
# State of Tennessee

## Division of TennCare

# TennCare Test Management Standard

Version: 12.0

June 2022

# CONTENTS

# Table of Figures



# Table of Tables

*Table 1: Test Management Standard Version History*

| Version | Description of Change | Date |
|---|---|---|
| 1.0 | Approved by TARB. | March 29, 2016 |
| 2.0 | Minor updates made, approved by TARB. | August 13, 2020 |
| 3.0 | Based on PSM Lessons Learned analysis performed by KPMG | July 2021 |

| | | |
|---|---|---|
| | TAS (Testing team)<br>• Added section: 5.2. Entrance Criteria<br>• Modified sections: 2.2., 3.4.3., 3.5.3., 3.5.4., 3.7.,4.1., 5.1.1., 5.3., 6.2., 7.1., 7.2., 8.1., 8.2., 9.1., 10.1.2., 10.2, 10.3., 10.7. | |
| 4.0 | • Added eTMO workstream to RACI (Section 4.2)Added new activities to RACI under section 4.2 based on Lessons Learned and Defect Analysis performed by eTMO<br>• Modified section 4.2 for RACI assignments as eTMO was included | August 2021 |
| 5.0 | • Added section 4.3. eTMO Project Management Tools ( 4.3.1. RAID Log and 4.3.2. CI Backlog)<br>• Added section: 3.4.4. Data Fixes<br>• Modified sections: 4.2. RACI with eTMO Roles and RACI assignments; 7.1., 10.1.2., 10.2. | September 2021 |
| 6.0 | • Added section 3.7. Scope Freeze and Subsection 3.7.1. Scope Freeze: TEDS Project example<br>• Deleted section 3.4.4. Data Fixes | October 2021 |
| 7.0 | • Added section 3.7.2. UAT Scope Freeze<br>• Added section 5.3.2.2. SIT Regression Acceptance Criteria<br>• Added section 5.3.3.2. UAT Regression Acceptance Criteria<br>• Modified sections 5.1.1., 5.3., 5.3.2.(5.3.2.1.), 5.3.3.(5.3.3.1.) and 9.1. to include SIT and UAT Functional and Regression Phase<br>• Modified section 2.2. Test Management Principles<br>• Modified section 3.5.3. User Acceptance Testing (UAT) Stage<br>• Added section 3.6.2 Production Incidents<br>• Modified sections 9 and 9.1 to include Production Incident re-testing and regression<br>• Modified section 3.8. (Testing Environments) by adding verbiage for Project Release Notes and attaching the template for Release Notes<br>• Modified section 10.2. (Key Terms Glossary) by defining the Release Notes | November 2021 |
| 8.0 | • Modified section 5.3.2. Testing Phase: SIT Acceptance Criteria and section 5.3.3. Testing Phase: UAT Acceptance Criteria | December 2021 |
| 9.0 | • Modified section 7.2 Test Data to align with revisions made by | January 2022 |

| | | |
|---|---|---|
| | the Data Governance Council | |
| 10.0 | • Added Section 8.3 Testing Evaluation Criteria<br>• Modified Section 4.3.1 (RAID Log): added verbiage for eTMO Responsibilities for managing the RAID Log<br>• Modified Section 4.3.2 (Continuous Integration backlog): added verbiage for Agile Board Management: eTMO Continuous Integration Backlog and eTMO Responsibilities for managing the CI Backlog<br>• Modified Section 7.1 (Test Cases, Scripts and Scenarios): added Definition for Test Cases, Scripts and Scenarios along with visual representation<br>• Modified Test data de-identification in Table 13 to include the STAGE environment (production domain only) | February 2022 |
| 11.0 | • Modified Table 12: Solution Test Plan Content to include the number of scripts on incidents for planning purposes, as well as a reference to the incident matrix template in the embedded eTMO Reporting Metrics document<br>• Modified Section 3.7 to include the step-by-step escalation process for incidents introduced after the scope freeze date | March 2022 |
| 12.0 | • Updated to include the Static Regression process in sections 3.6.1, 3.6.2, and 10.3 (Table 22)<br>• Added verbiage to reflect that the TennCare Test Management Standards is a living document that gets revised periodically based on eTMO observations from various projects | June 2022 |

# 1. Summary Diagram

Figure 1 summarizes the elements of this standard. The diagram also appears as Figure 3 and is explained in section 3.1. The righthand arrows provide navigation to their relevant sections within the document.

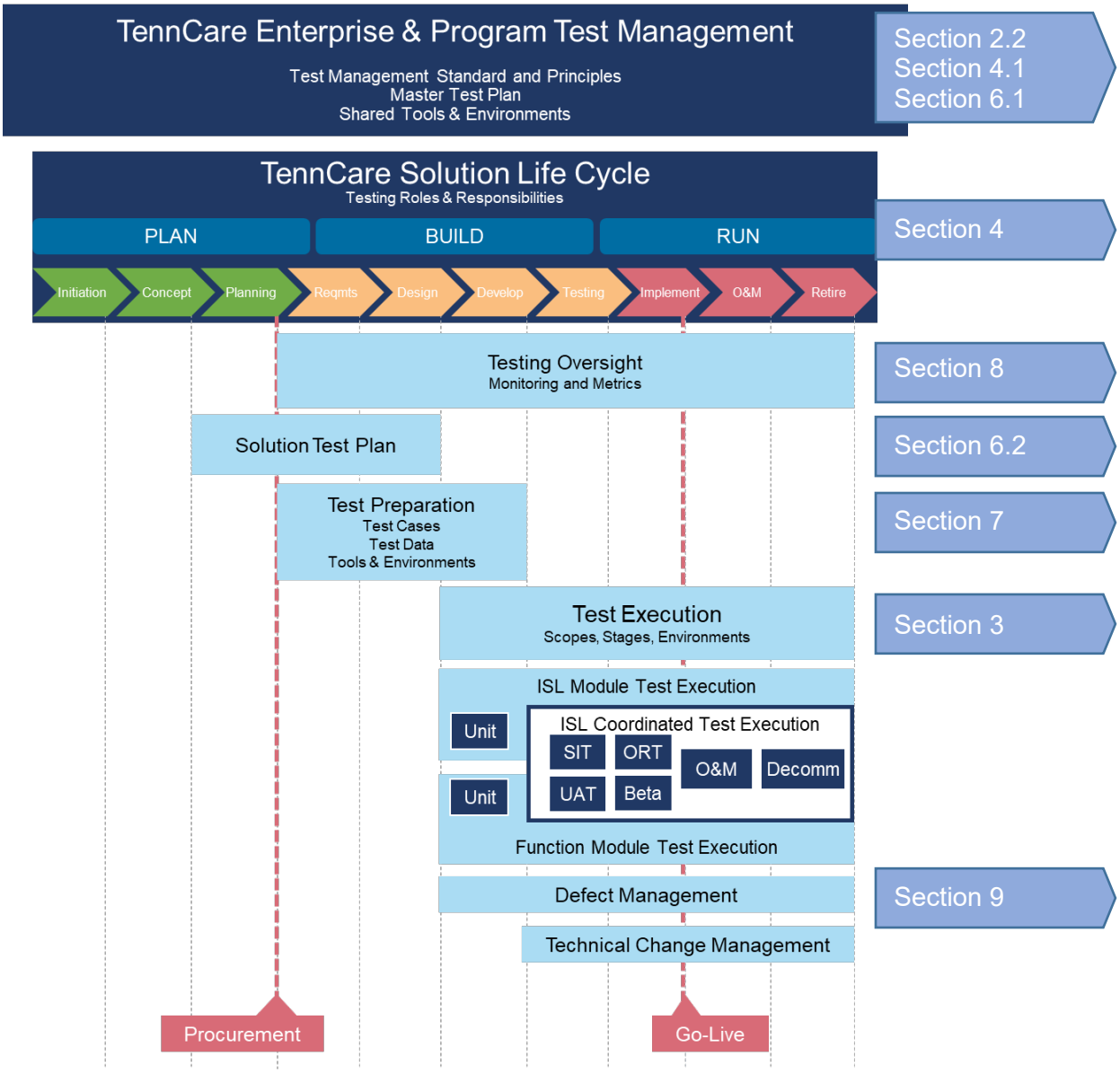



*Figure 1: Testing in the Solution Implementation Lifecycle*

# 2. Introduction

## 2.1. Purpose of this Standard

This standard sets general requirements for testing information systems used at TennCare. These requirements apply to TennCare and vendor partners, with the roles and responsibilities specified herein.

Testing ensures that information systems work as expected and required. This standard will help ensure that the correct testing activities are performed and managed effectively throughout the Solution Implementation Lifecycle for TennCare IS projects.

The TennCare Test Management Standard is a living document that will be updated periodically to address the evolving maturity of testing processes and procedures over time, based on observations by Enterprise Test Management Office on multiple projects under Division of TennCare.

## 2.2. Test Management Principles

This section outlines the guiding principles for Test Management of TennCare solutions. The rationale for these principles, and their implications, are described in appendix section 10.1. These principles build upon the higher level TennCare IS Guiding Principles.

Table 2: Test Management Principles

| Principle | Description |
|---|---|
| Complete Testing | Testing is a process to evaluate whether a developed solution meets specified requirements. Testing also assures that there are no unintended consequences when changes are made to a solution.<br><br>TennCare requires that all solutions be rigorously and completely tested, to assure functionality and adherence to requirements and design. |
| Integrated End-to-End Process Testing | TennCare requires testing of end-to-end business processes across all solutions that need to be integrated, including all function modules and the Integration Services Layer (ISL). |

| Principle | Description |
|---|---|
| Coordinated Release Management Testing | TennCare requires coordinated testing of new releases of the Integrated Services Layer modules and any solution modules that use ISL services. |
| Traceable Defect Management | To facilitate defect management in an integrated modular environment, all solution vendors must be able trace a defect to the requirements that it fails, the test cases that detected it, and the modules containing the defect. |
| Robust Regression Test Management | Regression testing is performed throughout the lifecycle of the system and is performed whenever a component is modified. The goal of regression testing is to ascertain that the modification has not introduced any new faults in the portion that was not subject to modification. TennCare requires a rigorous process for testing fixes and changes to solution modules, as these changes are promoted through various environments and test phases. Changes to other solutions and infrastructure also require regression testing of potentially-affected modules.<br>a) Regression testing is performed during Unit Testing Stage, System Integration Testing (SIT) Stage, and User Acceptance Testing (UAT) Stage and will focus on changed components and related functionality. As the goal of this test is to confirm existing functionality, the recommended approach for this type of regression test is to re-execute existing test cases. These regression tests are identified along with functional tests for the bug or CR.<br>b) A Static regression test repository should also be maintained for execution after all of the SIT and UAT tests have passed. The goal of this regression test is to ensure that production will receive a baseline of functionality that is expected to work, regardless of whether the component or related functionality has changed.<br>Automated regression testing that focuses on validating core set of functionality. Automated testing is geared for consistency and speed of execution, and are used for purposes such as post-migration smoke testing. |

| Principle | Description |
|---|---|
| Transparent Test Reporting | Solution vendors will report on test progress and results to TennCare for transparent governance aligning to TennCare defined test execution and defect metrics reporting cadence and definitions. |

## 2.3. Target Architecture Vision

The target Enterprise Architecture vision for TennCare is to separate Information Systems functions into multiple modules, each provided through a separate vendor contract. Releasing any new or upgraded module will require testing it in a complex multi-module, multi-vendor context. Automation of one business function will typically involve multiple modules, therefore the integration of solutions needs to be tested.

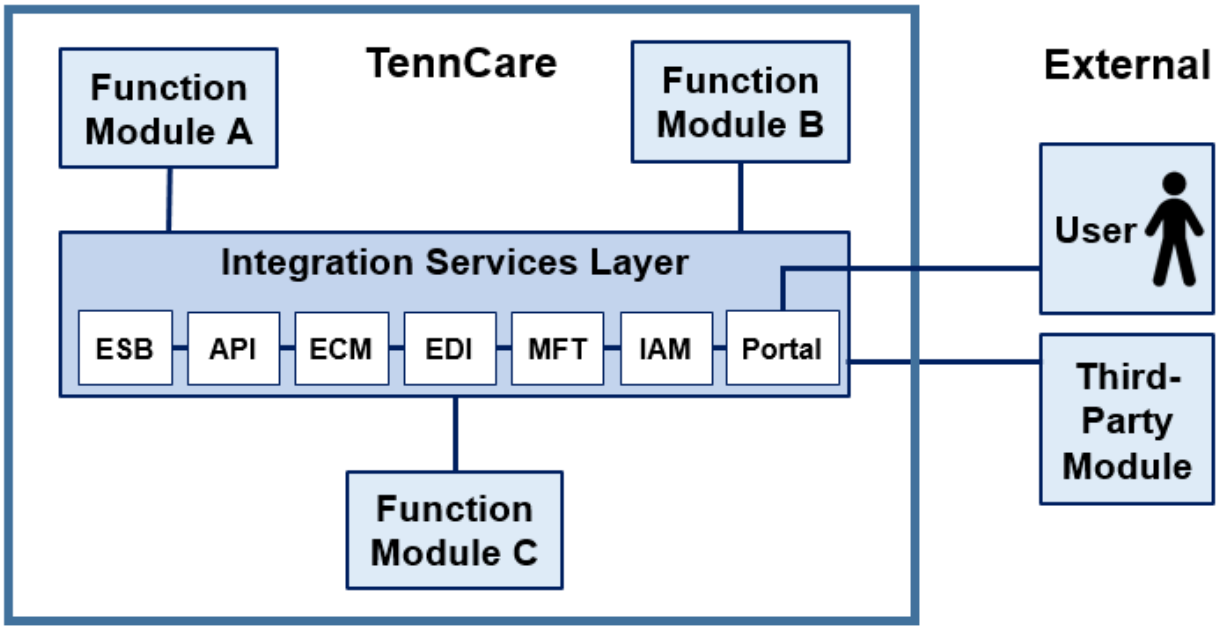



Figure 2: Function Modules and ISL Components

The following terminology is used within this standard:

TennCare information systems include many **solutions**. A solution is a combination or configuration of computer application software and infrastructure components that automates some processes within a business function. A **solution project** delivers a new or upgraded solution.

A solution may consist of one or more **modules**. Each module may be a commercial software product or custom-built code. These modules are integrated, configured, and customized to form a solution. This standard discusses “function modules” and “ISL components”: This standard also applies to a non-modularized solution, which may be treated as one module, or may have multiple components that can be treated as modules.

**Function modules** perform a business function, such as Eligibility and Enrollment or Financial Management. These are represented in Figure 2 as Function Modules A, B, and C.

