Amber Tool Validation

Data Management Software Solutions

December 24, 2018

Abstract

In this document Amber is the Configuration Item being validated and Report Package is the Amber Report Package. Amber is validated after this Report Package has been executed, test evidence has been obtained, and test evidence supports the conclusion Intended Use Requirements (IUR) are satisfied. This document is stored in Fresenius-Kabi's (Company) Quality Management System after Amber has been validated.

Contents

1 Introduction

1.1 Overview

This report demonstrates the Amber driver consumes YAML Test Plans, Test Suites, and Test Cases and produces output that is properly formatted for reports designed to tlc-article and audotoc conventions.

1.2 Purpose

This Report Package is a detailed record that provides a Configuration Item overview, a list of its Intended Use Requirements (IUR), one or more Test Reports, evidence the Test Reports ran, along with the output produced by the Test Report. The Test Report includes a pass/fail result for each Test Step and Test Report, a statement indicating the Configuration Item has a Configuration Identification and a conclusion that the Configuration Item has been validated for its intended use.

1.3 Scope

This Report Package applies to Company medical device software projects that have determined a Configuration Item must be validated for its intended use. This Report Package covers activities associated with validating a Configuration Item for its intended use requirements.

1.4 Deviations

The process governing the creation of this protocol and report deviates from the normal standard operating procedure (SOP-FQA02002 Protocols and Reports). This document combines both the protocol and the report. Normally the protocol is released first and report is released after the protocol is executed. This document represents an automated protocol execution facilitated through the use of automation scripting and software. The review of a paper protocol and pre-approval of said protocol does not satisfy the need to review the automated components used for the generation of this document. As a result, the automated components which codify the actual test protocol are reviewed by a technical approver as this document and the components are developed. This technical approver is an approver of this document and their approval indicates the automated components effectively test the article under test to meet the intended use as specified in the user requirements.

Additionally data obtained from the execution of the protocol is collected and presented in the grey boxes as objective evidence from the automated test application. Normally this would not be presented together with the protocol, but given this is an automated process in a combined document; this is an effective means of retaining and presenting the objective evidence for review and approval.

Finally, presenting the protocol and the report together allows for a single step automation process that can be easily maintained and re-executed. Re-execution is often desired due to changes to the article under test or changes to user needs.

1.5 Tool Validation Objectives

1. Describe the intended use of the tool.
2. Set the purpose and scope for the tool validation effort.
3. Enumerate intended use requirements.
4. Disclose compliance criteria.
5. Define Tool validation acceptance criteria.
6. Identify responsible persons and their roles.
7. Document required deliverables.
8. Define specific test steps and test steps to confirm that the Tool's intended use requirements have been met.
9. Collect test evidence.
10. Record Tool validation conclusion.

1.6 General Terms

Configuration Control
The systematic process for managing changes to and established baseline.

Configuration Identification
A unique identifier used to associate a collection of software artifacts.

Configuration Items
Software source code, executables, build scripts, and other software development and software test artifacts relevant to creating and maintaining a software project.

Configuration Status Accounting
The recording and reporting of the information needed to effectively manage the software and documentation components of a software project.

Report Package
A detailed record that provides a Configuration Item overview, a list of its Intended Use Requirements (IUR), one or more Test Reports, evidence the Test Reports ran along with the output produced by Test Report including a pass/fail result for each Test Step and Test Report, and a statement indicating the Configuration Item has a Configuration Identification, and conclusion that the Configuration Item has been validated for its intended use.

Test Plan
A test plan is a collection of one or more test suites a tester has determined to use to challenge requirements.

Test Suite
A test suite is a collection of one or more test cases a tester has determined to use to challenge requirements.

Test Case
A test case is a set of conditions under which a tester will determine whether the test is working as it was originally established for it to do.

Test Step
A unique test identifier with predetermined expectation, confirmation criteria, and pass/fail result.

Test Report
A test report consists of Detailed instructions for the setup, execution, and evaluation of results for a given test. The test protocol may include one or more test cases for which the steps of the protocol will repeat with different input data. Test cases are chosen to ensure that corner cases in the code and data structures are covered. A test protocol may be a script that is automatically run by the computer.

