2.17 - Validation Suite and Store Content

Validation Suite and Store Content — Appvance IQ documentation.

- Validation Suite and Store Content Overview
- Validation and Store Content in Scenario Editor
- Validate and Store HAR Content in AIQ
- Store Target and Post-Processing
- Extract and Validate PDF Content

Validation Suite and Store Content Overview

Recommended guidance

Use Validation Suite and Store Content features when AIQ tests need to verify content, files, responses, or extracted values as part of repeatable validation. Keep validation assets organized so the expected evidence is clear and maintainable.

Useful validation targets

- Downloaded files, PDFs, images, text blocks, metadata, and extracted values.
- API responses or service outputs that must match expected content.
- Scenario Editor evidence used to confirm application behavior.
- Stored reference content used for regression checks.

Maintenance checklist

1. Define what content is expected and what differences are acceptable.
2. Control dynamic values such as dates, IDs, timestamps, and generated content.
3. Store reference artifacts in a reviewed repository or approved project location.
4. Review failures to separate real application changes from outdated expected content.

Last updated: October 6, 2026

Validation and Store Content in Scenario Editor

Validation and Store Content in Scenario Editor

Use validation and stored content when a Scenario Editor test needs to capture a value from the application and compare it later in the same scenario or execution. This keeps the test focused on business behavior instead of fixed screen text.

Common uses

- Capture an order number, account value, token, or confirmation message.
- Validate that a value displayed later matches the value generated earlier.
- Store a dynamic value for a downstream step in the same test flow.

Recommended workflow

1. Identify the stable element or response value to capture.
2. Store the value with a clear name that describes its business meaning.
3. Use the stored value in a later validation step.
4. Run a short validation execution before adding the pattern to a larger suite.

Last updated: September 18, 2026

Validate and Store HAR Content in AIQ

Use this article when a test needs to capture, validate, or store HAR network content for later analysis.

Validation workflow

1. Capture the HAR during the scenario step that produces the network traffic.
2. Validate the request/response fields needed for the test objective.
3. Store or export the HAR only after checking that sensitive data has been removed or masked.

Last updated: September 25, 2026

Store Target and Post-Processing

The **Store** tab in the Scenario Editor allows you to define extractors that listen for a specific Step and Request and extract data from its response.

When creating a **Store** or **Assert** entry in the Scenario Editor, you must configure the **Target** according to the type and format of the response.

You can also apply **post-processing** to the extracted value. Post-processing allows you to transform, encode, decode, or otherwise manipulate the value before it is stored or used by the scenario.

This document provides examples of the values that can be used for the **Target** and **Post-processing** fields.

Post-Processing

Post-processing functions operate on the value extracted by the Target.

The `value()` function represents the value extracted by the configured Target.

Examples:

- `substring(value(), 0, 2)` — returns a substring of the extracted value.
- `URLEncode(value())` — URL-encodes the extracted value.
- `URLDecode(value())` — URL-decodes the extracted value.
- `append(value(), "-abc123")` — appends the specified string to the extracted value.
- `length(value())` — returns the length of the extracted value.
- `URLParam(value(), 'id')` — extracts the specified URL parameter from the value.

Target

The **Target** field supports different syntaxes depending on the type of response being processed.

The Target hint text in the Scenario Editor can change dynamically according to the selected target type.

The `//` characters indicate a comment and are not part of the Target syntax, except for **HTML Targets**, where `//` is part of the XPath expression.

Default Targets

The following Targets can be used with any supported response:

- `duration` — returns the request duration.
- `responseCode` — returns the HTTP response status code.
- `size` — returns the response size in bytes.
- `md5` — returns the MD5 checksum of the response.

HTTP Targets

For example, given the following response:

```
data=45, data=68
```

You can use:

- `raw/data[0]` — returns `data=45`.
- `raw/data[1]` — returns `data=68`.
- `#data` — returns `45`.
- `Header:Accept` — returns the value of the `Accept` header.
- `CookieValue:CSRF-TOKEN` — returns the value of the `CSRF-TOKEN` cookie.

HTML Targets

Use standard XPath expressions to extract values from HTML responses.

For HTML Targets, the `//` at the beginning of the Target is part of the XPath and is required.

Example:

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

This Target returns the value of the `value` attribute of the matching input element.

JSON Targets

Use JSON paths to extract values from JSON responses.

For example, given a JSON response containing an `employees` array:

- `employees[0].address.city` — returns the city of the first employee.
- `[2].carousel.images[1].path` — returns the `path` value of the second image in the carousel of the third element.

PDF Targets

The following Targets can be used to extract information from PDF responses:

- `document` — returns all text in the document, excluding form text.
- `page[2]` — returns all text from page 2, excluding form text.
- `documentforms` — returns a JSON representation of the forms contained in the document.
- `regex:Once upon a .* ever after.` — returns the text matching the specified regular expression, if found.

String Targets

String Targets can be used to extract a portion of a response based on delimiters or positions.

For example, given the following response:

```
The tested process returned: ID=ABCDEF.
```

Examples:

- `11,20` — returns `processed`.
- `ID=,false` — returns `ABCDEF.`
- `ID=,true` — returns `ID=ABCDEF.`
- `ID=,38,false` — returns the portion of the response according to the specified delimiter and position.
- `ID=38,true` — returns the portion of the response according to the specified position and inclusion option.

> **Note:** The exact behavior of positional String Targets depends on the Target syntax supported by the current AIQ version.

Text Targets

For text responses, specify the Target as a regular expression.

Examples:

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

The Target returns the text matching the specified regular expression.

CSV Targets

CSV Targets allow you to extract values from responses formatted as CSV.

For example, given the following CSV data:

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

Examples:

- `2,2` — returns `barry`.
- `last(),"name"` — returns `karla`.
- `,"name"` — returns an array containing `name`, `john`, `barry`, `stef`, and `karla`.
- `3,last()` — returns `Texas`.
- `id=3,2` — returns `stef`.

> **Note:** Make sure the indexing behavior used by CSV Targets matches the indexing convention of your AIQ version.

Last updated: August 12 2026

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

Extract and Validate PDF Content

Use PDF content extraction when a Scenario Editor test needs to confirm that a generated PDF contains expected text or values. This is most useful for invoices, statements, confirmations, reports, and other documents produced by the application under test.

Recommended workflow

1. Generate or download the PDF during the test run.
2. Extract the relevant text from the PDF output.
3. Validate only the stable business values needed for the test objective.
4. Record enough evidence in the execution output to diagnose failures.

Good practices

- Avoid validating full documents when a small set of fields proves the behavior.
- Normalize whitespace, dates, and dynamic values before comparison when needed.
- Keep expected values in approved test data sources instead of hard-coding them into multiple scenarios.

Last updated: September 18, 2026