The Integration Services Layer (ISL) is a planned solution, composed of the “ISL component”: technology tools that move data between modules and/or control access to data by users and third-party modules. Figure 2 shows acronyms representing ISL components:

- ESB: Enterprise Service Bus
- API: Application Programming Interface Gateway / Web Services
- ECM: Enterprise Content Management
- EDI: Electronic Data Interchange
- MFT: Managed File Transfer
- IAM: Federated Identity and Access Management
- Portal: Horizontal Portal

One business process may involve multiple users, third parties, function modules, and ISL components. Therefore, all of these modules need to be connected in order to send information between them.

In the target architecture, function modules **connect with** the ISL, meaning that there are interfaces connecting the function module to any number of the ISL components. Function modules should not connect directly to other function modules. Instead, all function modules should connect through the ISL in a hub-and-spoke structure.

There are also interface connections between modules within the ISL, which are represented in a simplified fashion in Figure 2.

External users may interact with TennCare through a “Horizontal Portal” in the ISL providing the user interface for function modules. Third party systems will also interact through the ISL with function modules.

A **release** of a module is a version of module code, configuration, or customization, delivered at one time. There may be multiple releases of a module over time, at a frequency determined by business requirements and the solution vendor’s methodology.

Both function and ISL components may be hosted in the cloud, or on-premises (by Strategic Technology Solutions on behalf of TennCare). Third-party business service providers may also have application modules hosted in the cloud or on their own premises. Testing must work across these various deployments.

## 2.4. Scope and Applicability

This standard applies to TennCare IS projects of any size or complexity, including function modules and ISL components as defined in section 2.3.

In this standard, the IS Vendor plays a coordination and support role, defined in section 4. The ISL procurement provides additional documentation about the IS Vendor's role and responsibilities as a solution vendor.

This standard also applies to testing solutions that are not connected to the Integration Services Layer. Statements about "function modules" also apply to these solutions.

This standard covers testing within Release Management of new or upgraded solutions. This includes tests executed during Development, Testing, and Implementation phases of the Solution Implementation Lifecycle (before Go-Live). Within these phases, after resolving a defect, the tests may go through the Change Management process.

This standard (section 3.6.1) covers testing of new releases of solutions that are in the Operations & Maintenance phase of the lifecycle. It assumes that tests developed in release management may be used by TennCare's Change Management process for new versions of infrastructure software and hardware, and fixes to systems in production at TennCare and third parties.

This standard covers testing of the procedures for decommissioning legacy systems. These procedures are designed along with a new/upgraded solution, as discussed in section 3.6.2.

## 2.5. Precedence of Documents

This standard sets general requirements for all TennCare systems and projects. More specific plans and requirements are set for each system by solution projects.

The provisions of vendor contracts with TennCare take precedence over this standard.

The TennCare Solution Implementation Lifecycle Standard takes precedence over this standard.

Exceptions to this standard may be granted through a recorded decision of a TennCare governance body, including Program/Project Steering Committees and the TennCare Architecture Review Board (TARB).

In case of any ambiguity within this standard, the RACI chart (section 4.2) shall take precedence over other text and diagrams.

## 2.6. Referenced Documents

Table 3: Referenced Documents

| Document | Sections Referencing Document |
|---|---|
| TennCare IS Guiding Principles | 1.2 |
| TennCare Solution Implementation Lifecycle Standard | 1.5, 2.1, 3.1 |
| TennCare Enterprise Architecture Framework | 5 |
| TennCare Requirements Management Standard | 5 |
| TennCare Technology Standard | 7.1 |

# 3. Testing Lifecycle, Stages and Scopes

## 3.1. Enterprise and Program Test Management

Testing needs to be managed across the program of solutions being integrated (as described in section 2.3), and across the whole enterprise of TennCare. The Test Management Principles (section 2.2) articulate this need for testing business processes from end-to-end, across multiple solution modules.

This program- and enterprise-wide test management is based on this standard and a Master Test Plan (section 6.1). The Master Test Plan includes a master schedule for coordinating the testing activities for multiple releases.

Testing multiple solutions requires some shared environments, described in section 3.7. TennCare may have one or more shared tools or repositories for testing or test management, as discussed in section 7.3.

Testing of a solution relies upon the above enterprise test management activities. Responsibilities for these activities are specified in section 4.2.

## 3.2. Testing in the Solution Lifecycle

This diagram shows how testing activities fit within the Solution Implementation Lifecycle of a solution project that delivers a new release of a function module or ISL module.

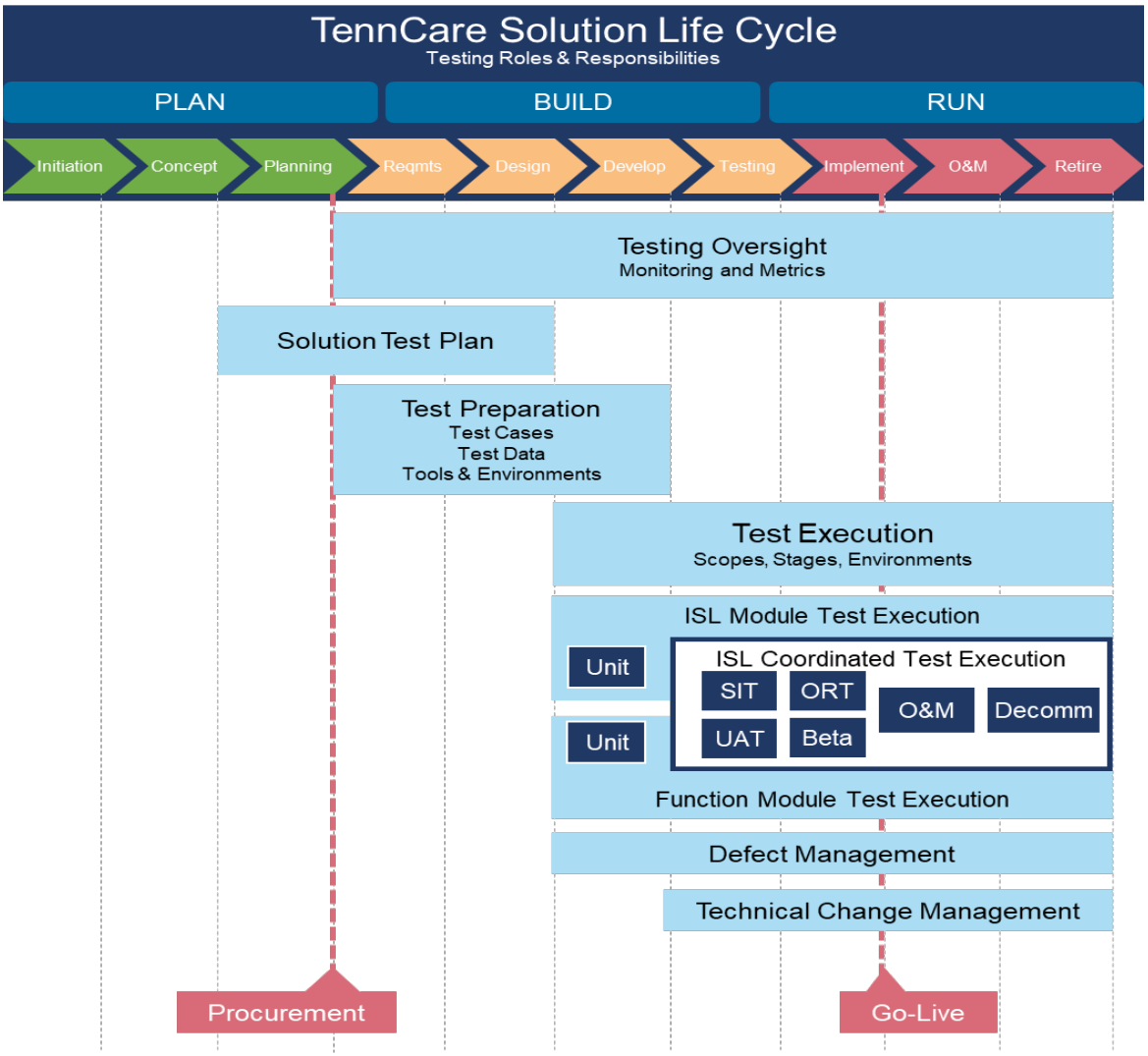



Figure 3: Testing a New Release in the Solution Implementation Lifecycle

The solution lifecycle of a new release is described in the TennCare Solution Implementation Lifecycle Standard, and shown in Figure 3: Testing a New Release in the Solution Implementation *Lifecycle*. Each chevron indicates a phase ending in an approval gate. For certain gates, there are testing-related activities and deliverables.

All testing activities of all solutions are subject to this Test Management Standard, as well as Master (enterprise-level or program-level) test planning, strategy, and reporting.

Testing activities for one module are subject to solution-level test planning, strategy and reporting.

Planning for testing starts in the Planning phase of a solution's lifecycle, with the development of a Request for Proposals (RFP). RFPs solicit solution vendors that can meet TennCare's requirements including its testing needs as described in this standard. Solution vendors must

finalize their Solution Test Plan (section 6.2) in the Requirements Review and Elaboration phase.

Preparation for testing happens once the vendor is on-board. During the Requirements Review and Elaboration, Design, and Development phases, test cases, data, tools, environments, and personnel are prepared for tests to be executed in the subsequent phases.

Tests are executed during the Development, Testing, and Implementation phases. Obtaining approval for completed tests allows the project to pass through those gates, to the point of Go-Live.

Five testing stages are illustrated and defined in section 3.5: Unit Testing, System Integration Testing, User Acceptance Testing, Operational Readiness Testing, and Beta Testing. Unit testing is executed separately by the solution vendor of the function or ISL module. At all remaining stages, modules integrated with the ISL need testing to be coordinated with the IS Vendor.

When the solution is in production, tests are required for technical changes made during that Operations & Maintenance phase. To retire a system, further testing is required of the decommissioning procedures (shown as Decomm in Figure ).

## 3.3. Test Execution Stages and Scopes

The following table summarizes the kinds of testing that need to be executed during the Solution Implementation Lifecycle.

The testing scopes are illustrated in section 3.4 and the testing stages are defined in section 3.5, with the environments required at each stage defined in section 3.8.

Each stage ends by delivering a Test Report defined in section 5.1.1. The Test Report must be accepted before moving to the next stage of testing, except point-to-point System Integration Testing, which may begin before Unit Testing is complete. Delivery and acceptance of the Test Reports is also required for passing gates to move to the next phase in the Solution Implementation Lifecycle. See the acceptance criteria in section 5.2.

These testing stages may be repeated in regression testing and in testing during the Operations & Maintenance phase (section 3.6.1) and Retire phase (section 3.6.2).

Table 4: Test Execution Stages, Scopes and Environments

| **Solution Implementation Lifecycle phase** | **Development** Develop | **Testing** Testing | | **Implementation** Implement | |
|---|---|---|---|---|---|
| **Test stage** | **Unit** | **SIT** System Integration Testing | **UAT** User Acceptance Testing | **ORT** Operational Readiness Testing | **Beta** |
| **Testing by** | Solution vendor | Solution vendors<br>IS Vendor | TennCare | Solution Vendors<br>IS Vendor<br>TennCare | TennCare |
| **Scope** | Module testing within 1 solution | Point to Point | - | Point to Point | |
| | | End-to-End | End-to-End | End-to-End | End-to-End |
| **Requirements** | Functional<br>Non-functional | Functional<br>Non-functional | Functional<br>Non-functional | -<br>Non-functional | Functional<br>Non-functional |
| **Regression** | Regression | Regression | Regression | Regression | Regression |
| **Environment** | DEV | SIT | STAGE | PREPROD<br>DR | PROD or PREPROD |
| **Deliverable** | Unit Test Report | System Integration Test Report | User Acceptance Test Report | Operational Readiness Test Report | Beta Test Report |

## 3.4. Testing Scopes

### 3.4.1. Module Testing

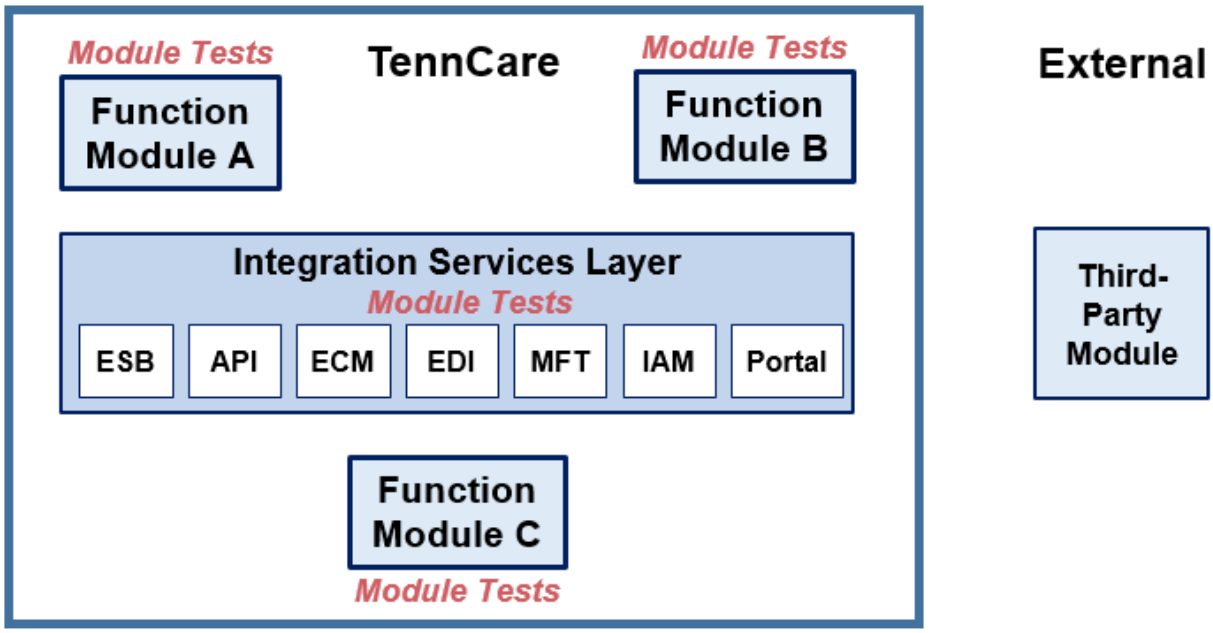



Figure 4: Module Testing Scope

Each module needs to be tested separately, before integrating it with other modules. Module-specific tests are needed within each solution project, as shown in Figure . Module testing is needed for function modules, and for ISL components within the ISL, and for solutions not being integrated with the ISL.

