Amber Tool Validation

Department Name

March 8, 2022

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 Acme Corporation's (Company) Quality Management System after Amber has been validated.

Contents

Approval Signatures

Role	Name	Signature
Author	Gary A. Howard	Electronic signature on file.
Project Lead	N/A	Electronic signature on file.
Technical Lead	N/A	Electronic signature on file.
System Engineer	N/A	Electronic signature on file.
Software Engineer	N/A	Electronic signature on file.
Test Engineer	N/A	Electronic signature on file.
Product Manager	N/A	Electronic signature on file.
Clinical App. Manager	N/A	Electronic signature on file.
RA Specialist	N/A	Electronic signature on file.
QA Engineer	Dayn A. Howard	Electronic signature on file.
Medical Doctor	N/A	Electronic signature on file.
Technical Reviewer	Cj Howard	Electronic signature on file.
Technical Reviewer	N/A	Electronic signature on file.
Technical Reviewer	N/A	Electronic signature on file.
Technical Reviewer	N/A	Electronic signature on file.
Independent Reviewer	N/A	Signature attached.

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 (SOP005 Validation of Computerized Systems). 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

Document 123-VNV-056022 Validation Determination report provides a Determination of Validation decision tree and Determination of Level of Risk and Validation Rigor decision tree to aid a Development Team when assessing the need for validation. 123-VNV-056022 was updated to indicate this tool required Validation and the Level of Risk needed. The steps below describe the steps used to validate a tool.

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 set-up, 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

- SOP004 Software Devopment Procedure
- SOP003 Software Configuration Management
- SOP001 Good Documentation Practices
- SOP005 Validation of Computerized Systems
- SOP002 Change Management

1.9 Training

Company's training records are stored in the Quality Management System. Additional training is not required because this is an automated test that is executed by the automated testing platform. SOP004 Software Devopment 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 system requirements. When a Test Step indicates that the system 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 SOP002 Change Management. 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 SOP004 Software Devopment Procedure. The Report Package is routed and archived in the Quality Management System. When it becomes necessary to annotate a computer generated document SOP001 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 control 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-IUR-006

As the designer, I want to demonstrate Amber can support embedded LaTeXenumerate and itemized commands so that Test Plans, Test Suites and Test Cases have beautifully typeset lists.

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 Step, save the test results, and generate this test report as specified in the appropriate design documentation. The example below runs these commands:

1. git help
2. cat .gitconfig

The output from both commands are written to the system console.

```
plan:
  name: A Test Plan Name
  purpose: purpose of the plan

suite:
  name: A Test Suite Name
  purpose: a suite purpose
  requirement: IUR01 and IUR02

  - case:
    name: A Test Case name
    purpose: A Test Case purpose
    steps:
      - confirm: Confirm git help is written to the console output.
        expectation: Git help is displayed.
        command: git
        argument: help

      - confirm: Confirm .git config is written to the console.
        expectation: .gitconfig is written to the console output.
        command: cat
        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: LaTeX. LaTeX This Test Plan demonstrates the Ruby gem Amber functions correctly. This Test plan shows Amber has met the intended-use requirements defined by the designer. This Test Plan includes the Test Suites listed below.

- Test Suite cli/options shows all permutations of command line options function correctly.
- Test Suite utility/substitution demonstrates how Amber substitutes values found in Test Plans, Test Suites, and Test Cases to simplify test maintenance.
- Test Suite tif/structure reveals concepts unique to a Test Input Factory
- Test Suite advanced-concept demonstrates embedded LaTeX commands throughout.

Amber Test Cases have been designed to showoff Amber concepts. There are many Test Steps that may report failure. In these cases, an operation system application is simply not installed. For example, most Linux systems come with the man program installed. Amber will issue a command 'which man' and report PASS if man is installed and FAIL if man is not installed. In the failed test case, Amber prints the environment path that was searched. In short, Amber simply ran the command it was instructed to run and captured the results. This is sufficient for Amber's validation purposes.

4.1.1 Test Suite: options

Purpose: This test suite demonstrates 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

Step: 1

Confirm: Amber properly consumes the command line argument –browser.

Expectation: Rspec output shows Amber::CommandLineOptions.browser_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO Browser'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Browser/}

Amber CLO Browser
  —browser=Chrome
    has been used from the command line.
  —browser Chrome
    has been used from the command line.
  –bChrome
    has been used from the command line.
```

```
  --browser=Firefox
    has been used from the command line.
  --browser Firefox
    has been used from the command line.
  -bFirefox
    has been used from the command line.
  --browser=IE
    has been used from the command line.
  --browser IE
    has been used from the command line.
  -bIE
    has been used from the command line.
  --browser=Opra
    has been used from the command line.
  --browser Opra
    has been used from the command line.
  -bOpra
    has been used from the command line.

Finished in 0.0081 seconds (files took 0.35206 seconds to load)
12 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 526 / 839
    LOC (62.69%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –case.

Expectation: Rspec output shows Amber::CommandLineOptions.case_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO Case'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Case/}

Amber CLO Case
  no -c
    has not been used.
  --case=foo
    has been used from the command line.
  -cbar
    has been used from the command line.
  --case baz
    has been used from the command line.
  -c foobar
    has been used from the command line.

Finished in 0.00647 seconds (files took 0.34793 seconds to load)
5 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 509 / 839 LOC (60.67%) covered.
```