1.7 General Acronyms

FDA Food and Drug Administration

IUR Intended Use Requirements

LMS Learning Management System

SOP Standard Operating Procedure

SOUP Software Of Unknown Provenance

1.8 References

1. SOP-PRC02004 Software Development Procedure
2. SOP-PRC02004 Software Development Procedure; Software Configuration Management
3. SOP-FE0101005 Good Documentation Practices
4. SOP-FQA02002 Protocols and Reports

5. SOP-PRC02001 Issue Tracking Procedure

6. 21 CFR 820.70(i) Automated Processes, and General Principles of Software Validation

7. General Principles of Software Validation; Final Guidance for Industry and FDA Staff; January 11, 2002

1.9   Training

Company's training records are stored in the Quality Management System. Additional training is not reqired because this is an automated test that is executed by the the automated testing platform. SOP-PRC02004 Software Development Procedure provides training required to create, maintain, and execute this testing protocol.

1.10   Tool Validation Test Approach

This Test Plan describes a series of Test Suites, Test Cases, and Test Steps. When executed, each Test Step determines if the Configuration Item satisfies one or more software requirements. When a Test Step indicates that the software requirements are satisfied, the Test Step's result is "pass". Otherwise, the Test Step's result is "fail". The computer records all "pass" and "fail" results in the Test Plan record. The Configuration Item is considered verified when all Test Steps are executed and the Test Plan record contains no "fail" results. Each Test Step that results in a deviation, observation, incident, or failure shall be represented in the final report.

1.11   Configuration Management

When a Configuration Item is changed, we will review the manufacturer's release notes or our design history file (DHF) to determine if regression testing or adjustments to this Report Package is necessary. We will verify the changes do not impact product operation, product quality, or quality decision made prior to performing the upgrade.

1.12   Test Plan Instructions

This Test Plan describes Test Suits, Test Cases, and Test Steps that demonstrate how the Configuration Item satisfies the IUR. Each Test Plan describes any setup criteria needed to conduct the test. Each Test Plan contains a list of IUR\s and the steps that demonstrate how the Configuration Item satisfies the IUR. Each Test Step is marked passed or failed as it is completed. Each Test Plan is marked passed when all Test Steps pass or failed if a single Test Step fails. Failures are addressed per SOP-PRC02001 Issue Tracking Procedure. This serves as a record of the completed test.

Test Plans are automatically run by the computer, generating a report in PDF format. This Report Package is reviewed prior to execution per SOP-PRC02004 Software Development Procedure. The Report Package is routed and archived in the Quality Management System. When it becomes necessary to annotate a computer generated document SOP-FE0101005 Good Documentation Practices must be followed.

1.13   Test Plan Storage and Review

This Test Plan is part of a Company's automated validation framework. The framework consists of following parts:

LaTeX files are used to provide an Abstract, Introduction, Intended Use Requirements, Test Plan Overview, Test Equipment, Configuration Item Validation, Conclusion, and Change Summary. LaTeX files are converted assembled into PDF documents. PDF documents are routed using the Company's document management system for approval.

Ruby software is used to run the automated framework to collect test evidence.

Git is used as the storage repository for LaTeX & YAML files, a Git pull-request is used to review the LaTeX & YAML files prior to use.

Evidence Test Plan output includes one Test Suite, Test Plan, and Test Step, and Test Evidence.

YAML files define the Test Plan, Test Suite, and Test Steps that are processed to generate test evidence.

2   Requirements

Intended-use requirements are defined using the following story format:

As a <type of user>, I want <some goal> so that <some reason>

AMBER-IUR-001

As the designer, I want to demonstrate Amber can process Test Plans, Test Suites, and Test Cases so that I can produce a testing report.

AMBER-IUR-002

As the designer, I want to demonstrate Amber can invoke an another executable program so that I can collect evidence for my test reports.

AMBER-IUR-003

As the designer, I want to demonstrate Amber can accept command line options so that I can controll the type of testing Amber conducts.

AMBER-IUR-004

As the designer, I want to demonstrate Amber can support nested Test Suites and Test Cases so that test objects can be logically organized.

AMBER-IUR-005

As the designer, I want to demonstrate Amber can substitute keywords when processing YAML files so that the maintenance of Test Plans, Test Suites, and Test Cases is minimized.

Amber Tool Validation

3 Test Plan Overview

This section describes Test Plans, Test Suites, Test Cases, and Test Steps that demonstrate how a Configuration Item satisfies the IUR. Each Test Plan describes any setup criteria needed to conduct the Test Steps. Each Test Plan contains a list of IUR\s and the Test Steps that demonstrate how the Configuration Item satisfies the IUR. Each Test Step is marked passed or failed as it is completed. Each Test Plan is marked passed when all Test Steps pass or failed if a single Test Step fails. This serves as a record of completed Test Plans and Test Steps.