The Module Testing scope is entirely covered by the Unit Testing stage, section 3.5.1.

Module testing includes testing within one vendor's solution project. See section 10.2 for definitions of these terms.

This standard covers the individual module testing of TennCare function and ISL components. Responsibilities for module testing are specified in the RACI chart, section 4.2.

This standard assumes that third parties test their own modules before integration with TennCare systems. This standard does not specify how they do module testing.

### 3.4.2. Point-to-Point Testing

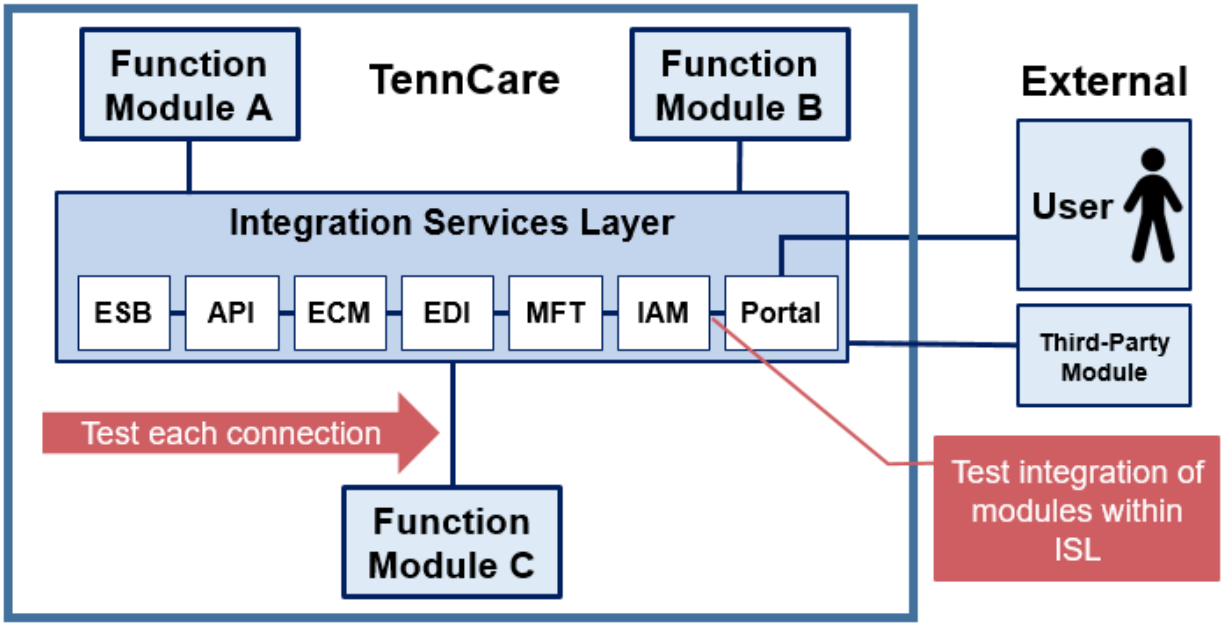



Figure 5: Point-to-point Testing Scope

After testing each module successfully, the connection or interface between each pair of modules needs to be tested. Figure  is a simplified representation of the point-to-point testing needed:

- Between each function module and each of the integration (ISL) modules it will connect to
- Between each pair of ISL components that will interact within the ISL
- Between third-party modules and the ISL components
- Between the computing environments of external users (e.g. common web browsers), and the Portal or any other user interfaces.

Point-to-point testing also includes testing any connections:

- Between modules or components of one solution
- Between solutions that do not go through the ISL.

Each point-to-point test could involve one or two vendors. Connections across two different deployments (cloud or on-premises hosting) must be tested.

- The two modules may be at different phases of their respective implementation lifecycles. For example, a module in development may need to test its connection to a module that has already been implemented. Test planning should anticipate the need for a test version of the implemented module, to avoid doing tests in a Production environment.
- If the solution design requires a connection to a module that has not yet been developed, a “dummy module” or “stub” must be created to conduct the testing.

Point-to-point testing is done at the SIT and ORT stages, as defined in section 3.8. To avoid project delays, point-to-point SIT testing may begin before the Development phase gate is passed.

Point-to-point testing is a technical activity, not normally involving business users.

Point-to-point testing must be conducted for any integration of a TennCare solution or system with a third-party module or system. Agreements between TennCare, its vendors, and the third party must specify who conducts point-to-point testing and who sees the testing results.

Point-to-point tests include:

1. Connectivity smoke test:
   - Can the two modules talk to each other? Does data flow through the pipe?
2. Functionality test
   - Can the two modules do the required tasks together? Does the expected data come out of the pipe?
3. Non-functional tests of the connection for performance, security, etc.

### 3.4.3. End to End Testing

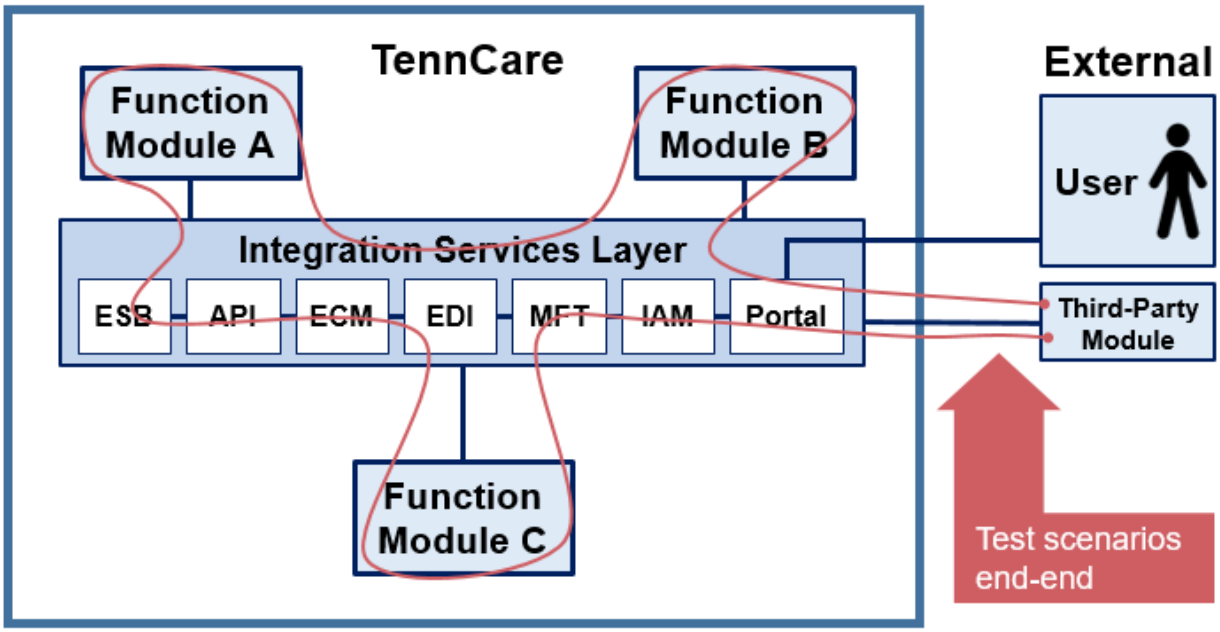



Figure 6: End-to-end Testing Scope

Business processes require “end-to-end testing” to ensure that all steps in a process can be successfully executed through the many modules and connections involved. These tests verify that the actual solution modules and interfaces, as developed and configured, function together as intended.

The curved red line in 6 represents a business scenario, which is modeled in solution architecture, to describe how each business process happens. Information moves between the users, external parties, function modules and ISL components in a sequence of steps defined by the scenario. An end-to-end scenario may also include solutions that are not connected to the ISL.

End-to-end testing is performed after module testing and point-to-point testing. It is performed during the SIT, UAT, ORT, and Beta stages (section 3.8). End-to-end testing may be executed by business users, technical staff, or test automation tools.

End-to-end testing must be conducted for any integration of a TennCare solution or system with a third-party module or system. Agreements between TennCare, its vendors, and the third party must specify who conducts end-to-end testing and in which phases it will be conducted. TennCare requires approval of all end-to-end test cases prior to execution. End-to-end test scenarios should be constructed and loaded into the test management tracking tool in a manner that allows for reporting down to the test step level when multiple steps are involved in the test case. End-to-end testing should be completed with no major blocking defects before the UAT testing phase to ensure any TennCare or third pary resources involved in UAT testing efforts are able to complete all business scenarios during the UAT testing phase. TennCare requires a formal approval of the certification process of end-to-end testing results. When planning end-to-end testing, consider privacy and security requirements, such as how sensitive data from one module will be handled in other modules.

## 3.5. Testing Stages

The new release of any solution goes through the following stages of testing, which are summarized in Table 4.

Regression testing of changes and fixes requires repeating some tests at each stage. Later changes to a solution may have an accelerated path through the stages, as determined by the Technical Change Management process.

### 3.5.1. Unit Testing Stage

At the Unit Testing stage, the solution vendor performs Module Testing (section 3.4.1) within their module or component. The Unit Testing stage includes:

- Testing functional requirements of the module
- Testing non-functional requirements (performance, security, etc.) at the module level
- Testing, by one solution vendor, of the integration of multiple components to form one module
- Testing compliance requirements at the module level
- Regression testing of the module, after defect resolution or other changes

Unit testing does not include:

- Testing of integrations with other modules of the solution (see SIT, section 3.5.2)
- Testing of integrations with other solutions or vendors (see SIT, section 3.5.2)
- Testing by users (see UAT, section 3.5.3)

If the module has a user interface, Unit testing also includes:

- Usability testing
- Accessibility testing for compliance to Section 508 of the Rehabilitation Act of 1973

If the user interface will be provided by another module, usability, and accessibility testing are included in unit testing of that other module.

If a module (such as commercial off-the-shelf software) has been tested previously, those previous results may be used in the Unit Test Report. Conduct additional tests as needed to show that the module fulfills TennCare's requirements and that any TennCare customizations or configurations are functional.

Regression testing of one module may be triggered by changes in another module or interface, before or after go-live. Automating a suite of regression tests for the module is recommended.

When planning unit testing, solution vendors should consider conducting many types of testing defined in section 10.2:

- Functional testing should include positive tests, negative tests, exploratory tests, sanity tests, error handling tests, boundary tests, time-travel testing, and alert/monitoring tests.
- Non-functional testing should include performance, capacity, volume, stress, load, recovery, reliability, scalability, security, and penetration tests.
- Usability testing (when included in unit testing) should include ease of use, localization, accessibility, GUI design and navigation.

When unit testing is complete and the Unit Test Report is accepted (see section 5.3.1), the code, configuration, and customization is promoted to the next stage and its environments. After that promotion, defect resolution or other changes may prompt further unit testing.

To avoid project delays, the point-to-point System Integration Testing may begin before the Unit Test Report is complete and accepted.

### 3.5.2. System Integration Testing (SIT) Stage

System Integration Testing (SIT) is a technical activity to ensure that a module meets requirements for integration with other modules including the ISL. It does not involve business users.

In SIT, functional and non-functional requirements must be tested with both point-to-point (section 3.4.2) and end-to-end (section 3.4.3) testing.

Therefore, the SIT stage has multiple sub-stages. The columns of Table 5 break down SIT by what the module is integrating with. The rows break down SIT by scope of testing:

Table 5: System Integration Testing sub-stages

| | | Integration of modules within one solution | Integration of module with ISL | Integration of module with other solutions |
|---|---|---|---|---|
| **Point-to-point testing**<br><br>of all interfaces and connections to or from the module | Connectivity smoke test | Each module can connect to each other module within the solution | Module can connect to ISL | Module can connect to other solutions |
| | Functional tests | Each connection within the solution functions as designed | Each connection to/from ISL functions as designed | Each connection to another solution functions as designed |
| | Non-functional tests | Each connection within the solution meets non-functional requirements | Each connection to ISL meets non-functional requirements | Each connection to another solution meets non-functional requirements |
| **End-to-end testing**<br><br>of all defined scenarios involving the module | Functional tests | All scenarios function as designed, across all integrations of the solution modules, the ISL and other solutions | | |
| | Non-functional tests | The integration of solution modules, the ISL and other solutions meets non-functional requirements | | |

To avoid project delays, the point-to-point connectivity testing may begin before the Unit Testing stage is completed.

Regression testing of the module will repeat some SIT tests. Regression testing of one module may be triggered by defect resolution or changes in another module or interface, before or after go-live. Automating a suite of SIT regression tests is recommended.

When planning SIT, vendors should conduct as many of the types of testing defined in section 10.2 as appropriate.

- Functional testing should include positive tests, negative tests, exploratory tests, smoke tests, sanity tests, error handling tests, boundary tests, time-travel tests, and alert/monitoring tests
- Non-functional testing should include performance, capacity, volume, stress, load, recovery, reliability, scalability, security, role-based-access, and penetration tests

When SIT is complete and the System Integration Test Report is accepted (see section 5.3.2 and 5.3.12), the code, configuration, and customization are promoted to the next stage and its environments. After that promotion, defect resolution or other changes may prompt further System Integration Testing.

### 3.5.3. User Acceptance Testing (UAT) Stage

User Acceptance Testing (UAT) is a business activity to ensure that a module meets business user needs, in concert with other integrated modules.

The scope of UAT is end-to-end testing (section 3.4.3) which includes both functional and UAT regression. Users (TennCare representatives) are asked to test defined test cases and requirements.

UAT Regression testing includes validation of a module that might have been affected by a change in another module or interface. A baseline set of UAT Regression test scripts are also executed once all functional tests have passed (see Table 2 in section 2.2, item b).

When TennCare has formally accepted the tested module, UAT is complete. When the User Acceptance Test Report is accepted (see section 5.3.3), the code, configuration, and customization is promoted to the next stage.

After that promotion, any defect resolution or other changes will prompt re-testing the UAT tests of the scenarios involving the change. Training materials should updated after any changes to functionality or when knowledge gaps are identified. The training materials should be utilized for UAT and training sessions. All training materials should be approved by TennCare and TAS OCMT prior to training sessions .

### 3.5.4. Operational Readiness Testing (ORT) Stage

Operational Readiness Testing (ORT) is a technical activity to ensure that a module, integrated with other modules, complies with non-functional requirements. It does not involve business users.

