Amber Tool Validation

Data Management Software Solutions

February 7, 2019

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 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

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.

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.

```
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: 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

```
       Step: 1
    Confirm: Amber properly consumes the command line argument --browser.

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

    Command: echo rspec --format documentation -e 'Amber Browser'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Amber Browser
```

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: echo rspec --format documentation -e 'Amber Case'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Amber Case
```

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 consructs a default Amber::Options object.

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

    Command: echo rspec --format documentation -e 'Amber Defaults'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Amber Defaults
```

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 enviornment option.

    Command: echo rspec --format documentation -e 'Amber Environment'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Amber Environment
```

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: echo rspec --format documentation -e 'Amber File'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Amber File
```

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: echo rspec --format documentation -e 'Amber Language'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Amber Language
```

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: echo rspec --format documentation -e 'Amber NoDryRun'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Amber NoDryRun
```

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: echo rspec --format documentation -e 'Amber Obliterate'
7 Test Result: PASS
8    Evidence: Starts on next line.
9 rspec --format documentation -e Amber Obliterate
```

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: echo rspec --format documentation -e 'Amber Plan'
7 Test Result: PASS
8    Evidence: Starts on next line.
9 rspec --format documentation -e Amber Plan
```

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: echo rspec --format documentation -e 'Amber Simulate'
7 Test Result: PASS
8    Evidence: Starts on next line.
9 rspec --format documentation -e Amber Simulate
```

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: echo rspec --format documentation -e 'Amber Suite'
7 Test Result: PASS
```

```
   Evidence: Starts on next line.
rspec --format documentation -e Amber Suite
```

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: echo rspec --format documentation -e 'Amber Verbose'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Amber Verbose
```

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: echo rspec --format documentation -e 'Amber Version'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Amber Version
```

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: echo rspec --format documentation -e 'Amber Writer'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Amber Writer
```

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: echo rspec --format documentation -e 'YAML Browser Substitutions'
7 Test Result: PASS
8    Evidence: Starts on next line.
9 rspec --format documentation -e YAML Browser Substitutions
```

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: echo rspec --format documentation -e 'YAML Extend Path Substitutions'
7 Test Result: PASS
8    Evidence: Starts on next line.
9 rspec --format documentation -e YAML Extend Path Substitutions
```

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: echo rspec --format documentation -e 'YAML Home Substitutions'
7 Test Result: PASS
8    Evidence: Starts on next line.
9 rspec --format documentation -e YAML Home Substitutions
```

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: echo rspec --format documentation -e 'YAML Language Substitutions'
7   Test Result: PASS
8      Evidence: Starts on next line.
9   rspec --format documentation -e YAML Language Substitutions
```

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: echo rspec --format documentation -e 'YAML Language Code Substitutions'
7   Test Result: PASS
8      Evidence: Starts on next line.
9   rspec --format documentation -e YAML Language Code Substitutions
```

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: echo rspec --format documentation -e 'YAML Strings Substitutions'
7   Test Result: PASS
8      Evidence: Starts on next line.
9   rspec --format documentation -e YAML Strings Substitutions
```

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
```

```
6) Nested Test Case name baz

    Command: echo rspec --format documentation -e 'Factory Structure'
Test Result: PASS
   Evidence: Starts on next line.
rspec --format documentation -e Factory Structure
```

4.2 System Environment

The hyphen character replaced the underscore character, and the forward slash character replaced the backslash character throughout this section.

ALLUSERSPROFILE:

APPDATA:

CLASSPATH: See below.

1. nil

COMPUTERNAME:

COMSPEC:

FP-NO-HOST-CHECK:

GIT-BRANCH: ReqList

HOME: /home/gary

HOMEDRIVE:

HOMEPATH: See below.

1. nil

HOSTNAME:

JRE-HOME:

LANG: en-US.UTF-8

LOCALAPPDATA:

LOGONSERVER:

NUMBER-OF-PROCESSORS:

OLDPWD: /home/gary/git/amber

OS:

PATH: See below.

1. /home/gary/.gem/bin
2. /home/gary
3. /usr/local/sbin
4. /usr/local/bin
5. /usr/sbin
6. /usr/bin
7. /sbin
8. /bin
9. /home/gary/.vim/bundle/vim-superman/bin

PRINTER:

PROCESSOR-ARCHITECTURE:

PROCESSOR-IDENTIFIER:

PROCESSOR-LEVEL:

PROCESSOR-REVISION:
PROFILEREAD:
ProgramData:
PROGRAMFILES:
ProgramFiles(x86):
ProgramW6432:
PSModulePath: See below.

1. nil

PUBLIC:
PWD: /home/gary/git/amber/report
SESSIONNAME:
SHELL: /bin/bash
SHLVL: 2
SYSTEMDRIVE:
SYSTEMROOT:
TEMP:
TMP:
TZ:
USER: gary
USERDNSDOMAIN:
USERDOMAIN:
USERNAME:
USERPROFILE:
WINDIR:

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.