Each Test Plan is described in its own section. The order the Test Plans are listed is the order they are run. Each Test Plan defines:

name	Each Plan, Suite, and Case has a unique name.
purpose	Each Plan, Suite, and Case has a purpose.
Test Steps	Each step has a confirmation and expectation along with the command needed to challenge the IUR.
Objective Evidence	A record the Test Plan was run along with any evidence collected while the Test Steps were run.
Traceability	Suites and Cases are traced to an IUR that is challenged. IUR can be traced to multiple Suites and Cases.

Each Test Plan, Test Suite, Test Case, and Test Step has been designed to be run by the computer. However, a person may choose to manually run the Test Plans, save the test results, and generate this test report as specified in the appropriate design documentation.

The example below runs two commands: 1) git help and 2) cat /gitconfig. The output from both commands are written to the system console.

```
1  plan:
2    name: A Test Plan Name
3    purpose: purpose of the plan
4
5  suite:
6    name: A Test Suite Name
7    purpose: a suite purpose
8    requirement: IUR01 and IUR02
9
10   - case:
11     name: A Test Case name
12     purpose: A Test Case purpose
13     steps:
14       - confirm: Confirm git help is written to the console output.
15         expectation: Git help is displayed.
16         command: git
17         argument: help
18
19       - confirm: Confirm .git config is written to the console.
20         expectation: .gitconfig is written to the console output.
21         command: cat
22         argument: .gitconfig
```

4 Test Evidence

The Company's automation framework assembles the content in this section. The section has one or more Test Plans, Test Suites, Test Cases, and Test Evidence. The evidence provided is used to conclude the Tool has met the Intended Use Requirements.

4.1 Test Plan: master

Purpose: This plan demonstrates Ruby gem Amber functions correctly. This plan uses shows that Amber has met the intended-use requirements as defined by the designer.

4.1.1 Test Suite: options

Purpose: This test suite demonstrats Amber consumes and uses command line arguments properly.

4.1.2 Test Case: browser

Purpose: This test case is used to demonstrate Amber properly uses the –browser command line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1        Step: 1
2    Confirm: Amber properly consumes the command line argument ——browser.
3
4 Expectation: Rspec output shows Amber::CommandLineOptions.browser_option handles
     supported argument formats.
5
6    Command: rspec ——format documentation −e 'Amber Browser'
7 Test Result: PASS
8    Evidence: Starts on next line.
9 Run options: include {:full_description=>/Amber\ Browser/}
10
11 Amber Browser
12   ——browser=Chrome
13     has been used from the command line.
14   ——browser Chrome
15     has been used from the command line.
16   −bChrome
17     has been used from the command line.
18   ——browser=Firefox
19     has been used from the command line.
20   ——browser Firefox
21     has been used from the command line.
22   −bFirefox
23     has been used from the command line.
24   ——browser=IE
25     has been used from the command line.
26   ——browser IE
27     has been used from the command line.
28   −bIE
29     has been used from the command line.
30   ——browser=Opra
31     has been used from the command line.
32   ——browser Opra
33     has been used from the command line.
34   −bOpra
35     has been used from the command line.
36
37 Finished in 0.00663 seconds (files took 0.41522 seconds to load)
38 12 examples, 0 failures
```

4.1.3 Test Case: case

Purpose: This test case is used to demonstrate Amber properly uses the –case command line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1          Step: 1
2       Confirm: Amber properly consumes the command line argument −−case.
3
4  Expectation: Rspec output shows Amber::CommandLineOptions.case_option handles supported
        argument formats.
5
6        Command: rspec −−format documentation −e 'Amber Case'
7  Test Result: PASS
8      Evidence: Starts on next line.
9  Run options: include {:full_description=>/Amber\ Case/}
10
11 Amber Case
12   −−case=foo
13     has been used from the command line.
14   −−case foo
15     has been used from the command line.
16   −cfoo
17     has been used from the command line.
18   −−case=foo/baz
19     has been used from the command line.
20   −−case foo/baz
21     has been used from the command line.
22   −−coo/baz
23     has been used from the command line.
24
25 Finished in 0.00616 seconds (files took 0.40266 seconds to load)
26 6 examples, 0 failures
```