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

Step: 1

Confirm: Amber properly constructs a default Amber::Options object.

Expectation: Rspec output shows Amber::Options object is initialized correctly.

Command: rspec –format documentation -e 'Amber CLO Defaults'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Defaults/}

All examples were filtered out

Finished in 0.00027 seconds (files took 0.33861 seconds to load)
0 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 490 / 839
    LOC (58.4%) covered.
```

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

Step: 1

Confirm: Amber understands when to record the operational environment.

Expectation: Rspec output shows Amber's understanding of the environment option.

Command: rspec –format documentation -e 'Amber CLO Environment'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Environment/}

Amber CLO Environment
  no -e
    has not been used.
  -e
    has been used from the command line.
  --environment
    has been used from the command line.

Amber CLO Environment
  dollar_signs are escaped.

Finished in 0.0062 seconds (files took 0.33378 seconds to load)
4 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 505 / 839
    LOC (60.19%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –file.

Expectation: Rspec output shows Amber::CommandLineOptions.file_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO File'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ File/}

Amber CLO File
  no -f
    has not been used.
  -fa.yaml
    has been used from the command line.
  --file=b.yaml
    has been used from the command line.
  --file c.yaml
    has been used from the command line.

Finished in 0.00608 seconds (files took 0.32647 seconds to load)
4 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 505 / 839
    LOC (60.19%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –language.

Expectation: Rspec output shows Amber::CommandLineOptions.language_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO Language'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Language/}

Amber CLO Language
  no -l
    has not been used.
  with unknown language
    --language XX
      raises an invalid argument exception
    --language=XX
      raises an invalid argument exception
    -l XX
      raises an invalid argument exception
    -lXX
      raises an invalid argument exception
  behaves like Amber CLO language parameter
    --language=zz
      returns n/a when run with double dash and equal sign
    --language zz
      returns n/a when run with double dash and a space
    -lzz
      returns n/a when run with dash and no space
    -l zz
      returns n/a when run with dash and a space
  behaves like Amber CLO language parameter
    --language=cs
      returns Czech when run with double dash and equal sign
    --language cs
      returns Czech when run with double dash and a space
    -lcs
      returns Czech when run with dash and no space
    -l cs
      returns Czech when run with dash and a space
  behaves like Amber CLO language parameter
    --language=cy
      returns Welsh when run with double dash and equal sign
    --language cy
      returns Welsh when run with double dash and a space
    -lcy
      returns Welsh when run with dash and no space
    -l cy
      returns Welsh when run with dash and a space
  behaves like Amber CLO language parameter
    --language=da
      returns Danish when run with double dash and equal sign
    --language da
      returns Danish when run with double dash and a space
    -lda
      returns Danish when run with dash and no space
    -l da
      returns Danish when run with dash and a space
  behaves like Amber CLO language parameter
    --language=de
```

```
      returns German when run with double dash and equal sign
  --language de
      returns German when run with double dash and a space
  -lde
      returns German when run with dash and no space
  -l de
      returns German when run with dash and a space
behaves like Amber CLO language parameter
  --language=en
      returns English when run with double dash and equal sign
  --language en
      returns English when run with double dash and a space
  -len
      returns English when run with dash and no space
  -l en
      returns English when run with dash and a space
behaves like Amber CLO language parameter
  --language=es
      returns Spanish; Castilian when run with double dash and equal sign
  --language es
      returns Spanish; Castilian when run with double dash and a space
  -les
      returns Spanish; Castilian when run with dash and no space
  -l es
      returns Spanish; Castilian when run with dash and a space
