

2.7 - Test Controller

Test Controller — Appvance IQ documentation.

- [Test Controller](#)
- [Appvance IQ Live Test Controller](#)
- [How to Access Test Controller](#)
- [Different Functions](#)
- [Review Live Reports During Execution](#)
- [Control Test Execution](#)
- [Review Execution Output](#)
- [Review Test Case Executions](#)
- [Monitor Test Nodes](#)
- [Monitor Active Virtual Users](#)
- [Assert Suite](#)
- [Use AutoCorrelation in Test Controller](#)

Test Controller

Contents

- [Appvance IQ Live Test Controller](#)
- [How to Access Test Controller](#)
- [Different Functions](#)
- [Live Reports](#)
- [Execution Control](#)
- [Output](#)
- [Test Case Executions](#)
- [Test Node](#)
- [Active Virtual Users](#)
- [Assert Suite](#)
- [AutoCorrelation](#)

Last updated: April 20, 2023

If you have any feedback on this topic, please contact us at documentation@appvance.ai

Appvance IQ Live Test Controller

The **Live Controller** provides real-time monitoring of scenario executions in Appvance IQ. It allows users to view execution data as it becomes available and monitor the current state and performance of a running test.

During a scenario execution, the Live Controller displays relevant information such as:

- Test case execution details
- Total transactions
- Successful and failed transactions
- Active virtual users
- Execution status and other real-time performance data

This information allows users to monitor a test while it is running and quickly identify changes in execution behavior, failures, or other conditions that may require attention.

The Live Controller is therefore an important component for monitoring and analyzing scenario executions in real time.

Last updated: August 12, 2026

If you have feedback about this topic, please contact documentation@appvance.ai.

How to Access Test Controller

To access **Live Controller**, you must first start or open a scenario execution in AIQ.

There are two ways to access Live Controller:

From Scenario Editor

When creating or editing a scenario in **Scenario Editor**, you can start its execution by clicking the **Play** button located in the upper-right corner of the AIQ navigation bar.

Starting the scenario opens **Live Controller** in a new browser tab.

From the Play Section

You can also access Live Controller from the **Play** section:

1. Click **Play** in the AIQ navigation bar.
2. Click **Browse Files**.
3. Select the existing scenario file you want to execute.
4. Click **Open** in the file chooser.

AIQ opens Live Controller for the selected scenario.

The **Play** section also displays scenarios that have been opened from this console. From there, you can use the **Play** button to start an execution and access Live Controller.

Using Live Controller

Live Controller provides tools for monitoring and controlling a scenario execution. It is also an important tool for troubleshooting and debugging scenarios.

Before using Live Controller for debugging, familiarize yourself with its basic functionality and the information displayed during a scenario execution.

Last updated: September 3, 2026

If you have any feedback on this topic, please contact us at documentation@appvance.ai.

Different Functions

Before we can continue, it's important for you to learn about the basic functionality from this section to have a better understanding of how to use it, and most of all, how to debug it. These are the basic functionality available from AIQ Live Controller.

Last updated: February 22, 2023

If you have any feedback on this topic, please contact us at documentation@appvance.ai

Review Live Reports During Execution

Review Live Reports During Execution

Live Reports provide a current view of a running test. Use them to monitor health during execution, but wait for the test to complete before making final performance conclusions.

Useful signals

- Active users and ramp behavior.
- Throughput, response time, and error trends.
- Step or transaction failures that appear during the run.
- Test Node or environment symptoms that affect result quality.

How to use live data

1. Confirm the run reaches the expected load pattern.
2. Watch for sudden error spikes or response-time changes.
3. Stop the run if continuing would produce invalid or unsafe results.
4. Use completed results for the final report and comparison.

Last updated: September 18, 2026

Control Test Execution

Control Test Execution

Use Test Controller execution controls to start, monitor, and stop AIQ test runs. Confirm the execution plan, environment, data, and test-node capacity before starting a load or scenario execution.

Before starting

- Verify the target environment and test data are approved for the run.
- Confirm Test Nodes are online and sized for the planned load.
- Review ramp-up, duration, and stop conditions.

During execution

1. Watch active users, throughput, errors, and output.
2. Pause or stop the run if the target environment becomes unstable or the test is invalid.
3. Capture the execution ID and relevant evidence for analysis.

Last updated: September 18, 2026

Review Execution Output

Review Execution Output

Execution Output provides run-level details, errors, and evidence collected during a test. Use it to understand whether a failure came from the scenario, test data, environment, or application under test.

What to review

- Error messages and stack traces tied to failed steps or requests.
- Screenshots, logs, or response details captured during execution.
- Timing patterns that point to slow pages, services, or environment issues.
- Test data consumed by the failed flow when available.