4.1.4 Test Case: default

Purpose: This test case is used to demonstrate Amber properly constructs a default Amber::Options object.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1          Step: 1
2       Confirm: Amber properly consructs a default Amber::Options object.
3
4  Expectation: Rspec output shows Amber::Options object is initialized  correctly.
5
6        Command: rspec −−format documentation −e 'Amber Defaults'
7  Test Result: PASS
8      Evidence: Starts on next line.
9  Run options: include {:full_description=>/Amber\ Defaults/}
10
11 Amber Defaults
12   for browser
13     has been provided.
14   for dryrun
15     has been provided.
16   for environment
17     has been provided.
18   for filename
19     has been provided.
20   for language
21     has been provided.
22   for parser
23     has been provided.
24   for obliterate
25     has been provided.
26   for simulate
27     has been provided.
28   for verbose
29     has been provided.
30   for writer
31     has been provided.
32   for okay_to_run?
33     has been provided.
34   for okay_to_echo_env?
35     has been provided.
```

```
36    for okay_to_obliterate?
37      has been provided.
38    for has_browser?
39      has been provided.
40    for has_language?
41      has been provided.
42
43 Finished in 0.00678 seconds (files took 0.4177 seconds to load)
44 15 examples, 0 failures
```

4.1.5   Test Case: environment

Purpose: This test case demonstrates that Amber can record the operational environment in which it was used.
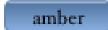
Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1        Step: 1
2     Confirm: Amber understands when to record the operational environment.
3
4 Expectation: Rspec output shows Amber's understanding of the enviornment option.
5
6     Command: rspec −−format documentation −e 'Amber Environment'
7 Test Result: PASS
8    Evidence: Starts on next line.
9 Run options: include {:full_description=>/Amber\ Environment/}
10
11 Amber Environment
12   no −e
13     has not been used.
14   −e
15     has been used erom the command line.
16   −−environment
17     has been used from the command line.
18
19 Finished in 0.00527 seconds (files took 0.40466 seconds to load)
20 3 examples, 0 failures
```

4.1.6   Test Case: file

Purpose: This test case is used to demonstrate Amber properly uses the –file command line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1        Step: 1
2     Confirm: Amber properly consumes the command line argument −−file.
3
4 Expectation: Rspec output shows Amber::CommandLineOptions.file_option handles supported
     argument formats.
5
6     Command: rspec −−format documentation −e 'Amber File'
7 Test Result: PASS
8    Evidence: Starts on next line.
9 Run options: include {:full_description=>/Amber\ File/}
10
11 Amber File
12   no −f
13     has not been used.
14   −fa.yaml
15     has been used from the command line.
16   −−file=b.yaml
17     has been used from the command line.
18   −−file c.yaml
19     has been used from the command line.
20
21 Finished in 0.00645 seconds (files took 0.40559 seconds to load)
22 4 examples, 0 failures
```

4.1.7 Test Case: language

Purpose: This test case is used to demonstrate Amber properly uses the –language command line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1         Step: 1
2      Confirm: Amber properly consumes the command line argument −−language.
3
4  Expectation: Rspec output shows Amber::CommandLineOptions.language_option handles
        supported argument formats.
5
6      Command: rspec −−format documentation −e 'Amber Language'
7  Test Result: PASS
8     Evidence: Starts on next line.
9  Run options: include {:full_description=>/Amber\ Language/}
10
11 Amber Language
12   no −l
13     has not been used.
14   −−language=cs
15     has been used from the command line.
16   −−language cs
17     has been used from the command line.
18   −lda
19     has been used from the command line.
20   −l=da
21     has been used from the command line.
22   −−language=de
23     has been used from the command line.
24   −−language de
25     has been used from the command line.
26   −−language=en
27     has been used from the command line.
28   −−language en
29     has been used from the command line.
30   −−language=es
31     has been used from the command line.
32   −−language es
33     has been used from the command line.
34   −−language=fr−ca
35     has been used from the command line.
36   −−language fr−ca
37     has been used from the command line.
38   −lfr−eu
39     has been used from the command line.
40   −l  fr−eu
41     has been used from the command line.
42   −−language=it
43     has been used from the command line.
44   −−language it
45     has been used from the command line.
46   −lne
47     has been used from the command line.
48   −l=ne
49     has been used from the command line.
50   −−language=no
51     has been used from the command line.
52   −−language no
53     has been used from the command line.
54   −lpl
55     has been used from the command line.
56   −l=pl
57     has been used from the command line.
58   −−language=ro
59     has been used from the command line.
60   −−language ro
61     has been used from the command line.
62   −lsv
63     has been used from the command line.
64   −l=sv
65     has been used from the command line.
```

```
66    −l sv
67      has been used from the command line .
68
69 Finished in 0.0105 seconds ( files took 0.41205 seconds to load )
70 28 examples , 0 failures
```