The ORT stage has two sub-stages, conducted in this sequence:

4. Point-to-point testing (section 3.4.2) of all interfaces and connections to or from the module
5. End-to-end testing (section 3.4.3) of all defined scenarios involving the module

Non-functional testing includes but is not limited to these types of testing defined in section 10.23:

- Performance Test
- Capacity Test
- Availability Test
- Volume Test
- Stress Test
- Load Test
- Reliability Test
- Scalability Test
- Recovery Test
- Accessibility Test
- Security Assessment Test
- Compliance Test
- Vulnerability Test
- Role-Based Access Test
- Penetration Test
- Disaster Recovery Test

The non-functional requirements will be specified in the vendor's contract, a Service Level Agreement, or the Solution Design. Those documents will also specify which ORT tests are to be executed by the solution vendor. Some ORT must be executed by TennCare IS, STS, another infrastructure provider, or specialized testing resources. Test planning ensures enough time in the project schedule for the various tests to be done.

Regression testing of the module will repeat some ORT tests. Regression testing of one module may be triggered by defect resolution or other changes in another module or interface, before or after go-live. The utilization of testing tools (such as Splunk) that captures and analyzes logs both within the suite of applications and between the modules and upstream/downstream interfaces should beuseful. Automating a suite of ORT regression tests is recommended.

When ORT is complete and the Operational Readiness Test Report is accepted (see section 5.3.4 and 5.3.1), the code, configuration, and customization are promoted to the next stage and its environments. After that promotion, defect resolution or other changes may prompt further Operational Readiness Testing.

### 3.5.5. Beta Testing Stage

Beta testing enables TennCare representatives to test the full and completed solution, prior to formal Go-Live. Business and/or IS users will ensure that a module works as expected, in concert with other integrated modules.

Beta testing is also known as pilot testing, or parallel testing. The scope of Beta testing is end-to-end testing (section 3.4.3) of the full solution (including all its modules and interfaces). The

scenarios normally encountered in daily operations are tested, rather than a set of formal test cases. During Beta testing, users may discover gaps or defects in functionality, interface integration problems, data conversion errors, access control, slow performance, or other non-functional issues.

If Beta testing is a parallel run with an existing system, it is conducted in the PROD (production) environment, with converted data and live interfaces. Automation, such as scripts, may be required to keep the existing and new systems (and their data) synchronized. If Beta testing is not a parallel run, it may be conducted in the PREPROD or PROD environments.

When the users have formally accepted the tested module, Beta testing is complete. The decision to Go-Live is made as described in the Solution Implementation Lifecycle Standard. After Go-Live, defect resolution or other changes may prompt retesting and regression testing that includes a Beta testing stage.

## 3.6. Testing After Go-Live

### 3.6.1. Testing in the O&M Phase

Testing continues in the Operations and Maintenance (O&M) phase of the Solution Implementation Lifecycle. Testing of a solution is required in the following circumstances:

- Releasing an update to the solution, after Go-Live
- Fixing a defect in a solution, after Go-Live
- Change is proposed to another module or system that might affect the solution
- Change is proposed to infrastructure software, such as a patch, fix, or new version of database software, which might be used by the solution

All of these changes are governed by the Technical Change Management process. A change is initiated by a solution vendor, TennCare IS or another party. In the change request, they document the anticipated impact on other systems, and the stages and types of testing required.

Test cases/scripts/scenarios need to be written and executed to test the solution components that will change.

To test a change, applicable testing stages from Unit to Beta may need to be executed, depending on the impacts identified in the Technical Change Request. The testing deliverables (section 5) are required at each stage of testing, though they may be simplified for smaller changes. The test data and environments for a new release will need to be maintained and refreshed after Go-Live to conduct O&M testing of changes.

### 3.6.2. Regression Testing

Regression tests are performed on components, modules, interfaces and systems to help ensure that existing functionality is not adversely impacted by the change. Regression tests are either: (a) specific to the changes or fixes that were made for the release, or (b) static regression tests that are performed regardless of functionality in that area of the application has changed or not.

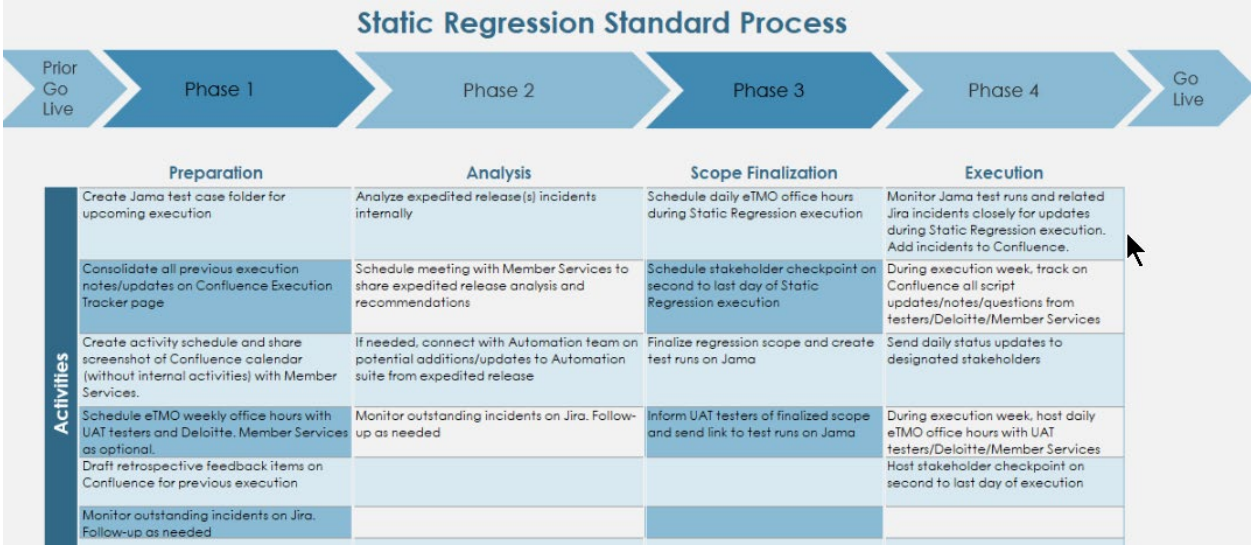
#### 3.6.2.1. Regression Testing

The regression tests that are release-specific are determined by the module solution vendor (but not precluding optional Ad Hoc tests) and are related to functionality currently being implemented. These dynamic (or functional) regression tests will vary according to the scope of the release.

#### 3.6.2.2. Static Regression Testing

Static Regression testing serves to validate application core functionality that is expected to work, regardless of whether the component or related functionality has changed in a release. The Static Regression Suite is executed for Major and Minor releases. Static regression testing is fixed, so it is not impacted by the scope of the release. The coverage of the Static Regression Suite may evolve at the request of business stakeholders. Changes to the Static Regression Suite can stem from prior defects, Expedited Releases, or application functionality updates. Expedited Releases may influence the addition of new scripts to the Static Regression or Automation Suite, or updates to existing scripts in either Suite.

Static Regression testing will follow a standard process consisting of four phases. The first phase focuses on preparation and communication with business stakeholders. In the second phase, Expedited Releases are analyzed to determine potential additions or updates to the Static Regression suite and Automation suite. In the third phase, Static Regression scope is finalized and shared with designated stakeholders. In the fourth phase, the Static Regression suite is executed. Once execution is complete, TennCare will provide confirmation to move forward with Go-Live.




#### 3.6.2.3. Automation Testing

Additions or updates to the Automation Suite may occur after a review of an Expedited Release.

### 3.6.3. Production Incidents

Production incidents typically require prompt attention due to undesirable business impacts. Production incidents, once properly triaged, are resolved and re-tested through the Defect Management Process described in section 9.

At the time of creation, high-level Incident-Retest Criteria should be recorded on the original production incident by the creator. This should include steps to reproduce the issue, any variations of these steps that could cause the same issue, and related areas of the application that might also be affected by the conditions that uncovered the incident.

The Incident-Retest Criteria should be referenced to create any new scenarios needed to re-test the fix. New scenarios related to the incident should be added to the project's test repository, with traceability to the incident; these scenarios may become regression candidates. The original incident should be updated with a reference to the test scenarios executed for verification (functional and regression), as well as the test results.

### 3.6.4. Testing in the Retire Phase

The Retire phase of the Solution Implementation Lifecycle is triggered when a new solution is designed to replace part or all of a legacy system.

The new system's Solution Design artifacts will include specifications for migration from the legacy system to the new solution. It must show how functions will be migrated and how data will be converted, before turning off the legacy system's functions.

A System Disposition Plan will specify the steps in decommissioning part or all of the legacy system. A technical Change Request will control the decommissioning, so this testing is a variant of O&M Testing above.

The decommissioning and migration steps in the System Disposition Plan and new Solution Design need to be tested, in concert with each other. All of the testing stages shall be executed:

Table 6: Testing migration and decommissioning

| Testing Stage | Migration to new solution | Decommissioning legacy system |
|---|---|---|
| Unit testing | | Decommissioning steps that are isolated to the legacy system |
| System Integration Testing | Test data conversion and other migration from legacy to new solution | Decommissioning steps that involve or affect other systems |
| User Acceptance Testing | Users test the results of migration | Users test the results of decommissioning |
| Operational Readiness Testing | Non-functional tests of scripts, etc., to be used in migration<br><br>Non-functional tests of the integrated systems after migration | Non-functional tests of scripts, etc., to be used in decommissioning<br><br>Non-functional tests of the integrated systems after decommissioning |
| Beta testing | Execute the migration, on a pilot basis | Execute the decommissioning, on a pilot basis |
| | Legacy system and new solution may be run in parallel | |

All of the activities in the RACI chart (section 4.2) need to be done, during the new solution's lifecycle phases. Testing of migration shall be included in the new solution's testing deliverables (section 5).

To test migration from a legacy system to a new solution, shared testing environments will be required.

Testing of decommissioning is the responsibility of the legacy system's manager (such as a solution vendor with ongoing maintenance responsibilities). Testing the migration to a new solution is the responsibility of the new solution vendor. These parties need to cooperate to plan, prepare, and execute the appropriate testing.

## 3.7. Scope Freeze

During SIT, the application can be dynamic, due to substantial code changes resulting from defects identified during the cycle. This increases the need for retesting areas previously tested. By contrast, the application should be somewhat static throughout the UAT phase. To further support test stability, scope freeze dates must be planned prior to the start of testing (SIT and UAT). Any code changes made after the freeze date due to unplanned Change Requests (CRs) or production incidents may cause instability in the environment, unpredictable test results, and ultimately lead to leaked defects. Because of this, unplanned scope change proposals made after the freeze date must be vetted through a risk assessment process involving three perspectives: Development, Quality Assurance, and the Business. A step-by-step escalation process for introducing incidents after the freeze date can be found in the following document:

SOP for Incident added after scope Fre

### 3.7.1. Scope Freeze: TEDS Project example

For every major release of the TEDS project, there are two planned scope freeze dates: a CR freeze date, and an Incident freeze date. Similarly, for every minor release, there is single scope freeze date. In all cases, it is crucial that these dates are strictly adhered to, and any proposed scope changes made after these freeze dates must follow the step-by-step process referenced in Section 3.7 to account for any additional development and test efforts needed, if the change is to be implemented. It is commonly understood that the timing and impact of production incidents are unpredictable, so an incident triage meeting must be in held to make crucial business decisions regarding the need for and feasibility of scope changes.

In advance of these business-oriented production incident triage meetings, the development, SIT and UAT impact of the scope change must be fully analyzed collaboratively by the respective teams, and the corresponding effort of implementing a fix must be estimated. The SIT and UAT impact requires careful planning, with a time-based estimate, down to the number and duration of test scripts, including re-executions. SIT and UAT representatives should bring this position to the incident triage meetings with the business, so that a proper risk assessment can be performed around the scope change, with all perspectives considered. Once the incident has been approved at the triage meeting, these details must be also be sent to the UAT Manager for approval. Visually, the process is depicted in Figure 7: Triage process for unplanned scope change after freeze date

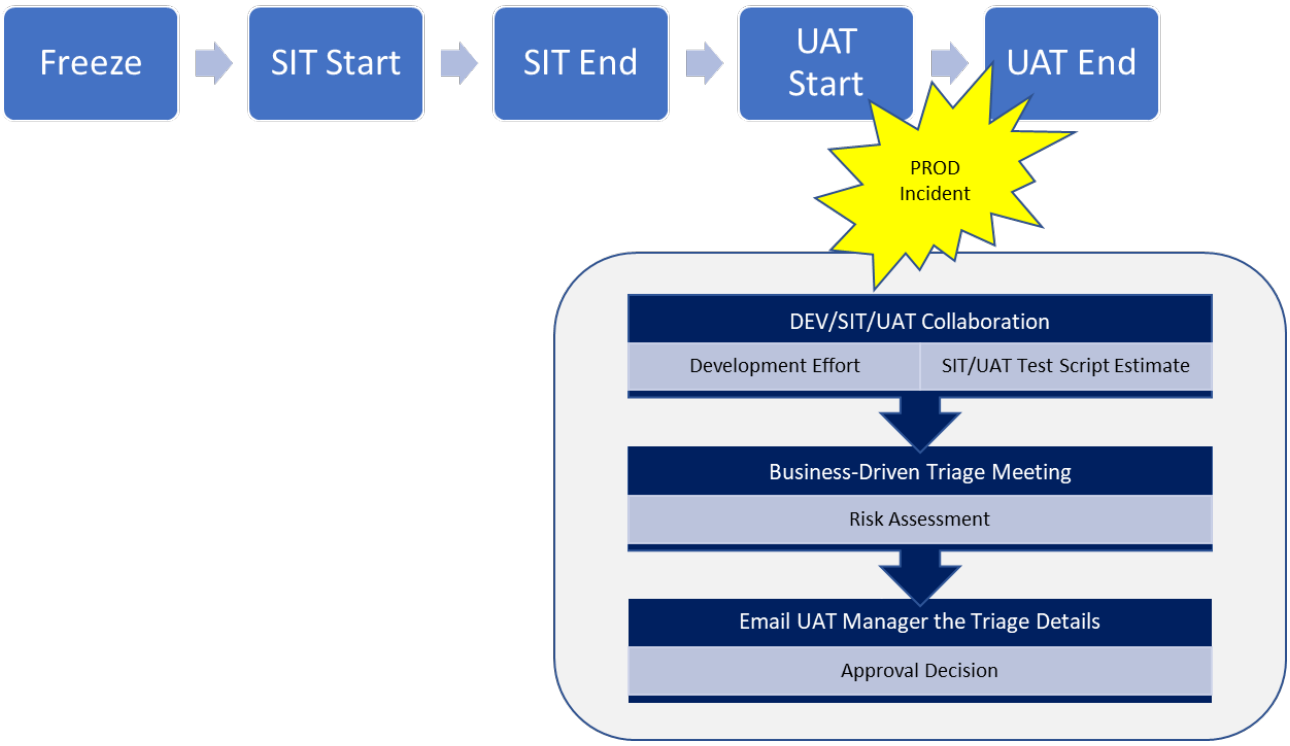



Figure 7: Triage process for unplanned scope change after freeze date

### 3.7.2 UAT Scope Freeze

For testing individual project features, the UAT team inherits a set number of functional scenarios to execute from the SIT team at least one week prior to the start of UAT. To help ensure more consistent and reliable testing, the number of scenarios identified at the start of UAT should ideally remain static throughout the cycle. To that end, the UAT Scope Freeze date should align with the main Scope Freeze dates mentioned in Section 3.7.

