

9.12 - Scenario Editor - Knowledge Base

Scenario Editor - Knowledge Base — Knowledge Base articles.

- [Scenario Editor Knowledge Base Overview](#)
- [Ability to connect to our DB to run DB validations. What are the possible connectivity issues and possibility to connect?](#)
- [Use SELECT Queries Safely in AIQ SQL](#)
- [Use AIQ Monitors](#)
- [Can we execute different combinations of scripts in a scenario](#)
- [Run AIQ Tests Across Windows and Linux Machines](#)
- [Common issues with AJT and how to solve them](#)
- [Validate PDFs and Other File Types in AIQ](#)
- [DPL Naming conventions](#)
- [Troubleshoot Node Localhost Init Errors](#)
- [Run Java Code with an Array Generator DPL](#)
- [Run Functional Tests Concurrently in AIQ](#)
- [Run Functional Tests from Different Locations](#)
- [Run a Scenario for Available CSV Entries](#)
- [IP Spoofing Setup Guidance for AIQ Tests](#)
- [Maximum Retry Settings for Scenario Editor Tests](#)
- [Need confirmation on other supported file types like JSON and YAML](#)
- [Configure Ramp-Up Settings in Scenario Editor](#)

- [Scenario Editor Store Target and Post-processing](#)
- [Obsolete Internet Explorer 11 Execution Guidance](#)
- [Unable to run scenario in AIQ due to DB connection error](#)
- [Understand Interval Time and Sleep Time](#)

Scenario Editor Knowledge Base Overview

Scenario Editor Knowledge Base Overview

Use this page as the overview for maintained Scenario Editor knowledge base articles. Keep current articles focused on reusable patterns, troubleshooting, test data, waits, execution behavior, and validation practices.

Keep content that explains

- Scenario Editor workflows and maintained terminology.
- Conditional waits, validations, stored values, and reusable test data.
- Execution troubleshooting with evidence, output, and Test Node context.
- Migration notes when older Designer or Scenario Builder content still matters.

Maintenance rules

1. Use “Scenario Editor” instead of “Scenario Builder” or old Designer wording unless documenting legacy migration.
2. Remove duplicate, empty, or product-version-specific content that no longer applies.
3. Do not include real credentials, tokens, internal tenant URLs, or customer-specific data.

Last updated: September 9, 2026

Ability to connect to our DB to run DB validations. What are the possible connectivity issues and possibility to connect?

You can create a SQL script that do DB SQL queries and validate them as the following example:

```
-- step=Connect
-- username=aiq
-- password=thePasswordGoesHere
-- url=jdbc:mysql://mysqismoketest.us-west-2.rds.amazonaws.com:3306/aiq
-- driver_class=com.mysql.jdbc.Driver
-- database=aiq
-- dialect=org.appvance.hibernate.dialect.MySQLDialect
-- connect
-- step=Step1
SELECT * FROM `aiq`.`SummaryElement` LIMIT 100;
-- step=Step2
SELECT * FROM `aiq`.`Definition` LIMIT 100;
-- step=Step3
SELECT * FROM `aiq`.`NodePerScenario` LIMIT 100;
-- step=Step4
SELECT * FROM `aiq`.`SummaryElement` LIMIT 100;
-- step=Step5
SELECT * FROM `aiq`.`UTPIterationPerExecution` LIMIT 100;
-- step=Step6
SELECT * FROM `aiq`.`TypeLong` LIMIT 100;
-- step=Step7
```

```

SELECT * FROM `aiq`.`UTPScenarioExecution` LIMIT 100;
-- step=Step8
SELECT * FROM `aiq`.`UTPIterationPerExecution` LIMIT 100;
-- step=Step9
SELECT * FROM `aiq`.`DefaultPercentil` LIMIT 100;
-- step=Step10
SELECT * FROM `aiq`.`DetailedDuration` LIMIT 100;
-- step=Step11
SELECT * FROM `aiq`.`DetailedElementAttribute` LIMIT 100;
-- step=Step12
SELECT * FROM `aiq`.`SummaryChildElement` LIMIT 100;
-- step=Step13
SELECT * FROM `aiq`.`SummaryElement` LIMIT 100;
-- step=Step14
SELECT * FROM `aiq`.`Definition` LIMIT 100;
-- step=Step15
SELECT * FROM `aiq`.`NodePerScenario` LIMIT 100;
-- step=Step16
SELECT * FROM `aiq`.`SummaryElement` LIMIT 100;
-- step=Step17
SELECT * FROM `aiq`.`UTPIterationPerExecution` LIMIT 100;
-- step=Step18
SELECT * FROM `aiq`.`TypeLong` LIMIT 100;
-- step=Step19
SELECT * FROM `aiq`.`UTPScenarioExecution` LIMIT 100;
-- step=Step20
SELECT * FROM `aiq`.`UTPIterationPerExecution` LIMIT 100;
-- step=Step21
SELECT * FROM `aiq`.`DefaultPercentil` LIMIT 100;
-- step=Step22
SELECT * FROM `aiq`.`DetailedDuration` LIMIT 100;
-- step=BeforeDisc
-- step=Disconnect
-- disconnect