4.1.8   Test Case: nodryrun

Purpose: This test case is used to demonstrate Amber properly uses the –nodryrun command
line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1         Step : 1
2      Confirm : Amber properly consumes the command line argument −−nodryrun .
3
4  Expectation : Rspec output shows Amber :: CommandLineOptions . nodryrun_option handles
     supported argument formats .
5
6      Command : rspec −−format documentation −e 'Amber NoDryRun'
7  Test Result : PASS
8      Evidence : Starts on next line .
9  Run options : include {: full_description=>/Amber\ NoDryRun/}
10
11 Amber NoDryRun
12   no −n
13     has not been used .
14   −n
15     has been used from the command line .
16   −−nodryrun
17     has been used from the command line .
18
19 Finished in 0.0052 seconds ( files took 0.42375 seconds to load )
20 3 examples , 0 failures
```

4.1.9   Test Case: obliterate

Purpose: This test case is used to demonstrate Amber properly uses the –obliterate command
line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1         Step : 1
2      Confirm : Amber properly consumes the command line argument −−obliterate .
3
4  Expectation : Rspec output shows Amber :: CommandLineOptions . obliterate_option handles
     supported argument formats .
5
6      Command : rspec −−format documentation −e 'Amber Obliterate '
7  Test Result : PASS
8      Evidence : Starts on next line .
9  Run options : include {: full_description=>/Amber\ Obliterate/}
10
11 Amber Obliterate
12   no −O
13     has not been used .
14   −O
15     has been used from the command line .
16   −−obliterate
17     has been used from the command line .
18
19 Finished in 0.00544 seconds ( files took 0.40102 seconds to load )
20 3 examples , 0 failures
```

4.1.10   Test Case: plan

Purpose: This test case is used to demonstrate Amber properly uses the –plan command line
option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1         Step: 1
2     Confirm: Amber properly consumes the command line argument −−plan.
3
4  Expectation: Rspec output shows Amber::CommandLineOptions.plan_option handles supported
       argument formats.
5
6     Command: rspec −−format documentation −e 'Amber Plan'
7  Test Result: PASS
8    Evidence: Starts on next line.
9  Run options: include {:full_description=>/Amber\ Plan/}
10
11 Amber Plan
12   no −p
13     has not been used.
14   −−plan=foo
15     has been used from the command line.
16   −pbar
17     has been used from the command line.
18   −−plan baz
19     has been used from the command line.
20   −p foobar
21     has been used from the command line.
22
23 Finished in 0.00576 seconds (files took 0.40256 seconds to load)
24 5 examples, 0 failures
```

4.1.11   Test Case: simulate

Purpose: This test case is used to demonstrate Amber properly uses the –simulate command line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1         Step: 1
2     Confirm: Amber properly consumes the command line argument −−simulate.
3
4  Expectation: Rspec output shows Amber::CommandLineOptions.simulate_option handles
       supported argument formats.
5
6     Command: rspec −−format documentation −e 'Amber Simulate'
7  Test Result: PASS
8    Evidence: Starts on next line.
9  Run options: include {:full_description=>/Amber\ Simulate/}
10
11 Amber Simulate
12   no −S
13     has not been used.
14   −S
15     has been used from the command line.
16   −−simulate
17     has been used from the command line.
18
19 Finished in 0.00539 seconds (files took 0.40122 seconds to load)
20 3 examples, 0 failures
```

4.1.12   Test Case: suite

Purpose: This test case is used to demonstrate Amber properly uses the –suite command line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1         Step: 1
2     Confirm: Amber properly consumes the command line argument −−suite.
3
4  Expectation: Rspec output shows Amber::CommandLineOptions.suite_option handles supported
       argument formats.
5
```