Good practices

1. Start with the first meaningful failure, not every downstream symptom.
2. Compare output with a known good execution when possible.
3. Attach relevant output details to the defect or maintenance ticket.

Last updated: September 18, 2026

Review Test Case Executions

Review Test Case Executions

Use Test Case Executions to review the result of individual scenarios within a run. This view helps identify which scenarios passed, failed, or require maintenance after execution.

Use this view to

- Find the failed scenario or step that caused a run to fail.
- Compare results across repeated executions.
- Review evidence such as output, screenshots, and timing details.
- Confirm whether a failure is caused by test data, application behavior, or environment instability.

Recommended workflow

1. Filter to failed or incomplete executions first.
2. Open the earliest meaningful failure.
3. Review output and related evidence before editing the scenario.
4. Re-run a small validation execution after maintenance.

Last updated: September 18, 2026

Monitor Test Nodes

Monitor Test Nodes

Test Nodes provide execution capacity for AIQ runs. Monitor them before and during execution to confirm that the planned load can run reliably.

What to check

- The required Test Nodes are online and assigned to the correct environment.
- CPU, memory, browser, and driver capacity are appropriate for the planned run.
- Network access to the application under test is available from the execution host.
- Node-side logs do not show browser startup, driver, or resource errors.

Troubleshooting

1. Reduce load and run a validation test if node capacity is uncertain.
2. Review node logs for startup or connectivity errors.
3. Confirm browser and driver versions match the supported AIQ release guidance.

Last updated: September 18, 2026

Monitor Active Virtual Users

Monitor Active Virtual Users

Active virtual users show the number of users currently executing during a test run. Use this view to confirm that load is ramping as expected and that execution capacity matches the test design.

What to check

- The active user count follows the configured ramp-up and ramp-down behavior.
- Unexpected drops are reviewed against errors, test-node capacity, and environment health.
- The test reaches the expected steady-state load before results are interpreted.

Troubleshooting

1. Review Test Node health and available capacity.
2. Check output for startup, data, or authentication failures.
3. Compare active users with throughput and error rate to identify bottlenecks.

Last updated: September 18, 2026

Assert Suite

This section provides the current execution results so you can add custom assertions to your scenario. Several assertion types are available from the **Assert Type** drop-down menu, and some are exclusively for requests. For most assertions, the AIQ UI provides current result information, so you don't have to look up the current results to validate against. These are some of the available assertions:

- **Assert Exists:** Validates whether the target element exists in the results tree.
- **Assert Size:** Validates the actual size of the desired target against the given custom value. Comparison criteria can be set to: equals, not equals, less than, greater than, less than or equal, greater than or equal.
- **Assert Duration:** Validates the actual duration of the desired target against the given custom value. Comparison criteria can be set to: equals, not equals, less than, greater than, less than or equal, greater than or equal.
- **Assert Matches (Request only):** Compares the request-response content to validate whether the given baseline string is contained in the request-response, delimited by the given target accessor. You must know and specify the request-response content format. The target accessor field changes according to the response format:
 - If the response format is plain text, use regular expressions to limit the response range to what is valid (using "." validates against the whole response).
 - If the response format is JSON, provide the JSON path to the desired element to assert.
 - If the response format is XML, provide the XPath to the desired tag.

The only formats that AIQ can handle so far for this procedure are: text, JSON, and XML.

- **Assert Checksum (Request only):** Validates the actual request-response checksum against the given custom checksum.
- **Assert Response Code (Request only):** Validates the actual request-response code against the given custom response code.
- **Store Content:** The very first thing to know about this is that it is NOT an assertion; it is an auxiliary function of AIQ that provides the user with the ability to store request-response content into an internal custom-named variable in The Locker, which can be used in a later request if required. For example, imagine you have a shopping cart web application. One of your requests generates a unique ID number associated with each cart. Let's assume that this number is required to process transactions. In this context, you could store the first request response (the ID) and then instruct AIQ to use it at the expected request, taking the value from the stored content.

Use AutoCorrelation in Test Controller

Use AutoCorrelation in Test Controller

AutoCorrelation helps identify dynamic values that must be captured and reused for a performance test to replay correctly. Use it during script validation before increasing load.

When to review correlation

- Requests fail after login, checkout, search, or session creation.
- The application returns tokens, IDs, or timestamps that change each run.
- A script works once but fails on replay or under load.

Recommended workflow

1. Run a small validation execution.
2. Review failed requests and dynamic response values.
3. Apply correlation rules only to values required by later requests.
4. Replay and verify the script before scaling.

Last updated: September 18, 2026