behaves like Amber CLO language parameter
  --language=fi
      returns Finnish when run with double dash and equal sign
  --language fi
      returns Finnish when run with double dash and a space
  -lfi
      returns Finnish when run with dash and no space
  -l fi
      returns Finnish when run with dash and a space
behaves like Amber CLO language parameter
  --language=fr
      returns French when run with double dash and equal sign
  --language fr
      returns French when run with double dash and a space
  -lfr
      returns French when run with dash and no space
  -l fr
      returns French when run with dash and a space
behaves like Amber CLO language parameter
  --language=fr-ca
      returns CA French - Canadian when run with double dash and equal sign
  --language fr-ca
      returns CA French - Canadian when run with double dash and a space
  -lfr-ca
      returns CA French - Canadian when run with dash and no space
  -l fr-ca
      returns CA French - Canadian when run with dash and a space
behaves like Amber CLO language parameter
  --language=fr-eu
      returns EU French - European when run with double dash and equal sign
  --language fr-eu
      returns EU French - European when run with double dash and a space
  -lfr-eu
      returns EU French - European when run with dash and no space
  -l fr-eu
      returns EU French - European when run with dash and a space
behaves like Amber CLO language parameter
  --language=fy
      returns Western Frisian when run with double dash and equal sign
  --language fy
      returns Western Frisian when run with double dash and a space
  -lfy
      returns Western Frisian when run with dash and no space
  -l fy
      returns Western Frisian when run with dash and a space
behaves like Amber CLO language parameter
  --language=it
      returns Italian when run with double dash and equal sign
```

```
  --language it
    returns Italian when run with double dash and a space
  -lit
    returns Italian when run with dash and no space
  -l it
    returns Italian when run with dash and a space
behaves like Amber CLO language parameter
  --language=nl
    returns Dutch; Flemish when run with double dash and equal sign
  --language nl
    returns Dutch; Flemish when run with double dash and a space
  -lnl
    returns Dutch; Flemish when run with dash and no space
  -l nl
    returns Dutch; Flemish when run with dash and a space
behaves like Amber CLO language parameter
  --language=no
    returns Norwegian when run with double dash and equal sign
  --language no
    returns Norwegian when run with double dash and a space
  -lno
    returns Norwegian when run with dash and no space
  -l no
    returns Norwegian when run with dash and a space
behaves like Amber CLO language parameter
  --language=pl
    returns Polish when run with double dash and equal sign
  --language pl
    returns Polish when run with double dash and a space
  -lpl
    returns Polish when run with dash and no space
  -l pl
    returns Polish when run with dash and a space
behaves like Amber CLO language parameter
  --language=pt
    returns Portuguese when run with double dash and equal sign
  --language pt
    returns Portuguese when run with double dash and a space
  -lpt
    returns Portuguese when run with dash and no space
  -l pt
    returns Portuguese when run with dash and a space
behaves like Amber CLO language parameter
  --language=ro
    returns Romanian; Moldavian; Moldovan when run with double dash and equal sign
  --language ro
    returns Romanian; Moldavian; Moldovan when run with double dash and a space
  -lro
    returns Romanian; Moldavian; Moldovan when run with dash and no space
  -l ro
    returns Romanian; Moldavian; Moldovan when run with dash and a space
behaves like Amber CLO language parameter
  --language=ru
    returns Russian when run with double dash and equal sign
  --language ru
    returns Russian when run with double dash and a space
  -lru
    returns Russian when run with dash and no space
  -l ru
    returns Russian when run with dash and a space
behaves like Amber CLO language parameter
  --language=sk
    returns Slovak when run with double dash and equal sign
  --language sk
    returns Slovak when run with double dash and a space
  -lsk
    returns Slovak when run with dash and no space
  -l sk
    returns Slovak when run with dash and a space
behaves like Amber CLO language parameter
  --language=sv
    returns Swedish when run with double dash and equal sign
  --language sv
```

```
      returns Swedish when run with double dash and a space
    -lsv
      returns Swedish when run with dash and no space
    -l sv