```
 6       Command: rspec --format documentation -e 'Amber Suite'
 7   Test Result: PASS
 8      Evidence: Starts on next line.
 9   Run options: include {:full_description=>/Amber\ Suite/}
10
11   Amber Suite
12     no -s
13       has not been used.
14     --suite=foo
15       has been used from the command line.
16     -sbar
17       has been used from the command line.
18     --suite baz
19       has been used from the command line.
20     -s foobar
21       has been used from the command line.
22
23   Finished in 0.00631 seconds (files took 0.40281 seconds to load)
24   5 examples, 0 failures
```

4.1.13 Test Case: verbose

Purpose: This test case is used to demonstrate Amber properly uses the –verbose command line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
 1           Step: 1
 2       Confirm: Amber properly consumes the command line argument --verbose.
 3
 4   Expectation: Rspec output shows Amber::CommandLineOptions.verbose_option handles
         supported argument formats.
 5
 6       Command: rspec --format documentation -e 'Amber Verbose'
 7   Test Result: PASS
 8      Evidence: Starts on next line.
 9   Run options: include {:full_description=>/Amber\ Verbose/}
10
11   Amber Verbose
12     no -v
13       has not been used.
14     -v
15       has been used from the command line.
16     --verbose
17       has been used from the command line.
18
19   Finished in 0.00574 seconds (files took 0.40474 seconds to load)
20   3 examples, 0 failures
```

4.1.14 Test Case: version

Purpose: This test case is used to demonstrate Amber properly uses the –version command line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
 1           Step: 1
 2       Confirm: Amber properly consumes the command line argument --version.
 3
 4   Expectation: Rspec output shows Amber::CommandLineOptions.version_option handles
         supported argument formats.
 5
 6       Command: rspec --format documentation -e 'Amber Version'
 7   Test Result: PASS
 8      Evidence: Starts on next line.
 9   Run options: include {:full_description=>/Amber\ Version/}
10
11   Amber Version
12     --version
13       was not used. However the version number must match 1.3.192
14     --version
```

```
15  1.3.192
16      has been used from the command line.
17    Version
18      has a version number
19      version number must match 1.3.192
20
21  Finished in 0.00636 seconds (files took 0.40603 seconds to load)
22  4 examples, 0 failures
```

4.1.15   Test Case: writer

Purpose: This test case is used to demonstrate Amber properly uses the –writer command line option.

Requirement: AMBER-IUR-001, AMBER-IUR-002 and AMBER-IUR-003

```
1         Step: 1
2      Confirm: Amber properly consumes the command line argument --writer.
3
4  Expectation: Rspec output shows Amber::CommandLineOptions.writer_option handles
      supported argument formats.
5
6      Command: rspec --format documentation -e 'Amber Writer'
7  Test Result: PASS
8    Evidence: Starts on next line.
9  Run options: include {:full_description=>/Amber\ Writer/}
10
11  Amber Writer
12    no -w
13      has not been used.
14    --writer=Ascii
15      has been used from the command line.
16    --writer Ascii
17      has been used from the command line.
18    -wAscii
19      has been used from the command line.
20    -w Ascii
21      has been used from the command line.
22    --writer=LaTeX
23      has been used from the command line.
24    --writer LaTeX
25      has been used from the command line.
26    -wLaTeX
27      has been used from the command line.
28    -w LaTeX
29      has been used from the command line.
30
31  Finished in 0.00651 seconds (files took 0.41716 seconds to load)
32  9 examples, 0 failures
```

4.1.16   Test Suite: substitute

Purpose: This test suite demonstrates Amber's runtime substitution capabilities. Amber has been designed to translate the keywords below.

1) browser or BROWSER 2) file or FILE 3) language or LANGUAGE 4) language-code or LANGUAGE-CODE 5) home or HOME or

4.1.17   Test Case: browser

Purpose: This test case is used to demonstrate the browser keyword is properly substituted by Amber.

Requirement: AMBER-IUR-004 and AMBER-IUR-005

```
1         Step: 1
2      Confirm: Amber properly substutes the {browser} keyword for all brower types.
3
4  Expectation: Rspec output shows Amber::Substitute.browser properly substutited {browser}
      and {BROWSER} keywords to Chrome, Firefox, Edge, and IE.
```