If de-scoping tests within a UAT cycle is imperative, this must be communicated through the proper channels: UAT Leads should be officially notified with an e-mail that fully describes the proposed change, and any related impacts. The e-mail should articulate the reason for de-scoping the tests, the analysis that led to the de-scoping decision, a traceabilty report to other impacted areas of the application, as well as any impacts on regression testing. If any additional script executions are required for coverage in related areas or for regression purposes, those scripts must be clearly identified in the notification as well.

## 3.8. Testing Environments

The following extract from Table 2: Test Management Principles shows the minimum environments needed for testing a new solution release in the integrated TennCare

environment. Additional environments may be requested to TennCare for approval as necessary, such as time travel environments.

Table 7: Testing Environments by Stage

| Test stage | Unit<br><br>Unit Testing | SIT<br><br>System Integration Testing | UAT<br><br>User Acceptance Testing | ORT<br><br>Operational Readiness Testing | Beta<br><br>Beta Testing |
|---|---|---|---|---|---|
| **Environment** | DEV | SIT | STAGE | PREPROD<br><br>DR | PROD or PREPROD |
| **Shared?** | Separate | Shared | Shared | Shared | Shared |
| **Test Data** | De-identified | De-identified | De-identified and Real | De-identified and Real | Real |

Solution vendors may set up and manage their own environments, separate from other vendors, as required for their project, for Module Testing, training, data conversion, or other purposes.

Solution vendors (including the IS Vendor) need to use shared hybrid cloud environments for SIT, UAT, ORT, and Beta testing of integrated modules. Responsibilities for shared environments are specified in section 4.2. For TennCare, hybrid cloud environments will include a mix of cloud, on-premises, and third-party hosting. To test migration from a legacy system to a new solution, shared testing environments will be required.

Unit testing is done within the solution vendor's own environment(s), such as a **DEV environment**.

System Integration Testing across multiple modules is done on a shared **SIT environment**.

User Acceptance Testing is done on a shared **STAGE environment**. It may be done with de-identified and/or real data.

Operational Readiness Testing is done on a shared **PREPROD environment**. Additional environments may be needed, such as a **DR environment** for Disaster Recovery testing. ORT may be done with de-identified and/or real data.

Beta testing may be done in the PREPROD environment. With careful preparation, Beta testing may be done in the **PROD environment**, meaning the production system with real data, just before go-live.

Regression testing is done in the same environments where the tests were originally done. For example, if an interface is changed during the ORT stage, regression testing starts in the SIT environment and flows through UAT, STAGE, and PREPROD.

The environments using real data need TennCare controls and audits for privacy and security. The approval of the Solution Test Plan will include TennCare authorization of any vendor personnel having access to real data.

During test planning, vendors will be required to provide planned code deployment schedules so that the schedules are coordinated and optimized to have minimal interruption during test execution. In addition, a notification of start and deployment completion, along with smoke testing completion and release notes (defined in section: 10.2.) should be published to notify stakeholders that the environments are ready. This secures a confirmation and avoids miscommunications in case there are delays in deployment activities.

Product release is as significant and important as any other stage of Software Development Life Cycle (SDLC). It is this stage that represents the whole development and testing process of the software and validates its quality, functionality, effectiveness, performance and more. Release note is an essential part of this stage and is used to deliver all the relevant information about the project and the completed software to Business Stakeholders and the UAT Team.

Release notes is a document, which is prepared by Module Solution Vendor/ SIT Team and is shared with UAT testers and Business Stakeholders as part of the project release that contains new enhancements (Change Requests) and the known issues (Incidents). Documenting the release of the product is as indispensable as the development or testing process.

The content of release notes varies according to the release type. For the State of Tennesse Project, a Release Notes document should include the following elements/sections like Project Name and Description, Project Start and Release Date, SIT/UAT Start and End Date, Project Release Lead Information(like Lead Name and Email information), Expected verses Actual Script Count, Count of SIT scripts moved into UAT, Release Notes Summary (High Level Summary for Change Requests and Incidents that are included for that project), For each Change Requests and Incidents Individual include a detail summary and an Excel Attachment (.xlsm or .xlsx file). This attached Excel File to the Release Notes Document should contain list of each individual Change Requests and Incidents with detailed information, and must include a section for Severity that should be assigned by the Module Solution Vendor Team before providing the release notes document to TennCare Business and TennCare IS. Based on the Severity assigned by the Module Solution Vendor, TennCare Business and TennCare IS members will then assign the Priority to those items/scripts.

TennCare Release Notes draft.docx

To support O&M testing and Decommissioning, it is recommended to maintain the testing environments after Go-Live.

# 4. Roles and Responsibilities

## 4.1. Roles

This section describes the roles and responsibilities of stakeholders involved in testing activities. Provided below is a high level description of each stakeholder's responsibilities. As solution projects progress through the phases outlined in the TennCare Solution Implementation Lifecycle Standard, stakeholders take on various responsibilities, as detailed in section 4.2.

Table 8: Roles and Responsibilities

| Role | Responsibility |
|---|---|
| TennCare Business | TennCare Business is an organizational unit that oversees the policies and operations of the Division of TennCare business functions, such as member services. In test management, they manage the resources required for test activities, particularly acceptance testing, and provide the final software approval as they are the end user of solutions.<br><br>This role includes any vendor engaged by TennCare to execute UAT and/or Beta tests, as a representative of the business. |
| TennCare Information Services (TennCare IS) | TennCare Information Services provides support for planning, design, implementation and operation of information technologies and methods. In test management, TennCare IS will collaborate with TennCare Business to coordinate testing activities across all projects and provide approval for all plans related to defect resolution and mitigation as well as the final software approval.<br><br>TennCare IS also has a dispute-resolution role: it assigns responsibility to particular solution vendors for defect resolution among the integrated set of software modules. |
| Program & Project Management | The Program & Project Management function manages programs and projects for TennCare. In test management, they are responsible for aligning testing activities with project management and the solution lifecycle. They monitor testing through reporting. |
| TennCare Program/Project Steering | The TennCare Program/Project Steering Committees for each program or project is comprised of sponsors and key stakeholders |

| | |
|---|---|
| Committees (TennCare PSC) | of the respective program or project. This body performs gate reviews and approves testing deliverables, as part of its mandate. TennCare business and/or IS may be represented on the PSC.<br><br>In the RACI chart, the Program & Project Management column includes PSC approvals and activities. |
| TennCare Architecture Review Board (TARB) | The TARB governs this Test Management Standard and reviews test software tools against the TennCare Technology standard. TARB may also review and approve technical deliverables on behalf of Program and Project Management.<br><br>In the RACI chart, the TennCare IS column includes TARB reviews and approvals of deliverables presented by a vendor. |
| Change Advisory Board (CAB) | The CAB governs the Technical Change Management process as described in the TennCare IT Service Management Standard.<br><br>In the RACI chart, the TennCare IS column includes CAB reviews and approvals of changes presented by a vendor. |
| Strategic Technology Solutions (STS) | STS is a State of Tennessee agency that provides direction, planning, resources, execution, and coordination in managing the information systems needs of the State of Tennessee. STS is a division within the Department of Finance & Administration. In test management, STS supports testing and integration of technologies on the environments they provide, which are currently on State Data Center premises.<br><br>On the RACI chart, the STS (Infra) column applies to STS and any other infrastructure provider. |
| Cloud Infrastructure Provider | The Cloud Infrastructure Provider is a vendor that provides cloud hosting and infrastructure. Applications to be deployed in the cloud will need test environments in the cloud.<br><br>On the RACI chart, the responsibilities of STS (Infra) also apply to the Cloud Infrastructure Provider. |
| Module Solution Vendor | The Module Solution Vendor develops a Function Module. This role applies to any vendor developing a solution other than the ISL, regardless of whether the solution is modularized or integrated with |

| | |
|---|---|
| | the ISL.<br><br>In test management, the Module Solution Vendor is responsible for:<br><br>• Testing the solution that they develop and deliver to TennCare<br>• Collaborating with TennCare and other vendors to test the module in the integrated environment<br>• Providing a test plan and test reports for TennCare approval<br>• Providing timely testing support to answer inquiries during testing, review defects during triage, and resolve defects |
| Integration Services (IS) Vendor | The IS Vendor is responsible for delivering its integration services modules and the vendor should ensure all members of the the team are educated/knowledgeable and capable of providing the quality insights into the module/product.It also has a role to support and coordinate testing of other solutions that use its integration services, which includes:<br><br>• Providing test cases, scripts, and scenarios for integration services<br>• Executing tests to ensure connectivity between modules and integration services with existing and future systems<br>• Recording and resolving defects<br>• Reporting and coordination for the Master Test Plan and Solution Test Plans<br>• Providing timely testing support to answer inquiries during testing, review defects during triage, and resolve defects<br><br>The IS Vendor is provided with an addendum that specifies their responsibilities for setting up the integration services layer. This standard only describes the IS Vendor's coordination with other module solution vendors.<br><br>The Vendor is responsible for providing a list of required key resources necessary for integration activities. The vendor should list and name primary contact information for all areas of integration to avoid delays in execution by the testers. Sufficient backup resources should be allocated and included in the list as well. |
| Third-Party Business Services Vendor | Third Party Business Services Vendors include Service Partners such as the Managed Care Organizations (MCOs).<br><br>The Third-Party Vendor is responsible for initiating tests with TennCare when the Partner change their own software. They are |

| | |
|---|---|
| | also responsible for participating in the testing of new solutions and integrations. |
| Technical Advisory Services (TAS) | TAS supports and advises the State in completing the TennCare IS program by offering Organizational Change Management and Training, Operations & Maintenance Planning, System Development Life Cycle Guidelines and Governance, Quality Management, and Enterprise Architecture services. |
| eTMO | <u>Test Strategy and Governance</u><br><br>Standards: Define and maintain testing standards, processes and procedures that are efficient and repeatable.<br><br>▪ Assist in the formulation of testing timelines and alignment<br>▪ Conduct and manage Release Scope Analysis<br>▪ Assit in maintaining the Master Test Scenario List<br>▪ Maintain and manage the Release Coverage Matrix<br><br>Budgeting: Develop and maintain the required financial tracking process across the testing function.<br><br>▪ Assist in the formulation of viable budget models to support various teams and tools for testing budget management<br>▪ Develop and maintain viable/consistent processes for reporting work costs<br>▪ Develop and maintain a reporting package that will enable detailed, custom, at-will financial trending and budget reporting<br><br>Test Policy and Governance: Assist in the development and maintenance of testing processes and procedures for:<br><br>▪ Requirements Management<br>▪ Test Monitoring and Control<br>▪ Defect and Incident Management<br>▪ Configuration and Environment Management<br>▪ Release and Project Management<br><br>Build Vendor Collaboration: Work with ISL and Module vendors to identify the appropriate collaborative operating model; specifically in regards too:<br><br>▪ Life expectancy of the test model |

| | |
|---|---|
| | ▪ Software development model<br>▪ Maturity of Code<br>▪ Release model<br>▪ Compliance and Regulatory Requirements<br><br><u>Test Management and Control</u><br><br>Test Strategy and Planning:<br><br>▪ Assist in the identification of components and associated test cases that need to be tested<br>▪ Guide the implementation of processes and procedures to mitigate testing risks<br>▪ Create and maintain a capacity & demand model to help manage the dynamic staffing requirements of UAT test projects<br>▪ Create and maintain a project/release tracker that captures pertinent project details up front and helps guide SIT and UAT test efforts from inception<br><br>Program and Project Coordination:<br><br>▪ Identify risks and issues to QA/testing and assit in the development of appropriate mitigations<br>▪ Provide recommendations for improvement<br>▪ Provide a consistent and insightful approach to proactively managing and monitoring the testing program throughout its life cycle<br><br>Continuous Assessment and Improvement<br><br>▪ Develop pragmatic solutions for an effective and efficient testing program<br>▪ Conduct reviews and adopt measures to evaluate faster time to completion with a standardized, repeatable, and efficient testing process<br>▪ Assist in the mitigation of excessive code rework and redundant testing costs via more detailed explanation of defects<br><br>Risk and Issue Management:<br><br>▪ TAS testing design takes a risk-based approach to testing early and often |

| | |
|---|---|
| | Identify and raise key risks and issues that the testing team will face when replicating production environments<br><br><u>Test Delivery Operations</u><br><br>Requirements Traceability:<br><br>▪ Assist in the verification and static analysis of requirements to ensure correctness<br>▪ Review Requirements Traceability Matrix to validate requirements through review, inspection and audit<br>▪ Assit in the validation of requirements to test case mappings for every test type and phase to ensure coverage and traceability<br>▪ Assist in the validation of change request/scope changes and ensure requirements matrix is updated<br>▪ Apply a risk based approach to maintain test cases and matrix for prioritization and impact analysis<br><br>Test Data and Environment Management: Assist in the formulation of a robust test data management strategy, includinng:<br><br>▪ Assit in the collection of appropriate data<br>▪ Recognition that single data sets will encompass multiple scenarios<br>▪ Detailed domain knowledge is critical to not “boil the ocean”<br>▪ Assist in alignmenet of test environment management processes to the in-scope test phases and components leading up to the end-to-end test<br><br>Defect Management:<br><br>▪ Delivery Quality Assurance Leaders are attached to subprojects to assist in the adherence to standards, enable continuous improvement and defect prevention<br>▪ Utilize defect management repositories where defects identified during testing will be recorded and managed<br><br>Test Maintenance:<br><br>▪ Assist in the maintenance of test cases relevance and facilitate updates when there is a change to requirements and/or functionality<br><br>Assist and facilitate updates to requirement traceability matrix to |

| | |
|---|---|
| | manage changes to requirements and or development |

## 4.2. Responsibilities and Approvals (RACI Chart)

The following table specifies who must do what, when, to complete testing within the lifecycle of a new or updated TennCare IS solution. The responsible Roles are defined in section 4.1.