      returns Swedish when run with dash and a space
  behaves like Amber CLO language parameter
    --language=zu
      returns Zulu when run with double dash and equal sign
    --language zu
      returns Zulu when run with double dash and a space
    -lzu
      returns Zulu when run with dash and no space
    -l zu
      returns Zulu when run with dash and a space

Finished in 0.03085 seconds (files took 0.32424 seconds to load)
93 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 521 / 839 LOC (62.1%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –nodryrun.

Expectation: Rspec output shows Amber::CommandLineOptions.nodryrun_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO NoDryRun'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ NoDryRun/}

Amber CLO NoDryRun
  no -n
    has not been used.
  -n
    has been used from the command line.
  --nodryrun
    has been used from the command line.

Finished in 0.00567 seconds (files took 0.3201 seconds to load)
3 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 498 / 839
    LOC (59.36%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –obliterate.

Expectation: Rspec output shows Amber::CommandLineOptions.obliterate_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO Obliterate'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Obliterate/}

Amber CLO Obliterate
  no -O
    has not been used.
  -O
    has been used from the command line.
  --obliterate
    has been used from the command line.

Finished in 0.00574 seconds (files took 0.32156 seconds to load)
3 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 498 / 839
    LOC (59.36%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –plan.

Expectation: Rspec output shows Amber::CommandLineOptions.plan_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO Plan'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Plan/}

Amber CLO Plan
  no -p
    has not been used.
  --plan=foo
    has been used from the command line.
  -pbar
    has been used from the command line.
  --plan baz
    has been used from the command line.
  -p foobar
    has been used from the command line.

Finished in 0.00615 seconds (files took 0.31851 seconds to load)
5 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 509 / 839 LOC (60.67%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –simulate.

Expectation: Rspec output shows Amber::CommandLineOptions.simulate_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO Simulate'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Simulate/}

Amber CLO Simulate
  no -S
    has not been used.
  -S
    has been used from the command line.
  --simulate
    has been used from the command line.

Finished in 0.00569 seconds (files took 0.31646 seconds to load)
3 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 498 / 839
    LOC (59.36%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –suite.

Expectation: Rspec output shows Amber::CommandLineOptions.suite_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO Suite'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Suite/}

Amber CLO Suite
  no -s
    has not been used.
  --suite=foo
    has been used from the command line.
  -sbar
    has been used from the command line.
  --suite baz
    has been used from the command line.
  -s foobar
    has been used from the command line.

Finished in 0.00613 seconds (files took 0.31517 seconds to load)
5 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 509 / 839 LOC (60.67%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –verbose.

Expectation: Rspec output shows Amber::CommandLineOptions.verbose_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO Verbose'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Verbose/}

Amber CLO Verbose
  no -v
    has not been used.
  -v
    has been used from the command line.
  --verbose
    has been used from the command line.

Finished in 0.00567 seconds (files took 0.31638 seconds to load)
3 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 498 / 839
    LOC (59.36%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –version.

Expectation: Rspec output shows Amber::CommandLineOptions.version_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO Version'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Version/}

Amber CLO Version
  Version
    has a version number
    version number must match 1.6.0.367
  no --version
    was not used. However the version number must match 1.6.0.367
  --version
1.6.0.367
    has been used from the command line.

Finished in 0.00613 seconds (files took 0.31579 seconds to load)
4 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 497 / 839
    LOC (59.24%) covered.
```