```

if the sql script contains a line after a SELECT line

```
-- storeInLocker=MyTestLockerKey1
```

then the result from the SELECT call will be stored in TheLocker (memory) with a name "MyTestLockerKey1"

You can then use a next script in your use case that consume that response (Java, TestNG, TestDesigner, or others)

Last updated: June 02, 2023

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

Use SELECT Queries Safely in AIQ SQL

Use SELECT Queries Safely in AIQ SQL

When using AIQ SQL in Scenario Editor or related test-data workflows, prefer explicit field lists over broad `SELECT *` patterns. Explicit fields make tests easier to maintain and reduce failures when schemas change.

Recommended query style

- Select only the columns needed by the test.
- Use clear aliases when values are mapped into scenario steps or DPL fields.
- Filter to the smallest safe data set for the validation.
- Avoid logging sensitive values in shared output.

Troubleshooting

1. Confirm the query runs in the approved database client or validation environment.
2. Check quoting rules for the database engine in use.
3. Review whether special characters are being interpreted by AIQ, the database, or the shell/script layer.

Last updated: September 8, 2026

Use AIQ Monitors

Use AIQ Monitors

Use AIQ monitors to observe target behavior and support execution analysis. Monitors should help explain run health, environment behavior, or application symptoms without replacing scenario validations.

When monitors help

- Tracking availability or health signals during long executions.
- Comparing application or environment behavior against test results.
- Collecting evidence for performance or reliability analysis.

Good practices

1. Monitor only signals that support the test objective.
2. Document thresholds, sampling behavior, and environment assumptions.
3. Review monitor data alongside output, reports, and Test Node health.
4. Remove monitors that no longer map to maintained systems or useful signals.

Last updated: September 8, 2026

Can we execute different combinations of scripts in a scenario

Yes. User can combine different script type in a single scenario and run it. If the user wants all the different script type to execute simultaneously then all the different script type should be uploaded in the different test case. If the user wants to run them sequentially then it should be saved in a single test case.

Last updated: June 02, 2023

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

Run AIQ Tests Across Windows and Linux Machines

Last reviewed: September 28, 2026

Use this article when planning Scenario Editor or execution-node coverage across mixed Windows and Linux machines.

What changed in this article

- Corrected the title grammar and changed the answer from a one-off FAQ into mixed-node execution guidance.
- Added practical compatibility checks for browser versions, drivers, filesystem paths, node labels, environment variables, and OS-specific dependencies.
- Clarified that mixed OS execution is valid only when the scenario and test data do not rely on OS-specific assumptions.

Validation workflow

1. Identify which nodes are Windows and which are Linux, including browser and driver versions.
2. Review the scenario for OS-specific paths, shell commands, downloads, or file permissions.
3. Run a small node-targeted execution on each OS before scheduling a full mixed run.

Review note: Added node-readiness validation so mixed Windows/Linux execution does not hide environment-specific failures.

Last updated: September 28, 2026

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

Common issues with AJT and how to solve them

Here is a list of all the issues you might have when dealing with AJT, and how to solved them.

1. Firefox 38 and on - Might have some security issues, use Firefox 27 with auto-update, install Firefox 37 from [{+} https://ftp.mozilla.org/pub/mozilla.org/firefox/releases/37.0/+](https://ftp.mozilla.org/pub/mozilla.org/firefox/releases/37.0/+) and make read-only
2. Firefox not starting - Checking port 9999 is not already being use by a third party app.
3. Firefox not moving forward - remove folder "utp\sahi\userdata\browser\ff\profiles" and restart Appvance UTP.
4. AJT not moving forward - Check port 8585 is already being used by third party app.
5. This files inside firefox (Usually located here C:\Program Files (x86)\Mozilla Firefox) can also cause some issues

Last updated: June 02, 2023

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

Validate PDFs and Other File Types in AIQ

Recommended guidance

AIQ can validate downloaded or generated content such as PDFs and other files when the test defines what should be checked. Choose the validation method based on the file type, expected content, dynamic values, and evidence requirements.