In the RACI chart, where the activity is to “approve” a deliverable, the A indicates who approves it, and the R indicates who requests the approval. All approvals shall be documented in a test management repository, by the party Accountable for approving, with approval information specified in section 5.2. Approvals that complete a phase gate review shall also be documented by the Program & Project Management Office (PMO).

When multiple solutions are going through development and implementation, the latest (newest) solution vendor will be responsible for testing its modules’ connectivity to the ISL and its integration with the previously connected modules.

The attached spreadsheet is the RACI chart applicable to function modules and Module Solution Vendors. For those modules, the IS Vendor plays the support and coordination role described in section 4.1. A supplementary IS Vendor RACI chart will be provided as part of procuring the ISL, to show the IS Vendor’s responsibility of delivering and testing the ISL modules.

Table 9: RACI Chart

| | Planning Phase | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Update | Deliverable | Activity | TennCare Business | TennCare IS | Program & Project | STS (Infra) | IS Vendor | Module Solution Vendor | TAS | Business Support Services | eTMO |
| U | | Ensure testing needs are within the vendor contract | C | A | I | C | - | - | C | C | R |
| U | Request for Proposal | Ensure testing needs are within the RFP | C | A | C | C | - | - | C | C | R |
| **Requirements Review Phase** | | | | | | | | | | | |

| Update | Deliverable | Activity | TennCare Business | TennCare IS | Program & Project | STS (Infra) | IS Vendor | Module Solution Vendor | TAS | Business Support Services | eTMO: TDO* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| U | Requirements Traceability Matrix | Update the Requirements Traceability Matrix with necessary test information | I | A | C | I | I | R | C | C | C |
| U | | Ensure all requirements are testable | I | A | C | I | I | I | C | C | R |
| U | | Set up any shared test management tools and repositories | I | A | C | I | R | I | C | C | C |
| U | | Acquire, stand-up, and configure tools for managing test cases and defects for module testing | I | A | I | I | I | R | I | C | C |
| N | | Provide strategic guidance to the eTMO team to meet committed scope and objectives | I | A | I | I | C | C | C | I | R |
| N | | Manages eTMO Analysts and provides oversight to TDO Leads | I | A | I | I | C | C | C | I | R |
| N | | Reviews and facilitates “phase gate” meetings | C | A | I | I | C | C | C | I | R |
| N | | Oversight of tools and processes that accurately capture and report on testing metrics | I | A | C | C | C | C | C | C | R |
| N | | Publish the standards for metrics and KPIs | I | A | C | C | C | C | C | C | R |
| N | | Publish standard metrics and reporting views | I | A | C | C | C | C | C | C | R |
| N | | Prepare materials for “phase gate” meetings | C | A | I | I | I | C | C | I | R |
| N | | Coordinates across vendors to ensure everyone adheres to the TennCare Defect Management Standards | C | A | I | I | I | C | C | I | R |
| Design Phase | | | | | | | | | | | |

| Update | Deliverable | Activity | TennCare Business | TennCare IS | Program & STS (Infra) | IS Vendor | Module Solution Vendor | TAS | Business Support Services | eTMO: TDO* |
|---|---|---|---|---|---|---|---|---|---|---|
| R | Solution Test Plan | Develop a SIT Solution Test Plan that details the approach, tools, resources, and activities for managing the schedule, scope, test environments, and test data throughout the life of the project | C | A | C | I | C | R | I | C | C |
| N | Solution Test Plan | Develop a UAT Solution Test Plan that details the approach, tools, resources, and activities for managing the schedule, scope, test environments, and test data throughout the life of the project | C | A | C | I | R | C | I | C | C |
| U | Solution Test Plan | Coordinate with the State Test Manager and any owners of data to appropriately capture the processes and activities needed to manage test data in the Solution Test Plans | I | A | C | I | C | R | C | C | C |
| U | Solution Test Plan | Review all test plans, ensuring they will test all modules and the integration between them | C | A | C | I | I | C | C | C | R |
| U | Solution Test Plan | Add to the Solution Test Plan by clarifying how the ISL will support Reporting and Coordination | I | A | C | I | C | R | C | C | C |
| R | Solution Test Plan | Approve SIT Test Plan | C | R/A | C | I | I | C | I | C | C |
| N | Solution Test Plan | Approve UAT Test Plan | C | R/A | C | I | C | C | I | C | C |
| U | | Create all test cases, scripts, and scenarios (including test cases for UAT) | C | A | I | C | R | R | C | I | C |
| U | | Maintain a shared central repository of test cases, scenarios, and scripts | I | A | I | I | R | R | C | C | C |
| R | | Facilitate cross functional quality assurance reviews for all new test cases, scenarios, and scripts and solicit approval from TennCare business stakeholders | I | A | I | I | I | I | C | C | R |
| R | | Facilitate cross functional quality assurance reviews for all new test data and solicit approval from TennCare business stakeholders | I | A | I | I | I | I | C | C | R |
| U | | Confirm the build, configuration, and deployment of test environments | I | A | I | C | C | R | C | I | C |
| U | | Solution vendor testing personnel on-boarded and trained | I | A | I | C | I | R | I | I | C |
| U | | UAT testing personnel on-boarded and trained | I | A | I | C | I | R | I | I | C |
| U | | Module test data ready and approved | I | A | I | I | I | R | I | I | C |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| U | | Shared test data ready and approved | I | A | I | I | R | I | I | I | C |
| U | | Vendor's own testing tools & repositories set up | I | A | I | I | I | R | I | I | C |
| U | | Grant access for shared test management tools to Solution Vendor personnel | I | A | I | I | R | I | C | I | C |
| N | | Works across all stakeholder groups to manage and guide testing efforts and drive consistent methods and approach | C | A | I | I | C | C | I | I | R |
| N | | Facilitates the planning and oversight of testing scope, schedule, risks, and interdependencies | C | A | C | I | C | C | I | I | R |
| N | | Facilitates the planning and control of the testing scope, schedule, risks, and interdependencies across all projects | C | A | C | I | C | C | I | I | R |
| N | | Initial and period inspection of SI Vendor / Supplier delivery model, test plan and test schedules, etc., to ensure alignment to the TennCare Test Management Standards and testing specific contractual terms, metrics and KPIs | C | A | I | I | C | C | I | I | R |
| N | | Prepare Executive Reports on the overall testing progress | I | A | I | I | C | C | I | I | R |
| N | | Establish testing metrics and testing targets for the project | C | A | I | I | C | C | I | I | R |
| N | | Oversee the creation and management of UAT resource capacity and demand models. | C | A | I | I | C | C | I | I | R |
| N | | Develop and maintain project/release tracker for UAT efforts | C | A | I | I | C | C | I | I | R |
| N | | Oversee and provide review analysis of test plans and test procedures for all components at the unit, module, system and integration levels, performing risk analysis when required. | C | A | I | I | C | C | I | I | R |
| N | | Formulate the data communication and architecture of the communications | C | A | C | C | C | I | I | C | R |
| N | | Oversee and provide review analysis for all test data strategies and activities | C | A | C | I | C | C | I | C | R |
| N | | Work closely with all participating systems stakeholders to determine the Integrated test schedule | C | A | C | I | C | C | I | I | R |
| Development Phase | | | | | | | | | | | |

| Update | Deliverable | Activity | TennCare Business | TennCare IS | Program & Project | STS (Infra) | IS Vendor | Module Solution Vendor | TAS | Business Support Services | eTMO: TDO* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| U | Unit Test Report | Execute Unit tests and document results | C | A | I | I | I | R | I | C | C |
| U | Unit Test Report | Open and resolve unit test defects. Comprehensive process for Defect Management found in the defect management section | C | A | I | I | C | R | I | C | C |
| U | Unit Test Report | Provide mitigation plans for defects that cannot be fixed. Review & approve | A | C | I | I | I | R | I | C | C |
| U | Unit Test Report | Obtain Approval/signoff of test results | C | A | I | I | I | R | I | I | C |
| U | | Support Tenncare's oversight of testing progress, quality, reports, schedules, and metrics | I | A | I | I | I | I | I | I | R |
| U | | Support Tenncare's oversight of shared test environments & promotion of code | I | A | I | I | I | I | I | I | R |
| U | | Prepare shared environments for SIT and UAT | I | A | I | I | R | I | C | I | C |
| N | | Collaborate with development and test team to migrate only the resolved defects | C | A | C | I | I | C | I | I | R |
| Testing Phase | | | | | | | | | | | |
| Update | Deliverable | Activity | TennCare Business | TennCare IS | Program & Project | STS (Infra) | IS Vendor | Module Solution Vendor | TAS | Business Support Services | eTMO: TDO* |
| U | System Integration Test Report | Execute SIT test and document results | C | A | I | I | C | R | I | I | C |
| U | System Integration Test Report | Confirmation verification of non-functional requirements | C | A | I | C | C | R | I | I | C |
| U | System Integration Test Report | Open and resolve SIT test defects. Comprehensive process for defect Management found in the defect management section | C | A | I | I | C | R | I | C | C |
| U | System Integration Test Report | Provide mitigation plans for defects that cannot be fixed. Review and approve. | C | A | I | I | C | R | I | C | C |
| U | System Integration Test Report | Obtain Approval/signoff of test results | C | A | I | I | C | R | I | I | C |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| U | System Integration Test Report | Regression testing after Defect Resolution | C | A | I | C | C | R | I | C | C |
| U | User Acceptance Test Report | Arrange and administer the UAT tests for TennCare business to execute | C | A | I | I | I | R | I | I | C |
| U | User Acceptance Test Report | Execute UAT test | C | A | I | I | R | C | I | C | C |
| U | User Acceptance Test Report | Document UAT test results | C | A | I | I | R | C | I | I | C |
| U | User Acceptance Test Report | Open and resolve UAT test defects. Comprehensive process for defect Management found in the defect management section | C | A | I | I | C | R | I | C | C |
| U | User Acceptance Test Report | Provide mitigation plans for defects that cannot be fixed. Review and approve. | A | C | I | I | C | R | I | C | C |
| U | User Acceptance Test Report | Obtain Approval/signoff of test results | C | A | I | I | R | C | I | I | C |
| U | User Acceptance Test Report | Regression testing after Defect Resolution | A | I | I | C | C | R | I | I | C |
| R | | Support Tenncare's oversight of testing progress, defect management, and testing specific risk/issue management, quality, reports, schedules, and metrics | C | A | I | I | C | C | C | C | R |
| U | | Support Tenncare's oversight of shared test environments & promotion of code | I | A | I | I | I | I | C | I | R |
| U | | All defects recorded in Shared Test Management Tool. | I | A | I | I | C | R | I | I | C |
| U | | Document and re-test all defect resolutions, perform regression testing in conjunction with all defect remediation releases | C | A | I | I | C | R | I | C | C |
| U | | Coordinate SIT and UAT testing across integrated modules | I | A | I | I | R | C | I | I | C |
| U | | Update the Master Testing Schedule | I | A | I | I | R | C | I | I | C |
| U | | Assign responsibility for defects between modules; resolve disputes | C | A/R | I | I | C | C | I | I | C |
| U | | Coordinate between vendors to do Technical Change Management activities | I | A | I | I | R | I | C | I | C |
| N | | Coordinates End of Day Reporting | I | A | I | I | C | I | C | I | R |
| N | | Create and maintain a standard set of reporting templates on test metrics for UAT and SIT | I | A | I | I | C | I | I | I | R |
| N | | Prepare and disseminate SIT reports to key stakeholders using the standard metrics template | I | A | I | I | R | C | I | I | C |
| N | | Prepare and disseminate UAT reports to key stakeholders using the standard metrics template | I | A | I | I | C | R | I | I | C |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N | | Retrieve test data daily to understand the trends on all phases of testing | I | A | I | I | C | C | I | I | R |
| N | | Consolidate data from vendors and generate portfolio reports on the testing progress, trends and the potential risks as a result of deviation from the scheduled plan | C | A | C | I | C | C | I | I | R |
| N | | Identify risks/concerns around SIT/UAT and work with stakeholders on a mitigation plan | C | A | C | I | C | I | C | C | R |
| N | | Monitor testing efforts to promote consistent methods and approach | C | A | I | I | C | C | I | I | R |
| N | | Monitor the planning and control of the testing scope, schedule, risks, and interdependencies | C | A | C | I | C | C | I | C | R |
| N | | Distribute End of Day SIT Reporting | I | I | I | I | C | R | I | I | A |
| N | | Distribute End of Day UAT Reporting | C | C | I | I | R | C | I | I | A |
| N | | Participate in daily testing and defect control meetings. | I | A | I | I | C | C | I | I | R |
| N | | Participates in defect triage calls to manage assignments of new defects and follow up on aging issues | C | A | I | I | C | C | I | I | R |
| N | | Help ensure that all new defects are discussed and impact assessment conducted before they are assigned | C | A | I | I | C | C | I | I | R |
| N | | Analyze defect trends and report these to stakeholders for decision making | C | A | I | I | C | C | C | I | R |
| N | | Develop defect trends and develop heat maps on aging defects and highlight areas of concern | C | A | I | I | C | C | C | I | R |
| Implementation Phase | | | | | | | | | | | |
| Update | Deliverable | Activity | TennCare Business | TennCare IS | Program & Project STS (Infra) | IS Vendor | Module Solution Vendor | TAS | Business Support Services | eTMO: TDO* | |
| U | Operational Readiness Test Report | Execute ORT test and document results | C | A | I | I | C | R | I | C | C |
| U | Operational Readiness Test Report | Organize ORT tests that must be done by others and document results | C | A | I | I | C | R | I | C | C |
| U | Operational Readiness Test Report | Execute ORT tests that must be done by TennCare IS | C | A/R | I | I | C | C | I | C | C |
| U | Operational Readiness Test Report | Execute ORT tests that must be done by infrastructure provider | C | A | I | R | C | C | I | C | I |
| U | Operational Readiness Test Report | Open and resolve ORT test defects. Comprehensive process for defect | C | A | I | I | C | R | I | C | C |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Management found in the defect management section | | | | | | | | | |
| U | Operational Readiness Test Report | Provide mitigation plans for defects that cannot be fixed. Review and approve. | A | C | I | I | C | R | I | C | C |
| U | Operational Readiness Test Report | Obtain Approval/signoff of test results | C | A | I | I | I | R | I | I | C |
| U | Operational Readiness Test Report | ORT regression testing after Defect Resolution | A | C | I | C | I | R | I | I | C |
| U | Operational Readiness Test Report | Organize ORT regression tests that must be done by others and document results | C | A | I | I | C | R | I | I | C |
| U | Beta Test Report | Arrange and administer the Beta tests for TennCare business to execute | C | A | I | I | I | R | I | I | C |
| U | Beta Test Report | Execute Beta test | R | A | I | I | I | C | I | C | C |
| U | Beta Test Report | Document Beta test results | C | A | I | I | I | R | I | I | C |
| U | Beta Test Report | Open and resolve Beta test defects. Comprehensive process for defect Management found in the defect management section | C | A | I | I | C | R | I | C | C |
| U | Beta Test Report | Provide mitigation plans for defects that cannot be fixed. Review and approve. | A | C | I | I | C | R | I | C | C |
| U | Beta Test Report | Obtain Approval/signoff of test results | C | A | C | I | I | R | I | C | A |
| U | Beta Test Report | Beta Regression testing after Defect Resolution | A | C | I | C | I | R | I | C | C |
| R | | Support Tenncare's oversight analysis of module vendor test execution progress, quality, reports, schedules, and metrics | I | A | I | I | I | I | I | I | R |
| U | | Support Tenncare's oversight of shared test environments & promotion of code | I | A | I | I | I | I | I | I | R |
| U | | All defects recorded in Shared Test Management Tool. | I | A | I | I | C | R | I | I | C |
| U | | Document and re-test all defect resolutions, perform regression testing in conjunction with all defect remediation releases | C | A | I | I | C | R | I | C | C |
| U | | Coordinate ORT and Beta testing across integrated modules | I | A | I | I | R | C | I | I | C |
| U | | Update the Master Testing Schedule | I | A | I | I | R | C | I | I | C |
| U | | Assign responsibility for defects between modules; resolve disputes | C | A/R | I | I | C | C | I | I | C |
| U | | Coordinate between vendors to do Technical Change Management activities | I | A | I | I | R | I | C | I | C |
| Operations & Maintenance Phase | | | | | | | | | | | |

