and an SNMP port open. Discovery tries the SSH probe first and it fails. This triggers the SNMP probe, which succeeds. Discovery uses SNMP first for subsequent discoveries on that device.
What discovery functionality allows the above to happen?

A.Classification
B.Credential affinity
C.MID Server affinity
D.IP service affinity

Answer: D

Explanation:

Here's a detailed justification for why the correct answer is D (IP service affinity) and why the other options are incorrect:

Justification for IP Service Affinity:

IP service affinity is the Discovery functionality that enables ServiceNow to remember the successful protocol (in this case, SNMP) used to connect to a specific IP address. When Discovery initially encounters a network device, it might attempt various probes (SSH, SNMP, WMI, etc.) based on defined classifications and configured credentials. If the SSH probe fails initially but the SNMP probe succeeds, ServiceNow's IP service affinity feature stores this information. Subsequent Discovery runs targeting the same IP address will prioritize the SNMP probe, as it was previously successful. This intelligent behavior optimizes Discovery's efficiency and reduces unnecessary probe attempts. It leverages the knowledge gained from previous successes to improve performance and minimize the risk of repeated failures. This is crucial for devices where one protocol is inherently more reliable or accessible than others. By remembering the successful protocol, Discovery avoids unnecessary delays associated with trying protocols that are likely to fail.

Why other options are incorrect:

A. Classification: Classifications in ServiceNow Discovery determine the type of device being discovered (e.g., server, router, printer). While classifications influence which probes are initially launched, they don't specifically dictate that a successful probe will be preferred in subsequent discoveries. Classifications are more about identifying what the device is, not remembering how to connect to it most efficiently.

B. Credential affinity: Credential affinity associates specific credentials with specific devices or IP ranges.
While important for ensuring the correct credentials are used, it doesn't explain why SNMP would be prioritized over SSH after the initial attempt. Credential affinity dictates which credentials to use, not which protocol to use based on past successes.

C. MID Server affinity: MID Server affinity determines which MID Server is used to perform the Discovery activities for a given IP address range or device. It controls where the discovery happens, not how it happens or which protocols are preferred. Even with MID Server affinity enabled, the system still needs a mechanism to remember the successful protocol, which is provided by IP service affinity.

Authoritative Links for Further Research:

ServiceNow Discovery Documentation: Search the ServiceNow documentation portal (requires a ServiceNow

account) for "IP Service Affinity", "Discovery Probes", "Discovery Classifications", "Credential Affinity", and "MID Server Affinity" to find the most up-to-date and detailed information.

ServiceNow Community Forums: Search the ServiceNow Community forums for discussions and examples related to Discovery and IP service affinity. Other users' experiences can provide valuable insights.

Question: 20

The CMDB contains which of the following record types? (Choose two.)

A.Model
B.Configuration Item (CI)
C.Asset
D.Relation Type

Answer: BD

Explanation:

The correct answer to the question of which record types the CMDB contains is **B. Configuration Item (CI)** and **D. Relation Type**. The CMDB (Configuration Management Database) is fundamentally designed to store information about configuration items (CIs) within an organization's IT infrastructure. These CIs represent tangible and intangible components, such as servers, software applications, network devices, services, and even personnel. Each CI is a record within the CMDB, holding attributes specific to that item.

Relation Types are crucial because they define the relationships between different CIs. The CMDB doesn't just store standalone items; it captures how these items connect and depend on each other. For example, a relation type might indicate that a server hosts a specific application, or that a network switch connects to a database. These relationships are vital for impact analysis, change management, and incident resolution. Without relation types, the CMDB would be a mere inventory list, lacking the necessary context for effective IT service management.

While Assets (C) and Models (A) are related concepts, they are not the primary record types contained within the CMDB. Assets represent the financial and contractual aspects of a CI (ownership, lease information, depreciation), while Models represent a definition or template for a type of CI. Often Asset information is linked to a CI record; likewise, a CI record is linked to a particular model. The CI itself is the core element, with Assets and Models providing supplemental data.