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

Step: 1

Confirm: Amber properly consumes the command line argument –writer.

Expectation: Rspec output shows Amber::CommandLineOptions.writer_option handles supported argument formats.

Command: rspec –format documentation -e 'Amber CLO Writer'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Amber\ CLO\ Writer/}

Amber CLO Writer
  no -w
    has not been used.
  --writer=Ascii
    has been used from the command line.
  --writer Ascii
    has been used from the command line.
  -wAscii
    has been used from the command line.
  -w Ascii
    has been used from the command line.
  --writer=LaTeX
    has been used from the command line.
  --writer LaTeX
    has been used from the command line.
  -wLaTeX
    has been used from the command line.
  -w LaTeX
    has been used from the command line.

Finished in 0.00728 seconds (files took 0.31233 seconds to load)
9 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 516 / 839
    LOC (61.5%) covered.
```

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 ~

This Test case also demonstrated embedded LaTeX syntax to define the list above.

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

Step: 1

Confirm: Amber properly substitutes the {browser} keyword for all browser types.

Expectation: RSpec output shows Amber::Substitute.browser properly substituted {browser} and {BROWSER} keywords to Chrome, Firefox, Edge, and IE.

Command: rspec –format documentation -e 'YAML Browser Substitutions'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/YAML\ Browser\ Substitutions/}

YAML Browser Substitutions
  Amber::Substitute.browser
    can substitute ${BROWSER} to None
    can substitute ${browser} to None
    can substitute ${BROWSER} to Brave
    can substitute ${BROWSER} to Chrome
    can substitute ${browser} to Edge
    can substitute ${BROWSER} to Firefox
    can substitute ${browser} to IE
    can substitute ${BROWSER} to Opera

Finished in 0.00632 seconds (files took 0.3174 seconds to load)
8 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 512 / 839
   LOC (61.03%) covered.
```

4.1.18 Test Case: extend-path

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

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

Step: 1

Confirm: Amber properly substitutes the tilde marker correctly for the operating system.

Expectation: Rspec output shows Amber::Substitute.extend_path substituted ~ to the home directory.

Command: rspec –format documentation -e 'YAML Extend Path Substitutions'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/YAML\ Extend\ Path\ Substitutions/}

YAML Extend Path Substitutions
  Amber::Substitute.expected_path
    can expand ~ to /home/traap
    can expand ~ and ~ to /home/traap and /home/traap

Finished in 0.00519 seconds (files took 0.31307 seconds to load)
2 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 492 / 839
    LOC (58.64%) covered.
```

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

Step: 1

Confirm: Amber properly substitutes the {home} keyword for all browser types.

Expectation: Rpec output shows Amber::Substitute.home properly substituted {home} and {HOME} keywords specific to this operating system.

Command: rspec –format documentation -e 'YAML Home Substitutions'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/YAML\ Home\ Substitutions/}

YAML Home Substitutions
  Amber::Substitute.home
    can substitute ${home} to ~
    can substitute ${HOME} to ~
    can substitute ${home} and ${HOME} to ~ and ~

Finished in 0.00528 seconds (files took 0.32014 seconds to load)
3 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 494 / 839
    LOC (58.88%) covered.
```

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

Step: 1

Confirm: Amber properly substitutes the {language} keyword for all supported languages.

Expectation: Rspec output shows Amber::Substitute.language properly substituted {language} and {LANGUAGE} keywords to zz, cs, da, de, en, es, fr-ca, fr-eu, it, ne, no, pl, so, and sv.