```
5
6      Command: rspec ——format documentation −e 'YAML Browser Substitutions'
7  Test Result: PASS
8      Evidence: Starts on next line.
9  Run options: include {:full_description=>/YAML\ Browser\ Substitutions/}
10
11 YAML Browser Substitutions
12   Amber::Substitute.browser
13     can substitute ${BROWSER} to None
14     can substitute ${browser} to None
15     can substitute ${BROWSER} to Chrome
16     can substitute ${browser} to Edge
17     can substitute ${BROWSER} to Firefox
18     can substitute ${browser} to IE
19     can substitute ${BROWSER} to Opera
20
21 Finished in 0.0046 seconds (files took 0.40263 seconds to load)
22 7 examples, 0 failures
```

4.1.18   Test Case: extend-path

Purpose: This test case is used to demonstrate the tilda marker is properly substituted by Amber.

Requirement: AMBER-IUR-004 and AMBER-IUR-005

```
1         Step: 1
2      Confirm: Amber properly substutes the tilda marker correctly for the operating
   system.
3
4  Expectation: Rspec output shows Amber::Substitute.extend_path substutited ~ to the home
      direcory.
5
6      Command: rspec ——format documentation −e 'YAML Extend Path Substitutions'
7  Test Result: PASS
8      Evidence: Starts on next line.
9  Run options: include {:full_description=>/YAML\ Extend\ Path\ Substitutions/}
10
11 YAML Extend Path Substitutions
12   Amber::Substitute.expected_path
13     can expand ~ to C:/Users/Owner
14     can expand ~ and ~ to C:/Users/Owner and C:/Users/Owner
15
16 Finished in 0.00326 seconds (files took 0.4056 seconds to load)
17 2 examples, 0 failures
```

4.1.19   Test Case: home

Purpose: This test case is used to demonstrate the home keyword is properly substituted by Amber.

Requirement: AMBER-IUR-004 and AMBER-IUR-005

```
1         Step: 1
2      Confirm: Amber properly substutes the {home} keyword for all brower types.
3
4  Expectation: Rpec output shows Amber::Substitute.home properly substutited {home} and {
   HOME} keywords specific to this operating system.
5
6      Command: rspec ——format documentation −e 'YAML Home Substitutions'
7  Test Result: PASS
8      Evidence: Starts on next line.
9  Run options: include {:full_description=>/YAML\ Home\ Substitutions/}
10
11 YAML Home Substitutions
12   Amber::Substitute.home
13     can substitute ${home} to ~
14     can substitute ${HOME} to ~
15     can substitute ${home} and ${HOME} to ~ and ~
16
17 Finished in 0.00351 seconds (files took 0.40025 seconds to load)
```

```
18  3 examples, 0 failures
```

4.1.20 Test Case: language

Purpose: This test case is used to demonstrate the language keyword is properly substituted by Amber.

Requirement: AMBER-IUR-004 and AMBER-IUR-005

```
1        Step: 1
2     Confirm: Amber properly substutes the {language} keyword for all supported languages
        .
3
4  Expectation: Rspec output shows Amber::Substitute.language properly substutited {
        language} and {LANGUAGE} keywords to zz, cs, da, de, en, es, fr−ca, fr−eu, it, ne,
        no, pl, so, and sv.
5
6        Command: rspec −−format documentation −e 'YAML Language Substitutions'
7  Test Result: PASS
8     Evidence: Starts on next line.
9  Run options: include {:full_description=>/YAML\ Language\ Substitutions/}
10
11  YAML Language Substitutions
12    Amber::Substitute.language
13      can substitute ${LANGUAGE} to n/a
14      can substitute ${langauge} to Chez
15      can substitute ${LANGAUGE} to Dansk
16      can substitute ${langauge} to Deutsch
17      can substitute ${LANGAUGE} to English
18      can substitute ${langauge} to Espanol
19      can substitute ${LANGAUGE} to CA French − Canadian
20      can substitute ${langauge} to EU French − European
21      can substitute ${LANGAUGE} to Italiano
22      can substitute ${langauge} to Nederlands
23      can substitute ${LANGAUGE} to Norsk
24      can substitute ${langauge} to Polish
25      can substitute ${LANGAUGE} to Romanian
26      can substitute ${langauge} to Svenska
27
28  Finished in 0.00602 seconds (files took 0.41277 seconds to load)
29  14 examples, 0 failures
```

4.1.21 Test Case: language-code

Purpose: This test case is used to demonstrate the language-code keyword is properly substituted by Amber.

Requirement: AMBER-IUR-004 and AMBER-IUR-005