The primary purpose of a CMDB is to hold CIs and the relationships between them, and the CMDB is at the center of other critical service management processes, such as incident management, problem management, change management, and service request management.

For further reading, consult the official ServiceNow documentation on Configuration Management and the CMDB:

ServiceNow CMDB Overview:https://www.servicenow.com/content/dam/servicenow/other-documents/service-brief/sb-cmdb.pdf
ServiceNow Documentation:https://docs.servicenow.com/bundle/utopia-release/page/product/configuration-management/concept/cmdb-concept.html

Question: 21

When is the Extension section in a horizontal pattern executed?

A.As part of the post sensor processing script
B.After the Identification sections
C.As part of the port scan
D.Before the Identification sections

Answer: B

Explanation:

The correct answer is **B. After the Identification sections.** Here's a detailed justification:

Horizontal Discovery patterns in ServiceNow are structured processes designed to identify and classify Configuration Items (CIs) within an environment. These patterns progress through distinct phases, each with a specific purpose. The Identification section focuses on matching existing CIs in the CMDB or creating new ones based on the discovered data. It's crucial to establish the identity of the device or application before enriching its data or making further customizations.

The Extension section builds upon the identified CI. It allows you to extend the discovered information about a CI by running custom commands or scripts, accessing specific data, or performing additional checks. These extensions are intended to add value after the core identification and classification are complete. They might fetch application-specific details, OS-level configurations, or any other data relevant to the identified CI that wasn't gathered in the initial identification phase.

Because the extension section aims to add more information based on a uniquely identified CI, it must execute after the Identification section has successfully determined the CI's identity. Running extensions before identification would be illogical, as they often rely on properties or attributes established during the identification phase.

The Post Sensor processing script (option A) is usually executed after all other sections of the pattern, including both Identification and Extension. The Port scan (option C) is often part of the Discovery schedule or probes, before the pattern is even triggered, while the Identification and Extension sections operate as part of a pattern. Option D is incorrect because extensions need identified CIs to be useful. They cannot operate on unidentified entities.

In summary, the Extension section's role is to enhance the data of an already-identified CI, making its execution logically dependent on the successful completion of the Identification section.

For further reading:

ServiceNow Discovery Documentation: https://docs.servicenow.com/bundle/utopia-servicenow-platform/page/product/discovery/concept/c_Discovery.html
Horizontal Pattern Documentation: Consult ServiceNow's pattern documentation for detailed workflow. While specific pages on extensions alone might be limited, the overall Discovery documentation helps establish the context. (Requires ServiceNow login to access specific instance documentation).

Question: 22

For CMDB Health, relationships can be which of the following choices? (Choose three.)

A.Duplicate
B.Stale
C.Orphan
D.Required
E.Recommended

Answer: ABC

Explanation:

The correct answer is indeed A, B, and C: Duplicate, Stale, and Orphan. These represent key relationship health aspects within ServiceNow's CMDB Health module.

Let's break down why:

Duplicate Relationships: These occur when the same relationship exists multiple times between two CIs. This could happen due to data import errors, integration issues, or manual creation. Duplicates introduce redundancy, confusion, and can lead to inaccurate dependency mapping and impact analysis. CMDB Health identifies these to ensure relationship integrity.

Stale Relationships: Stale relationships are those that are no longer valid. A device might have been decommissioned but the "Connected to" relationship pointing to that device from another CI is still present in the CMDB. These relationships lead to incorrect information, creating problems for change management and incident resolution. CMDB Health flags these as stale based on defined age criteria and reconciliation rules.

Orphan Relationships: Orphan relationships occur when one CI in a relationship is missing (e.g., deleted or retired). Consider CI A 'Connected to' CI B. If CI B is removed from the CMDB but the relationship on CI A isn't updated, the relationship on CI A becomes orphaned. These orphaned relationships can cause broken dependencies and negatively affect impact analysis. CMDB Health helps identify these discrepancies.