Command: rspec –format documentation -e 'YAML Language Substitutions'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/YAML\ Language\ Substitutions/}

YAML Language Substitutions
  behaves like Amber::Substitute.language
    can substitute ${language} to n/a
    can substitute ${LANGUAGE} to n/a
  behaves like Amber::Substitute.language
    can substitute ${language} to Czech
    can substitute ${LANGUAGE} to Czech
  behaves like Amber::Substitute.language
    can substitute ${language} to Welsh
    can substitute ${LANGUAGE} to Welsh
  behaves like Amber::Substitute.language
    can substitute ${language} to Danish
    can substitute ${LANGUAGE} to Danish
  behaves like Amber::Substitute.language
    can substitute ${language} to German
    can substitute ${LANGUAGE} to German
  behaves like Amber::Substitute.language
    can substitute ${language} to English
    can substitute ${LANGUAGE} to English
  behaves like Amber::Substitute.language
    can substitute ${language} to Spanish; Castilian
    can substitute ${LANGUAGE} to Spanish; Castilian
  behaves like Amber::Substitute.language
    can substitute ${language} to Finnish
    can substitute ${LANGUAGE} to Finnish
  behaves like Amber::Substitute.language
    can substitute ${language} to French
    can substitute ${LANGUAGE} to French
  behaves like Amber::Substitute.language
    can substitute ${language} to CA French – Canadian
    can substitute ${LANGUAGE} to CA French – Canadian
  behaves like Amber::Substitute.language
    can substitute ${language} to EU French – European
    can substitute ${LANGUAGE} to EU French – European
  behaves like Amber::Substitute.language
    can substitute ${language} to Western Frisian
    can substitute ${LANGUAGE} to Western Frisian
  behaves like Amber::Substitute.language
    can substitute ${language} to Italian
    can substitute ${LANGUAGE} to Italian
  behaves like Amber::Substitute.language
    can substitute ${language} to Dutch; Flemish
    can substitute ${LANGUAGE} to Dutch; Flemish
  behaves like Amber::Substitute.language
    can substitute ${language} to Norwegian
    can substitute ${LANGUAGE} to Norwegian
  behaves like Amber::Substitute.language
    can substitute ${language} to Polish
```

```
    can substitute ${LANGUAGE} to Polish
  behaves like Amber::Substitute.language
    can substitute ${language} to Portuguese
    can substitute ${LANGUAGE} to Portuguese
  behaves like Amber::Substitute.language
    can substitute ${language} to Romanian; Moldavian; Moldovan
    can substitute ${LANGUAGE} to Romanian; Moldavian; Moldovan
  behaves like Amber::Substitute.language
    can substitute ${language} to Russian
    can substitute ${LANGUAGE} to Russian
  behaves like Amber::Substitute.language
    can substitute ${language} to Slovak
    can substitute ${LANGUAGE} to Slovak
  behaves like Amber::Substitute.language
    can substitute ${language} to Swedish
    can substitute ${LANGUAGE} to Swedish
  behaves like Amber::Substitute.language
    can substitute ${language} to Zulu
    can substitute ${LANGUAGE} to Zulu

Finished in 0.01241 seconds (files took 0.31641 seconds to load)
44 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 492 / 839
    LOC (58.64%) covered.
```

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

Step: 1

Confirm: Amber properly substitutes the {language-code} keyword for all supported languages.

Expectation: Rspec output shows Amber::Substitute.language-code properly substituted {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.

Command: rspec –format documentation -e 'YAML Language Code Substitutions'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/YAML\ Language\ Code\ Substitutions/}

YAML Language Code Substitutions
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to zz
    substitutes ${langauge-code} to zz
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to cs
    substitutes ${langauge-code} to cs
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to cy
    substitutes ${langauge-code} to cy
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to da
    substitutes ${langauge-code} to da
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to de
    substitutes ${langauge-code} to de
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to en
    substitutes ${langauge-code} to en
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to es
    substitutes ${langauge-code} to es
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to fi
    substitutes ${langauge-code} to fi
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to fr
    substitutes ${langauge-code} to fr
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to fr-ca
    substitutes ${langauge-code} to fr-ca
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to fr-eu
    substitutes ${langauge-code} to fr-eu
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to fy
    substitutes ${langauge-code} to fy
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to it
    substitutes ${langauge-code} to it
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to nl
    substitutes ${langauge-code} to nl
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to no
    substitutes ${langauge-code} to no
```

```
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to pl
    substitutes ${langauge-code} to pl
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to pt
    substitutes ${langauge-code} to pt
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to ro
    substitutes ${langauge-code} to ro
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to ru
    substitutes ${langauge-code} to ru
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to sk
    substitutes ${langauge-code} to sk
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to sv
    substitutes ${langauge-code} to sv
  behaves like Amber::Substitute.language_code
    substitutes ${LANGUAGE-CODE} to zu
    substitutes ${langauge-code} to zu

Finished in 0.01246 seconds (files took 0.31448 seconds to load)
44 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 492 / 839
    LOC (58.64%) covered.
```