| Update | Deliverable | Activity | TennCare Business | TennCare IS | Program & Project | STS (Infra) | IS Vendor | Module Solution Vendor | TAS | Business Support Services | eTMO: TDO* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| U | | Execute O&M test and document results | C | A | C | I | C | R | I | I | C |
| U | | Open and resolve O&M test defects. Comprehensive process for defect Management found in the defect management section | C | A | C | I | C | R | I | I | C |
| U | | Provide mitigation plans for defects that cannot be fixed. Review and approve. | A | C | C | I | C | R | C | C | C |
| U | | Obtain Approval/signoff of test results | C | A | C | I | I | R | C | I | C |
| U | | Regression testing after Defect Resolution | A | C | I | C | C | R | I | C | C |
| U | | Perform Smoke Testing after each new release | C | A | C | C | C | R | I | I | C |
| U | | Perform Root Cause Analysis for any defects found as part of regression testing | C | A | C | C | C | R | I | I | C |
| U | | Support Tenncare's oversight of testing progress, quality, reports, schedules, and metrics | I | A | I | I | I | I | C | I | R |
| U | | Support Tenncare's oversight of shared test environments & promotion of code | I | A | I | I | I | I | C | I | R |
| U | | Document and re-test all defect resolutions, perform regression testing in conjunction with all defect remediation releases | C | A | I | I | C | R | C | C | C |
| U | | Update the Master Testing Schedule | I | A | I | I | R | C | C | I | C |

## 4.3. eTMO Project Management Tools (RAID Log and CI Backlog)

### 4.3.1. RAID Log

Project management involves multiple administrative facets, whose implementation and monitoring can be time consuming. Therefore, it is crucial to identify opportunities that help increase efficiency and the quality of the project. A **RAID Log** is an effective project management tool that is aimed at centralizing and simplifying the collection, monitoring, and tracking of the project information. RAID is an acronym that stands for **R**isks, **A**ctions, **I**ssues, and **D**ecisions.

Table 10: RAID

<table>
<tr><th>RAID Category</th><th>Definition + Objective</th></tr>
<tr><td>Risk (R)</td><td>A situation that may potentially impact program operations (i.e. larger than individual automations) and requires mitigation. Risks are graded by their potential severity and likelihood of occurring (refer Risk Level Matrix table).<ul><li>A Risk is a potential issue. If a risk becomes a reality it should be recorded as an Issue.</li><li>Risk Response: Response to a Risk can be based on the following options</li></ul>
<table>
<tr><th>#</th><th>Risk Response</th><th>Description</th></tr>
<tr><td>1</td><td>Accept</td><td>When the project team determines that the risk will occur and is ready to experience said risk</td></tr>
<tr><td>2</td><td>Avoid</td><td>When the project team determines that the risk needs to be evaded</td></tr>
<tr><td>3</td><td>Contingency</td><td>When the project team needs to develop a response plan (i.e. IF this occurs - THEN the project team will do this)</td></tr>
<tr><td>4</td><td>Mitigate</td><td>When the project team determines that an action plan needs to be developed to lessen the probability of the risk occurring</td></tr>
<tr><td>5</td><td>Transfer</td><td>When the project team shifts the risk from one party to another</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>RAID Category</th><th>Definition + Objective</th></tr>
<tr><td></td><td>
<ul>
<li>Each risk has an associated probability of occurrence along with an impact on the project
<ul>
<li>Risk Probability is the determination of the likelihood of a risk occurring</li>
</ul>
</li>
</ul>
<table>
<tr><th>#</th><th>Probability</th><th>Description</th></tr>
<tr><td>1</td><td>Rare</td><td>May only occur in exceptional circumstances; 1% - 20% chance of occurring</td></tr>
<tr><td>2</td><td>Unlikely</td><td>Could occur at some time; 21% - 40% chance of occurring</td></tr>
<tr><td>3</td><td>Moderate</td><td>Might occur at some time; 41%– 60% chance of occurring</td></tr>
<tr><td>4</td><td>Likely</td><td>Will probably occur in most circumstances; 61-80% chance of occurring</td></tr>
<tr><td>5</td><td>Almost Certain</td><td>Can be expected to occur in most circumstances; more than 80% chance of occurring</td></tr>
</table>
<ul>
<li>Risk Impact is the evaluation of the impact of a risk if it were to occur</li>
</ul>
<table>
<tr><th>#</th><th>Impact</th><th>Description</th></tr>
<tr><td>1</td><td>Minor/Negligible</td><td>Noticeable disruption to the achievement of results</td></tr>
<tr><td>2</td><td>Moderate</td><td>Material deterioration in the achievement of results</td></tr>
<tr><td>3</td><td>Significant</td><td>Significant deterioration in achievement of results</td></tr>
<tr><td>4</td><td>Severe</td><td>Fundamental threat to the desired project results</td></tr>
</table>
<ul>
<li>Risk Level Matrix: A visual representation of the results from Risk Probability and Impact Assessments</li>
</ul>
<table>
<tr><th>Risk Matrix</th><th colspan="5">Impact</th></tr>
<tr><th>Probability</th><th>Negligible</th><th>Minor</th><th>Moderate</th><th>Significant</th><th>Severe</th></tr>
<tr><td>Almost Certain (81% to 100%)</td><td>Low Risk</td><td>Moderate Risk</td><td>High Risk</td><td>Extreme Risk</td><td>Extreme Risk</td></tr>
<tr><td>Likely (61% to 80%)</td><td>Minimum Risk</td><td>Low Risk</td><td>Moderate Risk</td><td>High Risk</td><td>Extreme Risk</td></tr>
<tr><td>Moderate (41% to 60%)</td><td>Minimum Risk</td><td>Low Risk</td><td>Moderate Risk</td><td>High Risk</td><td>High Risk</td></tr>
<tr><td>Unlikely (21% to 40%)</td><td>Minimum Risk</td><td>Low Risk</td><td>Low Risk</td><td>Moderate Risk</td><td>High Risk</td></tr>
<tr><td>Rare (1% to 20%)</td><td>Minimum Risk</td><td>Minimum Risk</td><td>Low Risk</td><td>Moderate Risk</td><td>High Risk</td></tr>
</table>
</td></tr>
<tr><td>Action (A)</td><td>Actions may be the most frequently used RAID entry, and are useful to track</td></tr>
</table>

<table>
<tr><th>RAID Category</th><th>Definition + Objective</th></tr>
<tr><td></td><td>completion against "need-by" deadlines. Actions that are urgently needed are ranked with a higher priority than those that are less urgent or less critical to on-going CoE program operations<ul><li>These are typically actions that arise during project meetings and don't necessarily include all of the actions already on the project plan</li><li>Each action should have an owner, a due date, and eventually a date on which the action was completed</li><li>These actions/action items will be reviewed with associated stakeholders, to update completed actions and review progress against those not yet completed</li></ul></td></tr>
<tr><td>Issue (I)</td><td>A situation that will or is impacting program operations (i.e. larger than individual automations) and requires resolution. Issues are graded by their severity and urgency (refer Issue Level Matrix table below). Some issues may require additional investigation to determine cause or reach a solution. In these cases, a formal root cause analysis may be completed.<ul><li>If an issue results in a decision, the issue should be closed with the resolution documented. At the same time, a decision should be logged (with supporting documents and approval)</li><li>Priority/Severity for issues are assigned based on Urgency Level and Issue Impact<ul><li>Urgency Level: Indicates level denoting how quickly the issue needs to be resolved</li></ul></li></ul>
<table>
<tr><th>#</th><th>Urgency Level</th><th>Description</th></tr>
<tr><td>1</td><td>Critical</td><td>Issue that needs to be addressed immediately as the project cannot move forward until the issue is resolved</td></tr>
<tr><td>2</td><td>High</td><td>Issue that needs to be addressed but does not have an immediate impact on the project</td></tr>
<tr><td>3</td><td>Medium</td><td>Issue that should be addressed but does not have an immediate impact on the project</td></tr>
</table>
</td></tr>
</table>

| RAID Category | Definition + Objective |
|---|---|

- **Issue Impact:** Based off the risk matrix, this column describes the impact severity of the issue

| # | Issue Impact | Description |
|---|---|---|
| 1 | Critical | Issue has a significant impact on the scope, timeline, and/or budget |
| 2 | High | Issue has a moderately high impact on the scope, timeline and/or budget |
| 3 | Medium | Issue has an impact on the scope, timeline and/or budget |

- Assigning a priority rating to the issue based on the following options:

| # | Priority Level | Description |
|---|---|---|
| 1 | Critical Priority | The issue will have significant impact on project/program success; a detailed mitigation and remediation strategy is required, including completion of an issue mitigation document. Requires immediate escalation and resources to resolve |
| 2 | High Priority | The issue may require additional resources and/or escalation for resolution as it may affect the schedule or quality of the project. This issue should be closely monitored, and owner should have issue mitigation document prepared if issue escalates to red |
| 3 | Medium Priority | The issue is manageable and/or most likely would not affect the schedule and quality parameters of the project |

- **Issue Level Matrix:** A visual representation of the results from Urgency level and Issue Impact Assessments

| Issues Matrix | Impact | | |
|---|---|---|---|
| **Urgency** | **Medium** | **High** | **Critical** |
| **Critical** | High Priority | Critical Priority | Critical Priority |
| **High** | Medium Priority | High Priority | Critical Priority |
| **Medium** | Medium Priority | Medium Priority | High Priority |

- **Escalation of Issues**: An issue can be escalated by initiator of the

| RAID Category | Definition + Objective |
|---|---|
| | issue or supporting team member when:<br>• The priority of the issue increases<br>• The issue cannot be managed through direct discussions or day to day management<br>• It is apparent that the issue is not being, or cannot be managed by the current owner<br>• The issue resolution actions require intervention from more senior staff<br>• The project progress will be affected if it stays on the current priority<br>• The issue is overdue, and it is negatively impacting the project progress |
| **Decision (D)** | A conclusion required by the identified owner, usually a program position or business stakeholder. Decisions should be related to overall CoE operations or strategy, and ranked according to their urgency and impact.<br><br>• **For Logging Decisions:**<br>  • It is strongly recommended to attach supporting documentation and approval information i.e. decision document<br>  • All decisions must be logged to have a central repository. If decisions are made in a meeting, log them individually or attach to meeting minutes<br>  • Decisions must be approved by a client representative especially in the following scenarios:<br>  ▪ The decision will impact the business operations<br>  ▪ There is a change in the contractual obligations<br>  ▪ There is a change in the agreed upon project deliverables or milestones |

**eTMO Responsibilities for managing the RAID Log:**

- Record all Risks, Action Items, Issues and Decisions from the project/release related meetings. For each recorded item include the source, date created, due date, date closed, project/release name, owner, any action/mitigation steps, status and comments
- Actively monitor and track the open items (New, Open, In-progress and monitor) until closed
- Periodically review, groom and update the status for completed items as “Closed”, in order to maintain accurate tracking
- Distribute the weekly and monthly RAID Log Status Report for providing progress updates to TennCare Business
- Upload the Monthly RAID Log Status Report on the TennCare SharePoint at the end of each month and include the link as part of monthly deliverables

### 4.3.2. Continuous Integration (CI) Backlog

A Continuous Integration (CI) backlog is a prioritized and structured list of observations and improvements to current eTMO processes. These oberservations and improvements are converted into actionable deliverables for the eTMO team. The Agile Board within the ServiceNow Development Application is the landing page to access the key areas of standard scrum processes like managing backlog items, sprint planning and sprint tracking.