Options D (Required) and E (Recommended) are related to configuration management best practices, but are not direct outputs concerning relationship validity within CMDB Health's core reporting metrics. CMDB Health uses defined rules and reconciliation processes to identify and report on the health of relationships across various layers in the CMDB. CMDB health dashboard presents relationship data like completeness, accuracy, and compliance. Correcting health deficiencies ultimately leads to faster incident resolution, reduced downtime, and improved operational efficiency.

For further research, refer to the following ServiceNow documentation links:

CMDB Health Overview
CMDB Health Dashboard
Configure CMDB Health

Question: 23

Which of the choices provides active discovery errors with a help link for each error?

A.Discovery Dashboard
B.IP Address Failure Report
C.Discovery Schedule
D.MID Server Dashboard

Answer: A

Explanation:

The correct answer is A, the Discovery Dashboard. Here's why:

The Discovery Dashboard is specifically designed to provide a comprehensive overview of the Discovery process, including errors and warnings encountered during scans. One of its key features is presenting active Discovery errors directly. Crucially, it includes help links associated with each error. These links provide

immediate access to documentation and troubleshooting steps, enabling administrators to quickly understand the root cause of the error and implement corrective actions. This proactive approach to error management ensures the efficiency and accuracy of Discovery operations.

While the IP Address Failure Report (B) provides information about IP addresses that failed to be discovered, it doesn't necessarily offer the help links associated with the errors. Discovery Schedule (C) is used to configure and manage Discovery schedules and does not provide insights into active errors with remediation links. The MID Server Dashboard (D) primarily focuses on the status and health of MID Servers, although it might show some connection-related errors, it isn't the primary location for viewing and troubleshooting general Discovery errors along with help links. The Discovery Dashboard provides a centralized and actionable view of active Discovery errors with readily accessible support resources, making it the optimal choice.

ServiceNow Discovery DocumentationServiceNow Discovery Dashboard Documentation

Question: 24

What related list on a classifier dictates which Horizontal Pattern probe is launched?

A.Discovery Log
B.Classification Criteria
C.Pattern probes
D.Triggers probes

Answer: D

Explanation:

The correct answer is **D. Triggers probes**.

Triggers on a classifier determine which probe, including Horizontal Pattern probes, are launched during Discovery. Classifiers are designed to identify the CI type of a discovered device. When a classifier matches a device, the trigger probes defined within that classifier's "Triggers" related list are executed. These triggers specify which probes are launched to gather more detailed information about the CI, and they are associated with specific conditions that need to be met before the probe is launched. Horizontal Pattern probes are launched based on these triggers, allowing for targeted data collection.

The "Discovery Log" (A) primarily records the events and results of the discovery process itself and does not directly dictate probe launches. "Classification Criteria" (B) defines the conditions that must be met for the classifier to be considered a match, but it does not contain the probe launch specifications. "Pattern probes" (C) isn't a standard related list on a classifier. Pattern probes are launched by triggers, not defined within a related list named "Pattern probes." Triggers, therefore, are the key element that directly links classifiers to the specific probes, including horizontal patterns, that are launched during the discovery process to populate the CMDB. This mechanism ensures that only the relevant probes are executed, optimizing discovery performance and ensuring data accuracy.

For further research, explore the ServiceNow documentation on Classifiers and Triggers:

ServiceNow Documentation - Classifiers
ServiceNow Documentation - Discovery Probes

Question: 25

Which of the following does the ECC Queue provide? (Choose two.)

A.Login credentials for the MID Server host.

B.The actual XML payload that is sent to or from an instance.

C.A connected flow of probe and sensor activity.

D.The process responsible for defining, analyzing, planning, measuring, and improving all aspects of the availability of IT services.

Answer: BC

Explanation:

The ECC Queue (External Communication Channel Queue) in ServiceNow Discovery plays a crucial role in the communication between the ServiceNow instance and the MID Server. It acts as a central hub for all communication related to Discovery and Orchestration.

Option B is correct because the ECC Queue entries store the actual XML payload that is exchanged between the ServiceNow instance and the MID Server. This payload contains the instructions for the MID Server (e.g., to run a specific probe) and the data that the MID Server returns to the instance (e.g., the results of the probe). Analyzing these XML payloads is crucial for troubleshooting Discovery issues.