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

Step: 1

Confirm: Amber properly substitutes all keywords in a data stream.

Expectation: Rspec output shows Amber::Substitute.strings substituted all keywords without encountering an error.

Command: rspec –format documentation -e 'YAML Strings Substitutions'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/YAML\ Strings\ Substitutions/}

YAML Strings Substitutions
  Amber::Substitute.strings
    can substitute ${BROWSER} to Opera
    can substitute ${browser} ${file} ${language} and ${language-code} to Opera baz Swedish and sv
    can substitute ${language-code}${file}${language}${browser} to svbazSwedishOpera

Finished in 0.00534 seconds (files took 0.31813 seconds to load)
3 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 493 / 839 LOC (58.76%) covered.
```

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

Step: 1

Confirm: Amber properly locates nested Test Plan, Test Suite, and Test Case names.

Expectation: Rspec output shows Amber::FactoryStructure properly locates the YAML file below.

1. Test Plan foo
2. Nested Test Plan baz
3. Test Suite foo
4. Nested Test Suite name baz
5. Test Case foo
6. Nested Test Case name baz

Command: rspec –format documentation -e 'Factory Structure'

Test Result: PASS

Evidence: Starts on next line.

```
Run options: include {:full_description=>/Factory\ Structure/}

Factory Structure
  Test Plan Name
    is normal.
    is nested.
  Test Suite Name
    is normal.
    is nested.
  Test Case Name
    is normal.
    is nested.

Finished in 0.00592 seconds (files took 0.31934 seconds to load)
6 examples, 0 failures

Coverage report generated for Unit Tests to /home/traap/git/amber/coverage. 502 / 839
   LOC (59.83%) covered.
```

4.1.25 Test Suite: advanced-concept

Purpose: This Test Suite demonstrates a future concept that might be implemented. The concepts include the items below.

1. setup-before-all are Test Steps that are run before all Test Cases.
2. setup-before-each are Test Steps that are run before each Test Case.
3. teardown-after-all are Test Steps that are run before all Test Cases.
4. teardown-after-each are Test Steps that are run before each Test Case.

This Test Suite does demonstrate embedded LaTeX commands and it includes the following Test Cases.

- t001
- t002
- t003
- t005
- t006

Requirement: AMBER-IUR-006

4.1.26 Test Case: t001

Purpose: This Test Case demonstrates embedded LaTeX enumerate and itemized commands.

1. Step #1 uses the Linux echo command.
2. Step #2 will use Linux date command.
3. Step #3 confirms the Linux man program is installed.

Requirement: AMBER-IUR-006

Step: 1

Confirm:
- Program echo has been installed.

Expectation:
- echo installation location is displayed.

Command: sudo which echo

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/echo
```

Step: 2

Confirm:
- Program date has been installed.

Expectation:
- date installation location is displayed.

Command: which date

Test Result: PASS

Evidence: Starts on next line.

```
/usr/bin/date
```

Step: 3

Confirm:
- Program man has been installed.

Expectation:
- man installation location is displayed.

Command: which man

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/man
```

4.1.27 Test Case: t002

Purpose: This Test Case demonstrates embedded LaTeX enumerate command.

1. Step #1 grep check
2. Step #2 dumper check
3. Step #3 sed check
4. Step #4 tr check

Requirement: AMBER-IUR-006

Step: 1

Confirm: Program grep has been installed.

Expectation: grep installation location is displayed.

Command: sudo which grep

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/grep
```

Step: 2

Confirm: Program dumper has been installed.

