Boost up Your Certification Score

UiPath

UiPath-TCAPv1

UiPath Test Cloud Architect Professional Exam



For More Information – Visit link below:

https://www.examsboost.com/

Product Version

✓ Up to Date products, reliable and verified.
✓ Questions and Answers in PDF Format.

Visit us at: https://www.examsboost.com/test/uipath-tcapv1

Latest Version: 6.0

Question: 1

A test architect is designing a parameterized test case in Test Manager and must enforce uniqueness constraints across parameter definitions.
Within what documented scope must a parameter name be unique?

A. A parameter name must be unique within the context of a specific test case, meaning two parameters in the same test case cannot share the same name.
B. A parameter name must be unique across all test cases within the same test set, meaning no two test cases in the same test set can define a parameter with the same name.
C. A parameter name must be unique at the project level, meaning the same parameter name cannot be reused across any test case within the entire Test Manager project.
D. A parameter name must be unique within the Parameters tab of the test case only when the parameter is referenced in manual steps using the double curly bracket notation.

Answer: A

Explanation:
UiPath Test Manager supports parameterized test cases by allowing users to define reusable input values that can be referenced during manual test execution.
Option A correctly identifies the documented uniqueness constraint: each parameter name must be unique within its individual test case.
The requirement does not extend across an entire test set or Test Manager project. Consequently, different test cases can define parameters with identical names without violating the uniqueness constraint.
Parameters are created through the Parameters tab of an individual test case. Each parameter contains a name, an optional default value, and an optional hint describing its intended purpose.
UiPath supports a maximum of 250 parameters per test case, and manually created Test Manager parameters use the String data type.
During manual execution, a parameter can be referenced in a test step using double curly brackets, such as {{username}}. This syntax allows the configured value to be substituted during execution.
Option B incorrectly extends uniqueness to the test-set level. Option C imposes an unsupported project-wide restriction. Option D incorrectly makes uniqueness dependent on whether a parameter is referenced in manual steps.
Important distinction: Parameters created in the test case's Parameters tab are intended for manual executions. Automated test execution uses Studio arguments exposed through Test Manager's test-set-level parameter override functionality.
Official UiPath Reference(s): Test Manager User Guide — Parameters; Creating parameters.

Question: 2

Visit us at: https://www.examsboost.com/test/uipath-tcapv1

A developer needs to validate the same API sequence with multiple input combinations and cover several scenarios in a single test project.
Which option best aligns with the recommended approach?

A. Create a Data Driven Test Case, import the data set, and use the mapped arguments within the activity
B. Create an individual local Swagger file for each variation and load them separately, one at a time
C. Create a distinct namespace for every scenario and execute them through the Service Editor
D. Create a separate library for each data set and publish each library independently

Answer: A

Explanation:
UiPath Studio supports data-driven API test automation, enabling developers to execute the same sequence of API operations using multiple input combinations.
Option A is correct because a Data Driven Test Case separates test logic from the data used during each test execution.
A typical API testing implementation begins by importing the required API definitions into Studio through the New Service wizard. The imported service operations become available as activities that can be incorporated into a Test Automation project.
Developers can then create a Data Driven Test Case and import a dataset containing the required input variations.
The imported values are mapped to workflow arguments, allowing each variation to supply different parameters to the same API activities.
For example, a loan application API may need validation using different customer identifiers, loan amounts, repayment periods, and expected response conditions.
Rather than duplicating the API workflow, developers can execute the same sequence for each dataset variation.
Assertions implemented through UiPath Testing Activities verify whether the actual responses meet the expected conditions.
Option B unnecessarily duplicates API definitions. Option C confuses namespaces with test scenario management. Option D introduces redundant libraries without improving test coverage.
The recommended approach improves maintainability, execution consistency, and the systematic validation of API behavior across multiple scenarios.
Official UiPath Reference(s): Studio User Guide — API Test Automation; Load Library in API Test Project; data-driven testing example.

Question: 3

How do you create a local template in UiPath?