Option C is also correct. The ECC Queue maintains a connected flow of probe and sensor activity. Each entry in the queue represents either a probe being sent out to the MID Server or a sensor processing data returned by the MID Server. The correlation ID field within the ECC Queue entries links these probes and sensors together, creating a traceable chain of activity. This traceability is essential for understanding how data is collected and processed during Discovery. You can follow the entire sequence of events from the initial probe to the final data being inserted into the CMDB.

Option A is incorrect because login credentials for the MID Server host are not stored in the ECC Queue. MID Server credentials are saved securely in the ServiceNow instance's credential store and are only referenced by the MID Server configuration.

Option D is incorrect because it describes Availability Management, which, while important in IT service management, is a distinct process from the ECC Queue's function in facilitating probe and sensor communication for Discovery.

In summary, the ECC Queue provides visibility into the raw XML data being exchanged and a traceable flow of probe and sensor activity, making it essential for debugging and understanding Discovery processes.

Authoritative links:

ServiceNow Docs - ECC Queue: https://docs.servicenow.com/bundle/sandiego-platform-administration/page/administer/ecc_queue/concept/c_TheECCQueue.html
ServiceNow Community - Understanding the ECC Queue: (Search ServiceNow Community for relevant articles)

Question: 26

Which of the following choices are only used for the Application Pattern Type? (Choose two.)

A.Run Order

B.Identification Section

C.CI Type

D.Operating System

Answer: AD

Explanation:

The correct answer is A and D: Run Order and Operating System. Here's why:

Application Patterns: Application Patterns in ServiceNow Discovery are designed for in-depth discovery and mapping of complex applications and their dependencies. They go beyond basic CI identification, focusing on understanding how applications function.

Run Order (A): The "Run Order" field within an Application Pattern defines the sequence in which the steps within the pattern are executed. This is crucial for complex application discovery because the order of execution can significantly impact the data gathered. For example, you might need to identify a database server before attempting to discover databases residing on it. This level of control is primarily relevant to Application Patterns, as basic discovery methods typically do not require such granular step sequencing. The CMDB Query Builder in Service Mapping relies on Run Order. https://docs.servicenow.com/bundle/utah-servicenow-platform/page/product/service-mapping/concept/cmdb_query_builder.html

Operating System (D): While the Operating System is a standard CI attribute, it plays a specific and vital role in Application Patterns. Patterns often need to execute commands or scripts tailored to a particular OS (e.g., Windows, Linux). Therefore, the Operating System is a key parameter used to determine which specific steps within the pattern should be executed. This OS-specific behavior is much more critical in Application Patterns where precise execution is needed. For instance, to discover the application software, different commands may be used on the Windows operating system vs. the Linux operating system.

https://docs.servicenow.com/bundle/utah-servicenow-platform/page/product/discovery/concept/c_ApplicationDiscovery.html

Identification Section (B): The Identification Section is used in all discovery methods, not just application patterns. It determines how Discovery identifies unique CIs.

CI Type (C): CI Type is a fundamental attribute of any CI discovered in ServiceNow, regardless of the discovery method used (Basic, Horizontal, or Application Pattern). It defines what kind of CI it is (e.g., server, database, application).

In summary, Run Order is key to sequencing steps within a pattern and Operating System is important for determining OS-specific commands in Application Patterns.

Question: 27

By default, which of the following are automatically available as variables for horizontal discovery patterns? (Choose two.)

A.infrastructure_system
B.The CI Type on the Discovery Pattern form
C.windows_cmdb_ci
D.computer_system

Answer: BD

Explanation:

Here's a breakdown of why options B and D are correct in the context of ServiceNow's Horizontal Discovery patterns and variables, along with justifications and supporting links:

Why B (The CI Type on the Discovery Pattern form) is correct:

Horizontal Discovery patterns are designed to identify and classify configuration items (CIs) within your

infrastructure. A core element of a pattern is specifying which type of CI the pattern is intended to discover (e.g., Windows Server, Linux Server, Database Instance). The CI Type selected in the pattern definition directly influences the available variables. The system inherently understands which attributes and relationships are relevant based on this CI Type. Selecting "Windows Server" makes properties specific to Windows Servers available as variables. This selection is fundamental to the discovery process and dictates what the system can expect and manipulate.