Expectation: dumper installation location is displayed.

Command: which dumper

Test Result: FAIL

Evidence: Starts on next line.

```
1 which: no dumper in (/home/traap/.rbenv/versions/3.0.3/bin:/usr/lib/rbenv/libexec:/home/
    traap/.local/share/nvim/lsp_servers/sumneko_lua/extension/server/bin:/home/traap/.
    local/share/nvim/lsp_servers/latex:/home/traap/.rbenv/shims:/home/traap/.rbenv/
    versions/3.0.3/bin:/home/traap/git/dotfiles/bin:/home/traap/.bin:/usr/local/sbin:/
    usr/local/bin:/usr/bin:/usr/bin/site_perl:/usr/bin/vendor_perl:/usr/bin/core_perl:/
    var/lib/snapd/snap/bin:/home/traap/.fzf/bin)
```

Step: 3

Confirm: Program sed has been installed.

Expectation: sed installation location is displayed.

Command: which sed

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/sed
```

Step: 4

Confirm: Program tr has been installed.

Expectation: tr installation location is displayed.

Command: which tr

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/tr
```

4.1.28 Test Case: t003

Purpose: This Test Case demonstrates embedded LaTeX enumerate command.

1. Step #1 nice check
2. Step #2 nl check
3. Step #3 man check
4. Step #4 latexmk check
5. Step #5 git check

Requirement: AMBER-IUR-006

Step: 1

Confirm: Program nice has installed.

Expectation: nice installation location is displayed.

Command: sudo which nice

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/nice
```

Step: 2

Confirm: Program nl has installed.

Expectation: nl installation location is displayed.

Command: which nl

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/nl
```

Step: 3

Confirm: Program man has been installed.

Expectation: man installation location is displayed.

Command: which man

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/man
```

Step: 4

Confirm: Program latexmk has been installed.

Expectation: latexmk installation location is displayed.

Command: which latexmk

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/latexmk
```

Step: 5

Confirm: A developer is able to access Git help.

Expectation: Git help is displayed.

Command: git help

Test Result: PASS

Evidence: Starts on next line.

```
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

4.1.29 Test Case: t005

Purpose: This Test Case demonstrates embedded LaTeX enumerate command.

1. Step #1 whois check
2. Step #2 zip check

Requirement: AMBER-IUR-006

Step: 1

Confirm: Program whois has been installed.

Expectation: whois installation location is displayed.

Command: which whois

Test Result: FAIL

Evidence: Starts on next line.

```
1 which: no whois in (/home/traap/.rbenv/versions/3.0.3/bin:/usr/lib/rbenv/libexec:/home/
    traap/.local/share/nvim/lsp_servers/sumneko_lua/extension/server/bin:/home/traap/.
    local/share/nvim/lsp_servers/latex:/home/traap/.rbenv/shims:/home/traap/.rbenv/
    versions/3.0.3/bin:/home/traap/git/dotfiles/bin:/home/traap/.bin:/usr/local/sbin:/
    usr/local/bin:/usr/bin:/usr/bin/site_perl:/usr/bin/vendor_perl:/usr/bin/core_perl:/
    var/lib/snapd/snap/bin:/home/traap/.fzf/bin)
```

Step: 2

Confirm: Program zip has been installed.

Expectation: zip installation location is displayed.

Command: which zip

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/zip
```

4.1.30 Test Case: t006

Purpose: This Test Case demonstrates embedded LaTeX enumerate command.

1. Step #1 yacc check
2. Step #2 xxd check

Requirement: AMBER-IUR-006

Step: 1

Confirm: Program yacc has been installed.

Expectation: yacc installation location is displayed.

Command: which yacc

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/yacc
```

Step: 2

Confirm: Program xxd has been installed.

Expectation: xxd installation location is displayed.

Command: which xxd

Test Result: PASS

Evidence: Starts on next line.

```
1 /usr/bin/xxd
```

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
[A] - Initial version.	New document.
[B] - Section 1 changes.	Reference gSOP.