Validation options

- Text extraction for PDFs or documents when content must match expected values.
- Metadata checks such as file name, size, page count, or generated timestamp rules.
- Binary or checksum checks when the exact file must match.
- Visual or screenshot-based checks only when content extraction is not sufficient.

Best practices

1. Control dynamic values such as dates, IDs, and totals before comparing files.
2. Store expected reference files in a reviewed repository or approved project location.
3. Verify download location, browser policy, and file permissions.
4. Attach clear evidence when file validation fails.

Last updated: October 6, 2026

DPL Naming conventions

While saving a CSV file please make sure that the column name is "without spaces". If the column name is "with space" then the Data enabled script will fail with the below error :

```
"java.lang.Exception: Error on this step:Connection reset:Script failed to start withing 150 secs.  
at com.appvance.enterprise.sahiclient.TMSahiRunner.processSteps(TMSahiRunner.java:931)  
at com.appvance.enterprise.sahiclient.TMSahiRunner.runActions(TMSahiRunner.java:343)"
```

if we check in the Output tab we get the below message:

"RESULTS:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<report>
```

```
<step type="error" name="ERROR
```

```
__DIVIDER__SyntaxError: missing ; before statement" ts="1456309276929" dur="110" request=""  
response="" />"
```

Last updated: June 02, 2023

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

Troubleshoot Node Localhost Init Errors

Troubleshoot Node Localhost Init Errors

Use this checklist when a Scenario Editor or Service Suite execution reports a Node localhost init error. The cause is usually environment setup, dependency resolution, port usage, or Test Node runtime configuration.

Check first

- The Test Node is online and assigned to the correct execution.
- The expected Node.js runtime or package dependency is available on the execution host.
- No other process is using the required localhost port.
- Security, proxy, or firewall settings are not blocking localhost communication required by the test.

Validation workflow

1. Run a small validation execution on one Test Node.
2. Review output, node logs, stdout, and stderr.
3. Confirm package versions and working directory assumptions.
4. Escalate with execution ID, Test Node name, logs, and reproduction steps.

Last updated: September 8, 2026

Run Java Code with an Array Generator DPL

Run Java Code with an Array Generator DPL

Use an Array Generator DPL with Java code only when the test requires generated or transformed data that cannot be handled by simpler DPL configuration. Keep Java logic focused on data preparation and avoid embedding secrets or environment-specific values.

Before using this pattern

- Confirm the Java version supported by the AIQ execution environment.
- Validate the code with a small, controlled data set.
- Keep credentials, URLs, and tenant-specific values in approved secure configuration locations.
- Document input shape, output shape, and expected failure behavior.

Validation workflow

1. Run the DPL with a small sample.
2. Confirm generated values map correctly into Scenario Editor steps.
3. Review output for Java exceptions, empty arrays, and malformed values.
4. Scale only after the DPL is repeatable and easy to troubleshoot.

Last updated: September 8, 2026

Run Functional Tests Concurrently in AIQ

Run Functional Tests Concurrently in AIQ

Run functional tests concurrently when the scenarios, test data, and execution environment are ready for parallel execution. Concurrency can expose data collisions, environment limits, and locator or wait issues that do not appear in a single-user run.

Before increasing concurrency

- Validate each scenario with a small single-user execution.
- Confirm test data is isolated or safely reusable.
- Verify Test Node capacity, browser support, and target environment limits.
- Use Scenario Editor conditional waits instead of arbitrary fixed pauses.

Execution guidance

1. Increase concurrency gradually.
2. Monitor active users, errors, output, and environment health.
3. Stop and triage if failures indicate invalid data or unstable infrastructure.
4. Document the concurrency level used when reporting results.

Last updated: September 8, 2026

Run Functional Tests from Different Locations

Run Functional Tests from Different Locations

Last reviewed: September 21, 2026

Use this article when Scenario Editor functional tests must execute from different geographic regions, networks, or test nodes.

When to use this guidance

- You need to compare behavior by location or network path.
- Specific test nodes represent different data centers or regions.
- Results must identify where each execution ran.

Recommended approach

1. Confirm the required locations and assign appropriate test nodes or execution pools.
2. Verify browser, driver, network, DNS, and application access from each node.
3. Parameterize environment or location-specific data instead of duplicating scenarios.
4. Run a small smoke test from each location before full execution.
5. Label results by location so failures can be compared accurately.

Note: Use Scenario Editor terminology for functional tests; avoid older Scenario Builder wording.

Last updated: September 21, 2026

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

Run a Scenario for Available CSV Entries