Why D (computer_system) is correct:

computer_system is a common, foundational CI class within the CMDB, often inheriting from cmdb_ci. Discovery patterns, particularly those related to servers and workstations, often involve gathering information related to the underlying computer_system aspects of a device. Even if you're discovering a more specific CI type (e.g., a specific application), having access to variables related to the computer system it's running on is crucial for comprehensive information. These variables could include CPU information, memory, disk space, and operating system details. This access reflects the hierarchical nature of the CMDB and the importance of understanding the infrastructure on which services run.

Why A (infrastructure_system) is incorrect:

While 'infrastructure_system' might seem like a plausible variable name, it's not a standardized, automatically available variable for all Discovery patterns. The availability of such a variable would depend on custom configurations and potentially, extensions to the base ServiceNow platform. It's not provided by default across all Discovery pattern setups.

Why C (windows_cmdb_ci) is incorrect:

The windows_cmdb_ci prefix in the proposed variable name seems somewhat redundant. When defining a pattern for a specific CI type such as a Windows Server, the CI type from the pattern definition (option B) is already associated with the Windows CI type (e.g., cmdb_ci_win_server), providing a context for the discovery process. Also the table name is cmdb_ci_win_server, the variable is not named as windows_cmdb_ci. The system knows that you're discovering something Windows-related through that CI type selection. The variable names used depend on specific properties and configurations of the pattern and may not align with table name.

Supporting Links:

ServiceNow Docs - Horizontal Discovery:https://docs.servicenow.com/bundle/utah-it-operations-management/page/product/discovery/concept/c_HorizontalDiscovery.html
ServiceNow Docs - Discovery Patterns:https://docs.servicenow.com/bundle/utah-it-operations-management/page/product/discovery/concept/c_DiscoveryPatterns.html
ServiceNow Community - Discovery:https://community.servicenow.com/community?id=community_topic&sys_id=8673b21ddbd890d068c11485ca961955

Question: 28

Which choices are necessary to launch any pattern? (Choose two.)

A. CI Classification
B. CI Serial Number Attribute
C. Data Certification
D. CI Type

Answer: AD

Question: 29

Which of the below choices are horizontal pattern types? (Choose two.)

A.Hardware
B.Software
C.Infrastructure
D.Application

Answer: CD

Explanation:

The correct answer is C. Infrastructure and D. Application because they are both examples of horizontal discovery patterns in ServiceNow Discovery. Horizontal discovery patterns aim to discover and map infrastructure and application resources across an organization's environment. These types of resources can be deployed on various layers of the infrastructure.

Infrastructure patterns identify the underlying physical and virtual infrastructure components, such as servers, network devices, storage arrays, and cloud resources. These patterns help to establish the foundation of the CMDB and allow for visibility of the environment's physical footprint.

Application patterns discover and map the applications running on top of the infrastructure, along with their dependencies and configurations. They allow administrators to understand the application landscape and can provide insight into performance, configuration management, and license tracking. Application patterns often leverage techniques like process discovery, configuration file parsing, and database querying to gather information about applications.

Hardware and Software, on the other hand, while important for IT management, are not typically categorized as horizontal patterns. Hardware discovery is more granular and focused on device details, while Software discovery is more general and encompasses a wider range of software aspects.

Therefore, when it comes to the horizontal application of discovery patterns to cover a broad range of resources for a more foundational configuration management database (CMDB), infrastructure and application resources are the correct answer.

For further reading and research, refer to these authoritative resources:

1. ServiceNow Discovery documentation: https://docs.servicenow.com/bundle/utah-it-operations-management/page/product/discovery/concept/c_Discovery.html
2. ServiceNow Community forums: https://community.servicenow.com/ (Search for "Discovery patterns")

Question: 30

What does the MID Server need to collect vCenter events?