```
1        Step: 1
2     Confirm: Amber properly substutes the {language−code} keyword for all supported
        languages.
3
4  Expectation: Rspec output shows Amber::Substitute.language−code properly substutited {
        language−code} and {LANGUAGE−CODE} keywords to n/a, Czech, Dansk, Deutsch, English,
        Espanol, CA French − Canadian, EU French − European, Italiano, Nederlands, Norsk,
        Polish, Romanian, and Svenska.
5
6        Command: rspec −−format documentation −e 'YAML Language Code Substitutions'
7  Test Result: PASS
8     Evidence: Starts on next line.
9  Run options: include {:full_description=>/YAML\ Language\ Code\ Substitutions/}
10
11  YAML Language Code Substitutions
12    Amber::Substitute.langage_code
13      can substitute ${LANGUAGE−CODE} to zz
14      can substitute ${langauge−code} to cs
15      can substitute ${LANGAUGE−CODE} to da
16      can substitute ${langauge−CODE} to de
17      can substitute ${LANGAUGE−CODE} to en
18      can substitute ${langauge−code} to es
```

```
19    can  substitute  ${LANGAUGE−CODE}  to  fr −ca
20    can  substitute  ${langauge−code}  to  fr −eu
21    can  substitute  ${LANGAUGE−CODE}  to  it
22    can  substitute  ${langauge−code}  to  ne
23    can  substitute  ${LANGAUGE−CODE}  to  no
24    can  substitute  ${langauge−code}  to  pl
25    can  substitute  ${LANGAUGE−CODE}  to  ro
26    can  substitute  ${langauge−code}  to  sv
27
28 Finished  in  0.00492  seconds  ( files  took  0.4036  seconds  to  load )
29 14  examples ,  0  failures
```

4.1.22 Test Case: strings

Purpose: This test case is used to demonstrate the Amber substitutes multiple keywords in data stream.

Requirement: AMBER-IUR-004 and AMBER-IUR-005

```
1         Step :  1
2      Confirm :  Amber  properly  substutes  all  keywords  in  a  data  stream .
3
4 Expectation :  Rspec  output  shows  Amber :: Substitute . strings  substitued  all  keywords
       without  encountering  an  error .
5
6     Command :  rspec  −−format  documentation  −e  ’YAML  Strings  Substitutions ’
7 Test  Result :  PASS
8     Evidence :  Starts  on  next  line .
9 Run  options :  include  {: full_description=>/YAML\  Strings\  Substitutions /}
10
11 YAML  Strings  Substitutions
12   Amber :: Substitute . strings
13     can  substitute  ${BROWSER}  to  Opera
14     can  substitute  ${browser}  ${ file }  ${language}  and  ${language−code}  to  Opera  baz
       Svenska  and  sv
15     can  substitute  ${language−code}${ file }${language}${browser}  to  svbazSvenskaOpera
16
17 Finished  in  0.00416  seconds  ( files  took  0.41977  seconds  to  load )
18 3  examples ,  0  failures
```

4.1.23 Test Suite: structure

Purpose: This test suite demonstrated Amber's ability to locate a nested Test Plan, Test Suite, or Test Case YAML file.

4.1.24 Test Case: factory

Purpose: This test case is used to demonstrate Amber can properly locate a nested Test Plan, Test Suite, or Test Case.

Requirement: AMBER-IUR-004

```
1         Step :  1
2      Confirm :  Amber  properly  locates  nested  Test  Plan ,  Test  Suite ,  and  Test  Case  names .
3
4 Expectation :
5 Rspec  output  shows  Amber :: FactoryStrucutre  properly  locates  the  YAML  file
6 below .
7 1)  Test  Plan  foo
8 2)  Nested  Test  Plan  baz
9 3)  Test  Suite  foo
10 4)  Nested  Test  Suite  name  baz
11 5)  Test  Case  foo
12 6)  Nested  Test  Case  name  baz
13
14     Command :  rspec  −−format  documentation  −e  ’Factory  Structure ’
15 Test  Result :  PASS
16     Evidence :  Starts  on  next  line .
17 Run  options :  include  {: full_description=>/Factory\  Structure /}
18
```

```
19 Factory Structure
20   Test Plan Name
21     is normal.
22     is nested.
23   Test Suite Name
24     is normal.
25     is nested.
26   Test Case Name
27     is normal.
28     is nested.
29
30 Finished in 0.00464 seconds (files took 0.40486 seconds to load)
31 6 examples, 0 failures
```

5   Configuration Item Conclusion

This Report Package has satisfied the IUR for the Configuration Item described herein thus the Configuration Item is considered validated for its intended use.

Change Summary

Change	Justification
Initial version.	New document.