A. By publishing a template to the Orchestrator and then downloading it to the Templates folder
B. By directly saving a workflow in the Templates folder
C. By importing a template from an external source into the Templates folder
D. By extracting a file as a template or by manually adding a template to the Templates folder

Answer: D

Explanation:
UiPath Studio provides local file templates to standardize the structure of workflows and test cases within an automation project. These templates enable developers to reuse predefined workflow structures without recreating common activities and configurations.
Option D correctly identifies the two documented methods for creating local templates.
The first method is to extract an existing file as a template. Developers locate the workflow or test case in Studio's Project panel, open its context menu, and select Extract as Template. Studio creates the corresponding template within the project's Templates folder.
The second method is to add a template manually. Developers right-click the Templates folder and select the appropriate template type from the Add menu.
UiPath supports three local template categories: Workflow Template, Test Case Template, and Execution Template.
These templates are available within the project in which they are created. When developers create new files, they can select an appropriate existing template as the starting point.
Option A incorrectly introduces an Orchestrator publishing requirement. Option B does not describe the supported template-creation workflow. Option C is not one of the two standard creation methods.
Local templates improve maintainability, encourage consistent implementation patterns, and reduce repetitive configuration in test automation projects.
Official UiPath Reference(s): Studio User Guide — Local File Templates; Test Case Templates.

Question: 4

What is the main role of connectors in UiPath Integration Service?

A. Provide prebuilt integrations that expose external application capabilities as actions, triggers, and data models for automation
B. Manage execution queues and schedules to ensure external system interactions are processed in a controlled order
C. Define custom API schemas that allow developers to manually configure endpoints and authentication flows for each integration
D. Store automation assets and credentials that can be reused across workflows interacting with external services

Answer: A

Explanation:
UiPath Integration Service provides a centralized integration framework that allows automations to interact with external applications through predefined API integrations.
Option A is correct because connectors expose supported external application operations through reusable activities, event triggers, and associated application data structures.
Integration Service provides a catalog of connectors for supported enterprise applications. Each connector defines how UiPath communicates with the corresponding service and exposes available operations to automation developers.

Connections establish the authenticated relationship with the external application. Once the connection is configured, developers can use connector-specific activities in Studio to perform operations such as retrieving records, creating objects, updating data, or accessing external application resources.
Event triggers enable automations to respond to events occurring in connected systems. Depending on the connector, these events may include the creation or modification of records.
Option B describes execution orchestration capabilities more closely associated with Orchestrator.
Option C focuses on manually developing API integrations rather than the primary purpose of prebuilt connectors. Option D describes credential and configuration management, not connector functionality.
Architecturally, connectors separate external API integration details from business automation logic, reducing the need to implement repetitive authentication and HTTP communication procedures.
Official UiPath Reference(s): Integration Service User Guide — Introduction; Connectors; Triggers.

Question: 5

A Test Manager user opens a generated report and wants to investigate systemic issues instead of individual messages.
Which section should the user review first to identify broader recurring themes behind failed test cases?

A. Common Errors, because it groups similar error messages semantically and is intended to reveal broader systemic categories.
B. Recommendations, because it guides next steps and is intended to categorize failed test cases into broader issue groups.
C. Error Patterns, because it categorizes failed test cases into broader categories that reveal recurring themes and systemic problems.
D. Top Failing Tests, because it lists frequently failed test cases and is intended to classify them into recurring patterns.

Answer: C

Explanation:
UiPath Autopilot supports test execution analysis by generating reports that identify recurring failures, categorize execution problems, and recommend improvements to test stability.
Option C is correct because the Error Patterns section specifically identifies broader categories of failures that may indicate systemic issues.
Autopilot-generated test reports can contain five sections: Overview, Top Failing Tests, Common Errors, Error Patterns, and Recommendations.
Each section provides a different analytical perspective. Common Errors groups similar error messages to identify frequently occurring exceptions or messages. Error Patterns, however, categorizes failures into broader groups that reveal recurring themes and underlying problems.
For example, several failed test cases may produce different exception messages while sharing a common synchronization or application-availability problem. Error Patterns helps testers recognize this relationship rather than investigating each message independently.
Top Failing Tests identifies test cases with frequent execution failures, while Recommendations provides actions intended to improve testing stability.
Option A incorrectly assigns broader systemic categorization to Common Errors. Option B confuses recommendations with classification. Option D incorrectly describes Top Failing Tests as a failure-pattern analysis mechanism.

The appropriate analytical sequence is to examine Error Patterns, investigate the associated test case logs, and then review Recommendations for suitable corrective actions.
Official UiPath Reference(s): Autopilot User Guide — Generate test reports; Example prompts for test result analysis.

Question: 6

What is the purpose of setting the Clipboard data field in the manual steps of a test case?