A.vCenter Event Collector extension
B.MID SNMP Trap Listener extension
C.Firewall
D.vCenter probes

Answer: A

Explanation:

The correct answer is A, vCenter Event Collector extension. Here's why:

The ServiceNow MID Server is the bridge between your on-premises or private cloud infrastructure (like vCenter) and your ServiceNow instance. To collect vCenter events, the MID Server needs the capability to communicate with the vCenter server and understand its event data. This is achieved through a specific extension tailored for vCenter events.

vCenter Event Collector extension: This extension contains the necessary scripts, probes, and sensors to communicate with the vCenter API, retrieve event data, and transform it into a format understandable by ServiceNow. It understands the nuances of vCenter's event structure.

MID SNMP Trap Listener extension: This extension is primarily used for receiving SNMP trap notifications, which are a different type of event notification mechanism, not typically used for comprehensive vCenter event monitoring.

Firewall: While a firewall configuration is essential for connectivity, the firewall itself is not the component responsible for data collection or translation. The firewall must allow the MID Server to communicate with the vCenter server on the necessary ports, but it's the Event Collector that handles the event information.

vCenter probes: While probes are involved in discovery and data collection, simply having "vCenter probes" isn't sufficient for event collection. The Event Collector uses probes, but it's the overall extension that provides the complete functionality. The collector configures and manages these probes for event-specific tasks.

In essence, the vCenter Event Collector extension is a specialized package designed specifically to extract event information from vCenter. It utilizes the vCenter API to subscribe to, retrieve, and process events, ultimately enabling ServiceNow to respond proactively to changes in the virtualized environment. Without this extension, the MID Server wouldn't "know" how to interpret vCenter events, and they wouldn't be properly captured in ServiceNow.

Further Research:

ServiceNow Docs - MID Server Extensions: https://docs.servicenow.com/bundle/vancouver-servicenow-platform/page/product/mid_server/concept/mid-server-extensions.html
ServiceNow Community Forums (search for "vCenter Event Collector"): https://community.servicenow.com/

Question: 31

A Discovery Schedule contains a /24 subnet IP Range and a Shazzam batch size of 5000. How many times will a Shazzam probe be launched during discovery?

A.1
B.2
C.5000
D.254

Answer: A

Explanation:

The question concerns the number of Shazzam probes launched by a ServiceNow Discovery schedule targeting a /24 subnet with a batch size of 5000. A /24 subnet encompasses 256 IP addresses (2^(32-24)). However, because network addresses and broadcast addresses are usually unusable, that leaves 254 usable

addresses in that range.

Shazzam is the probe responsible for identifying network devices during Discovery. The batch size determines how many IP addresses are sent to the Shazzam probe at once.

Since we have 254 usable IP addresses to scan, and the Shazzam batch size is 5000, a single Shazzam probe is sufficient to cover the entire /24 subnet. The probe will be launched only once, even though it only requires probing 254 IPs as it can handle up to 5000 in a batch.

Therefore, the correct answer is A. 1.

Authoritative Links:

ServiceNow Discovery: (Unfortunately, direct links to specific ServiceNow documentation are behind a login. Access ServiceNow documentation through your ServiceNow instance's help resources, searching for "ServiceNow Discovery", "Shazzam", and "Discovery Schedules")

Subnetting Explanation:https://www.cloudflare.com/learning/network-layer/what-is-a-subnet/ (Provides general networking subnetting concepts)

Question: 32

Which method is used by Discovery to determine if a Host IP is active or alive?

A.Port Scan
B.Traceroute
C.Ping
D.Classification

Answer: A

Explanation:

The correct answer is A. Port Scan. Here's a detailed justification:

Discovery uses various methods to determine if a host is alive and responsive, and a port scan is a crucial part of this process. While ping (ICMP Echo Request) can verify network connectivity, it's not the definitive method for determining if a host is truly active and providing services. Many firewalls and security devices block ICMP traffic, leading to false negatives. Traceroute maps the network path to a host but doesn't reliably indicate if the host is actively listening on ports. Classification occurs after a device is discovered and identified.