Run a Scenario for Available CSV Entries

When a Scenario Editor test uses CSV-backed data, align the execution count with the available rows so the run does not reuse, skip, or exhaust data unexpectedly.

Recommended setup

- Confirm the CSV contains only valid rows intended for the execution.
- Document whether the header row is excluded from the data count.
- Decide whether data rows should be consumed once, reused, or reset between runs.
- Keep sensitive data out of CSV files unless the storage and retention process is approved.

Validation workflow

1. Run with a small CSV sample and confirm row consumption behavior.
2. Review output to verify the expected row was used by each iteration.
3. Set the scenario execution count to match the intended number of available data entries.
4. Archive or reset test data according to the environment's data policy.

Last updated: September 8, 2026

IP Spoofing Setup Guidance for AIQ Tests

IP Spoofing Setup Guidance for AIQ Tests

Use IP spoofing only when the test objective explicitly requires traffic to appear from different source addresses and the network owner has approved the setup. This is an advanced environment configuration, not a default Scenario Editor requirement.

Before enabling IP spoofing

- Confirm the target network, firewall, and load-balancer teams approve the test.
- Document the source IP range, target environment, Test Nodes, and test window.
- Verify the application under test is authorized to receive this traffic pattern.
- Keep the configuration out of shared articles if it contains customer-specific network details.

Validation workflow

1. Run a small connectivity test from one Test Node.
2. Confirm the target sees the expected source address behavior.
3. Monitor errors, routing, firewall, and application logs during scale-up.
4. Record the configuration with the execution evidence and rollback steps.

Last updated: September 9, 2026

Maximum Retry Settings for Scenario Editor Tests

Recommended guidance

Use retry settings to reduce noise from temporary timing or environment issues, not to hide unstable locators, incorrect waits, broken test data, or real application defects. A retry should make a valid test more resilient; it should not turn a failing test into an unclear result.

Good use cases

- Short-lived network delays during page load or service calls.
- Minor timing variation in asynchronous UI behavior.
- Known non-deterministic behavior in a test environment that is already being tracked.

Use caution when

- The same step fails repeatedly across retries.
- The retry makes execution time too long for CI/CD or nightly runs.
- The failure involves data creation, purchases, messages, or any action that should not be repeated blindly.
- The scenario lacks clear assertions after the retried action.

Recommended workflow

1. Fix selectors, waits, and test data before increasing retries.
2. Set the smallest retry count that addresses the known transient issue.
3. Document why the retry exists and what condition should allow it to be removed.
4. Review retry-heavy scenarios during maintenance because they often indicate hidden instability.

Last updated: October 5, 2026

Need confirmation on other supported file types like JSON and YAML

Currently CSV and Excel file formats are supported by the Tool as external data file options. Need confirmation on other supported file types like JSON and YAML

To data-drive test designer scripts you can use directly CSV files or RDBMS databases. If you have data that needs to be consumed by a UX script on JSON file or YAML file then you can create a setup java script that read the file and pass the data to the test designer script.

Last updated: June 02, 2023

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

Configure Ramp-Up Settings in Scenario Editor

Configure Ramp-Up Settings in Scenario Editor

This article replaces obsolete “Scenario Builder” wording with current Scenario Editor terminology. Ramp-up controls how quickly virtual users or executions are introduced, which helps avoid unrealistic spikes and makes results easier to interpret.

How to choose ramp-up

- Use a short ramp-up for smoke validation or low-volume checks.
- Use a gradual ramp-up for performance tests where steady-state behavior matters.
- Avoid interpreting response-time results during ramp-up unless ramp behavior is the test objective.
- Confirm Test Node and target environment capacity before increasing load.

Validation workflow

1. Start with a small run and review active users, errors, and output.
2. Increase ramp and duration in controlled stages.
3. Compare steady-state periods between runs.
4. Document ramp-up, duration, load, and environment with the result.

Last updated: September 9, 2026

Scenario Editor Store Target and Post-processing

You can use the Store tab in the scenario editor to define extractors that can "listen" for a particular Step and Request and extract data of its response.

When creating a Store entry or an Assert entry in the Scenario Editor, you need to configure the Target depending on the type of response.

Also, you might require to apply some post-processing to the response. In this document, you will find examples of the possible values that can be used for the Target and the Post-processing inputs.

PostProcessing

Here is a list of examples of all the possible functions that can be used for post-processing entry. `value()` returns the extracted value.