A. Holding the expected state of the application after the test case was executed
B. Outlining the tasks that need to be completed before the start of testing
C. Eliminating the need for manual entry into input fields
D. Saving the output of the test case for future use

Answer: C

Explanation:
The Clipboard data field in UiPath Test Manager simplifies manual test execution by allowing predefined input values to be transferred to the operating system clipboard.
Option C is correct because this functionality eliminates the need for testers to type specified input values manually into application fields.
When defining manual test steps, testers can specify clipboard content alongside the step description and expected behavior. During manual test execution, the configured value is made available through the Windows clipboard, enabling the tester to paste it directly into the target application's input field.
This capability is particularly useful when testing applications requiring lengthy identifiers, complex input strings, customer reference numbers, or structured data values.
For example, a test step requiring a specific account identifier can provide that identifier through Clipboard data, reducing transcription errors and ensuring consistent test inputs.
The feature improves manual testing efficiency without converting the manual interaction into an automated UI activity. The tester remains responsible for pasting the data and evaluating the application response.
Option A describes expected behavior or postconditions. Option B describes preconditions. Option D concerns output retention, which is unrelated to clipboard-based input preparation.
Clipboard data therefore serves as an input assistance mechanism rather than an assertion, prerequisite definition, or persistent test results repository.
Official UiPath Reference(s): Test Manager User Guide — Copying clipboard values; Adding manual steps to a test case.

Question: 7

A test lead wants to rerun two failed test cases individually from a test set execution, but the option is not available.
What is the most likely reason?

A. The test set is linked from Orchestrator, which restricts execution to full test set runs and does not support individual test case execution from within Test Manager.

B. The execution type has changed to Mixed due to a combination of manual and automated test cases, which disables individual test case selection until the full execution is completed.
C. The two test cases are automated, and individual automated execution is only supported through a pending execution created before the test set is initially triggered.
D. Execution order is enabled on the test set, which prevents individual test cases from being executed independently within that execution.

Answer: D

Explanation:
UiPath Test Manager supports individual test case re-execution, allowing test engineers to rerun failed scenarios without executing the entire test set. However, this capability is subject to the execution-order configuration.
Option D is correct because enabling Enforce Execution Order prevents individual test cases from being executed independently within the ordered test set.
When execution order is enforced, Test Manager preserves the predefined test sequence to maintain execution consistency. Allowing isolated re-execution could produce misleading results if tests depend on intermediate states established by earlier cases.
UiPath explicitly identifies enforced execution order and individual test case re-execution as mutually exclusive capabilities.
To resolve this limitation, the test lead should disable Enforce Execution Order for the test set and then navigate to the relevant execution results to rerun the failed test cases.
Option A is incorrect because individual execution is supported for eligible test sets linked from Orchestrator. Option B is incorrect because mixed manual and automated execution does not inherently prevent individual execution. Option C incorrectly restricts individual automated execution to previously created pending executions.
As an architectural best practice, independently executable test cases should avoid unnecessary dependencies on preceding test cases.
Official UiPath Reference(s): Test Manager User Guide — Re-running only failed test cases; Executing tests.

Question: 8

An organization wants Azure DevOps to authenticate to UiPath Test Cloud in a fully automated, headless CI/CD process that requires no user interaction.
Which external application design best meets that requirement?

A. Register a public OAuth application with user scopes enabled so the pipeline can prompt for sign-in whenever it needs to access Test Cloud.
B. Register a confidential OAuth application with application scopes enabled so the pipeline can store a secret securely and act on behalf of the app itself.
C. Register a confidential OAuth application with only user scopes so the pipeline can inherit permissions from whichever developer triggered the run.
D. Register a public OAuth application with no secret so the pipeline can authenticate independently through the repository connection alone.

Answer: B

Explanation:
Headless CI/CD integration requires an authentication mechanism that operates independently of interactive user sessions. UiPath supports this architecture through confidential external OAuth applications configured with application scopes.
Option B is correct because confidential applications can authenticate using the OAuth 2.0 client credentials grant.
In this configuration, administrators register an external application and assign the application scopes required to access specific UiPath services.
The registration provides a client identifier and client secret. Azure DevOps can securely store these credentials using protected pipeline variables or an appropriate secrets-management service.
During pipeline execution, the automation requests an access token from the UiPath identity service using the client credentials flow. The resulting token represents the application identity rather than an individual user.
This approach supports unattended build, deployment, and testing operations without browser-based authentication or developer interaction.
Option A requires interactive authentication. Option C relies on user scopes, which require user-context authorization. Option D incorrectly assumes that a public application and repository connection provide independent service authentication.
For production implementation, administrators should apply least-privilege scopes, protect client secrets, manage credential rotation, and ensure that the external application has the necessary resource-level permissions.
Official UiPath Reference(s): Automation Cloud Admin Guide — Managing external OAuth applications; Orchestrator User Guide — Authenticating with an external application.