A port scan, on the other hand, actively attempts to connect to common ports on the target host (e.g., port 22 for SSH, port 80 for HTTP, port 443 for HTTPS). If a port responds, it signifies that a service is listening on that port, thus confirming the host's active state and the presence of running applications. Discovery leverages this information to not only identify the active status but also gather information on what services are running on the target. Successful port connections provide concrete evidence of an active service, whereas a filtered or closed port implies either a service is unavailable or a firewall is blocking access.

ServiceNow utilizes port scans to populate the CMDB (Configuration Management Database) with more detailed information about the discovered devices, going beyond simply verifying network reachability. This detailed service-level visibility helps in asset management and incident troubleshooting. This approach provides more accurate results than simply relying on ping, as it provides information about running services, which are key indicators of an active and functioning host.

Therefore, the combination of connectivity tests (like ping, although not the primary method for determining activity) along with port scanning provides a more comprehensive and reliable assessment of host activity for

Discovery. Port scanning is considered the most definitive initial assessment of a host's 'aliveness' from a functional perspective, making it the right answer.

Authoritative links for further research:

ServiceNow Discovery Documentation: Look for details on "Discovery probes and sensors" and "Horizontal Discovery" on the official ServiceNow documentation portal: https://docs.servicenow.com/ (Requires ServiceNow account access)
Network Port Scanning Techniques: Research general network port scanning concepts and techniques.

Question: 33

Discovery finds and maps dependencies for the following types of storage devices. (Choose three.)

A.Direct-attached storage (DAS)
B.Network-attached storage (NAS)
C.Storage area network (SAN)
D.Multiple area network (MAN)
E.Redundant Array of Independent Disks (RAID)

Answer: ABC

Explanation:

The correct answer identifies the types of storage devices that ServiceNow Discovery typically maps dependencies for: Direct-Attached Storage (DAS), Network-Attached Storage (NAS), and Storage Area Network (SAN).

Direct-Attached Storage (DAS): DAS refers to storage directly connected to a server, like an internal hard drive or an external USB drive. Discovery can identify DAS devices and their relationship to the servers they are attached to. The dependency mapping helps understand the impact of a server outage on the data stored on its DAS devices.

Network-Attached Storage (NAS): NAS devices are file-level storage servers connected to a network, allowing multiple devices to access files from a central location. Discovery identifies NAS devices on the network, their configurations (like shared folders), and the servers/clients accessing them. This reveals which applications and services rely on the NAS for data storage and retrieval.

Storage Area Network (SAN): SAN is a dedicated high-speed network connecting servers to storage devices, providing block-level access. Discovery can map the SAN infrastructure, including the switches, storage arrays, and the servers connected to them. Understanding SAN dependencies is crucial for identifying bottlenecks, ensuring data availability, and preventing service disruptions.

Multiple Area Network (MAN): MAN is a larger network than LAN but smaller than WAN, generally covering a city. While network connectivity is discovered, the specific dependency mapping related to the storage characteristics implied in the other choices is outside its scope.

Redundant Array of Independent Disks (RAID): RAID is a data storage virtualization technology that combines multiple physical disk drive components into one or more logical units for data redundancy, performance improvement, or both. RAID is a configuration option found in DAS, NAS, and SAN systems, rather than a distinct type of storage discoverable at a dependency mapping level. Discovery may identify RAID configuration details within DAS/NAS/SAN, but it doesn't treat "RAID" itself as a standalone storage device type for dependency mapping in the same way.

In essence, Discovery focuses on identifying and mapping dependencies based on the architectural types of

storage (DAS, NAS, SAN) and their connections to servers and applications. It maps where data resides and who is accessing it. While the other options are relevant within broader IT environments, they don't represent distinct storage entities for dependency mapping in the same way as DAS, NAS, and SAN.

Further research:

ServiceNow Documentation on Discovery: https://docs.servicenow.com/ (Search for "Discovery" and "Storage Discovery")

ServiceNow Community Forums: https://community.servicenow.com/ (Search for "Discovery Storage Mapping")