```
“ substring(value(), 0, 2)
  substring(URLEncode(value()), 2)
  URLEncode(value())
  URLDecode(value())
  append(value(), "-abc123")
  length(value())
  URLParam(value(), 'id')
```

Target

In the same way, the TARGET can have a hint text that should be dynamic according to the selected Target.

The `//` characters indicate a comment and is not part of the target, with the exception of the HTML target that is an XPath.

DEFAULT Targets

```
“ duration          // returns the request duration
  responseCode      // returns the response status code
  size              // returns de size in bytes of the response
  md5               // returns checksum md4 value of the response
```

HTTP Targets

Given the response is the following string: "**data=45, data=68**" then

```
“ raw/data[0]          // returns 'data=45' and "raw/data[1]" returns 'data=68'
  #data                // returns 45
  Header:Accept        // returns de value of the header Accept
  CookieValue:CSRF-TOKEN // returns de value of the cookie CSRF-TOKEN
```

HTML Targets

Use normal XPathS to extract values from HTML. Note that in this case the double slash at the beginning is needed in the target.

```
“ //input[@name='__RequestVerificationToken']/@value
```

JSON Targets

Use JSON paths to extract values from JSON responses, like

```
“ employees[0].address.city
  [2].carousel.images[1].path
```

PDF Targets

```
“ document          // gets all text in document except forms text
  page[2]           // returns al text in page 2 except forms text
  documentforms      // gets a json with contents of the forms in the d
  regex:Once upon a .* ever after. // returns text that matches the regular expressio
```

STRING Targets

Given the response is the following string: "**The tested processed returned: ID=ABCDEF.**" then

```
“ 11,20"          // returns "processed"
   ID=false       // returns "ABCDEF."
   ID=true        // returns "ID=ABCDEF."
   ID=38,false    // returns "ABCDE"
   ID=38,true     // returns "ID=ABCDE"
```

TEXT Targets

The target should be specified as a regular expression, like:

```
“ (?<=\"__RequestVerificationToken\" type=\"hidden\" value=\")[a-zA-Z0-9\\-_]+
   id=.*
```

CSV Targets

Given that you have a response formatted as a CSV with the following data

id	name	state
1	john	California
2	barry	Nevada
3	stef	Texas
4	karla	New York

Then:

```
“ 2,2            // returns barry
   last(),"name" // returns karla
   ,"name"       // returns a string array {"name","john","barry","stef","karla"}
   3,last()      // returns Texas
   id=3, 2       // returns stef
```

Last updated: June 02, 2023

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

Obsolete Internet Explorer 11 Execution Guidance

Last reviewed: October 2, 2026

This page is retained only to flag that Internet Explorer 11 execution guidance is obsolete and should not be used for current AIQ testing.

What changed in this article

- Marked IE11 content as obsolete because Internet Explorer 11 is retired and not a current browser target for AIQ guidance.
- Replaced troubleshooting steps with a migration note toward supported Chromium-based execution paths.
- Added checks to confirm whether the issue still reproduces in supported browsers before treating it as an AIQ scenario problem.

Validation workflow

1. Do not create new AIQ test coverage that depends on IE11.
2. Reproduce the workflow in a supported browser and review browser-specific differences.
3. If a customer still has an IE11 dependency, document it as legacy risk and plan migration.

Review note: This article should be removed or archived once no customers require the legacy reference.

Last updated: October 2, 2026

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

Unable to run scenario in AIQ due to DB connection error

User is getting below error while trying to run scenarios in AIQ

"Database connection failed: Io exception: The Network Adaptor could not establish the connection"

The above issue is because DB is not reachable and hence AIQ is not able to establish connection.

Please follow the below points to make sure DB is up and running

- 1) Go to Option page in AIQ
- 2) Go to Preferences->Results
- 3) Click "Test Connection"
- 4) If you get a pop up "Connection success" then the DB is up and running and you can the scenario without any issues
- 5) if you still receive an error then there is an issue with your DB.

Last updated: June 02, 2023

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

Understand Interval Time and Sleep Time

Understand Interval Time and Sleep Time

Interval time and sleep time both affect execution pacing, but they should be used carefully. Prefer condition-based waits when the test needs to wait for application state, and use timing settings only when pacing is the actual test objective.

Concepts

- **Interval time** controls spacing or pacing between repeated activity, depending on the execution configuration.
- **Sleep time** pauses execution for a fixed duration.
- Fixed waits can hide real application issues and make tests slower than needed.

Good practices

1. Use Scenario Editor conditional waits for page readiness, element state, and asynchronous behavior.
2. Use fixed sleep only when the delay itself is required by the business or system workflow.
3. Keep wait values documented and review repeated timeout failures before increasing them.
4. Validate timing changes with a small execution before scaling.

Last updated: September 9, 2026