Question: 9

A test case runs and throws an exception.
Which strategy is recommended to fail the test case and capture a screenshot of the application's current state?

A. Surround the test case in a Try Catch. Add a Verify Expression in the Catch block, configured in the following way: Expression: False TakeScreenshotIfFailed: True
B. Surround the test case in a Try Catch. Add a Verify Expression in the Catch block, configured in the following way: Expression: False ContinueOnFailure: False
C. Surround the test case in a Try Catch. Add a Verify Expression in the Catch block, configured in the following way: Expression: True TakeScreenshotIfFailed: True
D. Surround the test case in a Try Catch. Add a Verify Expression in the Finally block, configured in the following way: Expression: 1 > 2 TakeScreenshotIfFailed: True

Answer: A

Explanation:

UiPath Studio supports structured exception handling through the Try Catch activity, allowing automated test workflows to respond to unexpected execution failures while preserving diagnostic evidence.
Option A correctly combines exception handling, explicit verification failure, and screenshot capture. The Try block contains the primary test execution activities. When an exception occurs, control passes to the corresponding Catch block, where the developer can implement diagnostic logging and verification logic.
The Verify Expression activity evaluates a Boolean condition and determines whether the expected verification has succeeded.
Setting Expression to False forces the verification to fail. Configuring TakeScreenshotIfFailed as True instructs the activity to capture a screenshot when that verification fails.
This enables the test execution result to contain evidence of the application state associated with the failure, provided screenshot capture is supported and the application context is available.
Option B produces a failed verification but does not explicitly enable screenshot capture. Option C evaluates a successful expression and therefore does not trigger failure-based screenshot capture.
Option D places the verification in the Finally block, which executes regardless of whether the original test case throws an exception. Consequently, it could incorrectly fail otherwise successful executions.
For reliable reporting, the Catch block should also preserve the original exception details and avoid suppressing unexpected failures.
Official UiPath Reference(s): Testing Activities — Verify Expression; TakeScreenshotIfFailed; ContinueOnFailure; Test Cloud — Application testing best practices.

Question: 10

What distinctive functionality does a test automation project in UiPath Studio provide?

A. It provides the tools necessary for testing software products
B. It enables logging and reporting of workflows
C. It provides the tool for charting and analyzing data
D. It provides the tools necessary for process automation development

Answer: A

Explanation:
A Test Automation project in UiPath Studio is specifically designed to develop, manage, and execute automated tests that validate software application behavior.
Option A correctly describes the primary purpose of this project type. Unlike a standard Process project, which focuses on automating business operations, a Test Automation project provides a development structure optimized for software quality assurance.

UiPath Studio supports both low-code and coded test cases, allowing developers to validate application functionality using Visual Basic or C#. Test automation projects can include multiple test cases, each functioning as an independent execution entry point.
Developers can use the UiPath Testing Activities package to implement assertions, generate test data, and verify expected application outcomes. Data-driven test cases extend this capability by executing the same testing logic against multiple input combinations.

Studio also supports test case templates, execution templates, and integration with Test Manager for test management and execution orchestration.
Option B describes logging and reporting capabilities that are available across multiple project types.
Option C represents analytical functionality rather than the distinctive purpose of a testing project.
Option D describes conventional process automation development.
The architectural distinction is that Test Automation projects are structured around verifiable testing outcomes, whereas Process projects primarily deliver automated business operations.
Official UiPath Reference(s): Studio User Guide — About Automation Projects; Test Cases; Creating application test cases.

Visit us at: https://www.examsboost.com/test/uipath-tcapv1

Thank You for Trying Our Product

For More Information – Visit link below:

https://www.examsboost.com/

15 USD Discount Coupon Code:

G74JA8UF

FEATURES

- ✓ 90 Days Free Updates
- ✓ Money Back Pass Guarantee
- ✓ Instant Download or Email Attachment
- ✓ 24/7 Live Chat Support
- ✓ PDF file could be used at any Platform
- ✓ 50,000 Happy Customer