Agile Board Management: eTMO Continuous Integration Backlog

- Create a personalized eTMO Continuous Improvement backlog with a list of user stories from observations and required area of improvements recorded as an outcome from discussion, status, triage & other meetings between the Module Solution Vendor, TennCare Stakeholders and eTMO team. Each backlog item contains the following fields

| # | Fields | Description |
|---|---|---|
| 1 | Short Description | High level summary of the backlog item |
| 2 | Description | User story along with detailed description of the backlog item |
| 3 | Acceptance Criteria | Action steps required to meet the expectations for the completion of the backlog item |

- Create scrum tasks for each user story based on the acceptance criteria to track the progress of the backlog item towards completion
- Assign a specific name for “Product”, “Theme” and “Epic” fields for each user story to efficiently manage and differentiate the items from other workstreams

| # | Fields | Description |
|---|---|---|
| 1 | Product | Identifies the workstream that the story is associated with. For any item created by eTMO team, “ServiceNow Test Management” will be the displayed name under this field |
| 2 | Theme | Identifies the source of the created item. Typically it’s the meeting name from where the item was captured |
| 3 | Epic | Identifies the focus area for the created item. Below is the list of standard focus areas:<br>• eTMO: TennCare Test Management Standards<br>• eTMO: TennCare Testing Standards<br>• eTMO: Capacity Demand Model<br>• eTMO: Project / Release Tracker<br>• eTMO: TennCare Test Reporting Standards<br>• eTMO: UAT Regression Suite Enhancement<br>• eTMO: Test Data Management Strategy<br>• eTMO: Test Oversight and Governance |

- Evaluate the user story impact, and accordingly MoSCow assignment (a prioritisation technique to understand and manage priorities) will be done by eTMO team to select items for TennCare Business to prioritize for the upcoming sprint

| # | MoSCow Assignment | Priority Assignment | Description |
|---|---|---|---|
| 1 | Must have | Critical | Non-negotiable needs for the project, product, or release |
| 2 | Should have | High | Essential needs for the project,product, or release |
| 3 | Could have | Moderate | Desirable but not urgent for the project,product, or release |
| 4 | Would have | Low | Good to have and not urgent for the project,product, or release |

- Understand the planned sprint activities to estimate and track how much time or effort an initiative will take for completion. In agile board the effort estimation for the scrum user story completion is tracked through “Points” field. Typically, the user story is first assigned an owner and then points are assigned with owners agreement

| # | Points | Description |
|---|---|---|
| 1 | 1 | Estimated completion of the user story will be between 1-2 days |
| 2 | 2 | Estimated completion of the user story will be within less than 1 week |
| 3 | 3 | Estimated completion of the user story will be between 1-2 weeks |
| 4 | 4 | Estimated completion of the user story will take more than 1 sprint |

**eTMO Responsibilities for managing the Continuous Integration (CI) Backlog:**

- Understand the current status of key deliverables at all times
- Create realistic timelines that meet objectives by delivery dates
- Distribute deliverables/user stories to team members
- Review the ongoing sprint items twice a week to track the progress made by the team
- Identify delays and mitigation plans to keep the in sprintuser stories on track
- Prioritize the backlog item(s) to favor the most strategically important user stories for the team to work on —to avoid having an open-ended list
- Periodically groom and update the status for completed user stories as “Closed”, in order to maintain accurate deliverable tracking

- Generate awareness to TennCare Business for any user story that will be moved out of the ongoing sprint into the backlog/future sprints
- Develop and apply appropriate work processes, best-practices, and methodologies that enhance the teams productivity
- Distribute the Continuous Improvement Backlog Status Report weekly and end of each sprint for providing progress updates to TennCare Business on items assigned for the ongoing sprint
- Upload the Monthly Continuous Improvement Backlog Status Report on the TennCare SharePoint at the end of each month and include the link as part of monthly deliverables

# 5. Deliverables and Acceptance Criteria

## 5.1. Deliverables

Section 4.2 lists responsibilities for a set of Deliverables required for a solution project. These deliverables include the Solution Test Plan defined in section 6.2. They also include the following Test Reports:

### 5.1.1. Test Reports

To complete each stage of testing, a Test Report is delivered. Acceptance of these formal reports is required to pass gates in the Solution Implementation Lifecycle. The RACI table (section 4.2) indicates who is responsible for creating each report (in most cases the solution vendor), and who is accountable for approving the report to certify that the testing is complete and accepted (TennCare Business or IS). These deliverables are listed in the RACI table:

- Unit Test Report
- Functional System Integration Test Report
- System Integration Test Regression Report
- Functional User Acceptance Test Report
- User Acceptance Test Regression Report
- Operational Readiness Test Report
- Beta Test Report

Each Test Report deliverable declares that the solution vendor has completed a stage of testing of a module. It indicates:

- what module version was tested, with what data on what environment
- the percentage of test cases passed, failed and not completed
- number of defects outstanding, by severity
- mitigation plan for each outstanding defect

If a module or solution has been tested previously (such as commercial off-the-shelf software), those previous results may be used in the applicable Test Report. Conduct additional tests as needed to show that the module or solution fulfills TennCare's requirements, and that any TennCare customizations, configurations and integrations are functional.

Test reporting governance and oversight will occur to prevent delays in Solution Vendor responsiveness to reported issues.  Defect SLA guidelines are as follows:

- Sev-1 Critical Defects – Triaged, root-caused and assigned in less than 12 hours
- Sev-2 High – Triaged, root-caused and assigned within 1 day (24 hours)
- Sev-3 Medium – Triaged, root-caused and assigned within 2 days (48 hours)
- Sev-4 Low – Triaged, root-caused and assigned within 5 days (1 week)

Each Test Report shall include the elements in Table 10, which refers to some data elements in section 10.4.

Table 10: Test Report Deliverable Content

| Topic | Report Content |
|---|---|
| Report Submission | Declaration that the stage of testing is complete and the solution module is ready for promotion to the next stage.<br><br>Organization, role/position, person name, and signature of the responsible party (e.g. vendor's testing manager).<br><br>Date of submission. |
| Approval / Certification | Checkbox indicating that the stage of testing has been verified to be complete, and the solution module is certified as ready for promotion to the next stage.<br><br>Organization, role/position, person name and signature of the accountable party.<br><br>Date of approval. |
| What was Tested | • Solution Name<br>• Module Name and Version<br>• Dataset(s) Name and Version<br>• Environment(s) Names |
| Test Stage | Unit, SIT(Functional and Regression), ),UAT (Functional and Regression)ORT or Beta. |
| Test Results | Percentage of test cases/scripts/scenarios, by status:<br><br>• Passed (Completed successfully upon latest test)<br>• Failed (Tested with defect outstanding)<br>• Not completed (never tested) |
| Defects | Number of defects outstanding (according to the Defect Status)<br><br>• by Defect Severity |

| Topic | Report Content |
|---|---|
| Defects | Mitigation plan for each outstanding defect |

## 5.2. Entrance Criteria

The following entrance criteria confirms readiness to start a testing stage. The minimum recommended entry criteria are as follows:

- All planned test cases have been written, prioritized, traced to the requirements in the Requirements Traceability Matrix, and approved by TennCare.
- The Defect Management Process has been agreed upon and the responsibilities of each stakeholder is understood. This includes understanding of the defect management tool usage, defect workflow, defect statuses, severities, priorities, and defect resolution expectations.
- Prior test stage Acceptance Criteria approved by TennCare.
- The required Test Environment for the test stage is available and stable. All smoke tests to validate environment availability have been executed and passed.
- The code deployment schedules across all modules for the Test Environment have been established and published, along with the process for deployment activity notification and release notes.

In order to facilitate readiness, and minimize risk at day 1 of a test stage. An entrance criteria review should be conducted several days prior to determine trajectory of completion and mitigation steps can be taken if there is risk of not meeting any criteria.

## 5.3. Acceptance Criteria

The following test-completion criteria determine whether a solution will pass a gate approval to the next phase. The gate approvals also depend upon completion of the Test Report deliverables (section 5.1.1). The RACI chart (section 4.2) specifies who the accountable approver is at each stage. See the definitions of defect severity levels in Table 24.

Recommend insertion of exit criteria matrix.

| Testing Stage | Execution Percentage | Threshold Rate | Critical/Blocking |
|---|---|---|---|
| Unit Test | 100% | **Per RFP, Contract, or approved Solution Test Plan* | Zero |

| Testing Stage | Execution Percentage | Threshold Rate | Critical/Blocking |
|---|---|---|---|
| Functional SIT | 100% | **Per RFP, Contract, or approved Solution Test Plan* | Zero |
| SIT Regression | 100% | **Per RFP, Contract, or approved Solution Test Plan* | Zero |
| Functional UAT | 100% | **Per RFP, Contract, or approved Solution Test Plan* | Zero |
| UAT Regression | 100% | **Per RFP, Contract, or approved Solution Test Plan* | Zero |
| ORT | 100% | **Per RFP, Contract, or approved Solution Test Plan* | Zero |
| Beta | 100% | **Per RFP, Contract, or approved Solution Test Plan* | Zero |

It is not recommended to proceed to the next testing phase if the above is not achieved. Exception governance rules should apply(Exception process is to record a project issue that requires either mitigation or resolution and TennCare acceptance should progress). Each test phase execution reporting should have a clear status of progress towards exit criteria.

### 5.3.1. Development Phase: Unit Test Acceptance Criteria

The **Unit Test Report** declares that the solution vendor has completed the Unit Testing stage for a module.

TennCare will certify that the solution module is ready for promotion to the SIT stage if:

- The Unit test cases are approved (as an appropriate and complete set)
- 100% of approved Unit test cases are tested (completed)
- Threshold % of Unit test cases are successful (passed) (The threshold percentage is to be defined in the RFP, Contract, or approved Solution Test Plan.)
- All defects have been logged in the defect management tool with a severity level
- Defects are fixed where feasible
- All unresolved defects have a mitigation plan approved by TennCare

### 5.3.2. Testing Phase: SIT Acceptance Criteria

#### 5.3.2.1. Functional SIT Acceptance Criteria

The Functional **System Integration Test Report** declares that the solution vendor has completed the Functional SIT Stage of testing of a module.

TennCare will certify that the solution module is ready for promotion to the UAT stage if:

- The Functional SIT test cases are approved (as an appropriate and complete set)
- Test execution results are documented for all tested test cases
- Rework, retest and regression tests are repeated until quality criteria satisfied
- Threshold % of Functional SIT test cases are successful (passed) (The threshold percentage is to be defined in the RFP, Contract, or approved Solution Test Plan.)
- All critical and blocking defects (from Functional SIT or previous stages) have been resolved
- All high, medium and low severity defects are allowed subject to the acceptance by defect management processAll unresolved defects have a mitigation plan approved by TennCare
- Acceptance of test summary report for phase by TennCare
- Functional SIT sign-off is completed

#### 5.3.2.2. SIT Regression Acceptance Criteria

The execution of Functional SIT test cases should be complete before promoting to the SIT Regression Stage. The **System Integration Test Regression Report** declares that the solution vendor has completed the SIT Regression Stage of testing of a module.

TennCare will certify that the solution module is ready for promotion to the UAT stage if:

- The SIT Rregression test cases are approved (as an appropriate and complete set)
- 100% of approved SIT Regression test cases are tested (completed)
- Test execution results are documented for all tested test cases
- Threshold % of SIT Regression test cases are successful (passed) (The threshold percentage is to be defined in the RFP, Contract, or approved Solution Test Plan.)
- All critical and blocking defects (from SIT Stage, SIT Regression Stage or previous stages) have been resolved
- All high, medium and low severity defects are allowed subject to the acceptance by defect management process
- All unresolved defects have a mitigation plan approved by TennCare
- Acceptance of test summary report for phase
- SIT Regression sign-off is completed

### 5.3.3. Testing Phase: UAT Acceptance Criteria

#### 5.3.3.1. Functional UAT Acceptance Criteria

The Functional **User Acceptance Test Report** declares that the solution vendor has completed the Functional UAT Stage of testing of a module.

TennCare will certify that the solution module is ready for promotion to the ORT stage if:

- The Functional UAT test cases are approved (as an appropriate and complete set)
- 100% of approved Functional UAT test cases are tested (completed)
- Test execution results are documented for all tested test cases
- Rework, retest and regression tests are repeated until quality criteria satisfied
- Threshold % of Functional UAT test cases are successful (passed) (The threshold percentage is to be defined in the RFP, Contract, or approved Solution Test Plan.)
- All critical and blocking defects (from Functional UAT or previous stages) have been resolved
- All high, medium and low severity defects are allowed subject to the acceptance by defect management process
- All unresolved defects have a mitigation plan approved by TennCare
- The TennCare UAT testing/ Business team accepts the solution module
- Acceptance of test summary report for phase
- Functional UAT sign-off is completed

#### 5.3.3.2. UAT Regression Acceptance Criteria

The execution of Functional UAT test cases should be complete before promoting to the UAT Regression Stage. The **User Acceptance Test Regression Report** declares that the solution vendor has completed the UAT Regression Stage of testing of a module.

TennCare will certify that the solution module is ready for promotion to the ORT stage if:

- The UAT Regression test cases are approved (as an appropriate and complete set)
- 100% of approved UAT Regression test cases are tested (completed)
- Test execution results are documented for all tested test cases
- Threshold % of UAT Regression test cases are successful (passed) (The threshold percentage is to be defined in the RFP, Contract, or approved Solution Test Plan.)
- All critical and blocking defects (from UAT Stage, UAT Regression Stage or previous stages) have been resolved
- All high, medium and low severity defects are allowed subject to the acceptance by defect management process
- All unresolved defects have a mitigation plan approved by TennCare
- The TennCare UAT testing/ Business team accepts the solution module
- Acceptance of test summary report for phase

- UAT Regression sign-off is completed

### 5.3.4. Implementation Phase: ORT Acceptance Criteria

The **Operational Readiness Test Report** declares that the solution vendor has completed the ORT stage of testing of a module.

TennCare will certify that the solution module is ready for promotion to the Beta stage if:

- The ORT test cases are approved (as an appropriate and complete set)
- 100% of approved ORT test cases are tested (completed)
- Threshold % of ORT test cases are successful (passed) (The threshold percentage is to be defined in the RFP, Contract, or approved Solution Test Plan.)
- All critical and blocking defects (from ORT or previous stages) have been resolved.
- All unresolved defects have a mitigation plan approved by TennCare
- The TennCare IS testing team accepts the solution module

### 5.3.5. Implementation Phase: Beta Test Acceptance Criteria

The **Beta Test Report** declares that the solution vendor has completed the Beta Test or Pilot stage of releasing a module.

TennCare will certify that the solution module is ready for Go-Live if:

- Beta or Pilot testing has been completed as planned
- All critical and blocking defects have been resolved
- All unresolved defects have a mitigation plan approved by TennCare
- The TennCare Business and IS testing team accepts the solution module

# 6 Test Planning

## 6.1. Master Test Plan

The Master Test Plan will provide coordination of testing multiple interdependent releases of solutions across a program, such as the Medicaid Modularization Program or enterprise-wide. It is not part of any one solution's lifecycle and is distinct from the Solution Test Plan.

TennCare IS will own, and be accountable for, this plan. The IS Vendor will provide reporting and coordination for the Master Test Plan, which should be drafted before vendor selection, and should later have input from both the IS Vendor and module vendors.

The Master Test Plan should include items listed in the following table:

Table 11: Master Test Plan Content

| Master Test Plan Content | Guidance |
|---|---|
| Master Testing Strategy | The Master Testing Strategy section will contain a coordinated plan to support the execution of the Solution Test Plans for each solution. It will be based on a program-level or enterprise-level strategy that sets the sequence for releasing solutions and modules. The Master Testing Strategy should be coordinated with project plans through the applicable project management office. |
| Master Testing Schedule | The Master Testing Schedule section will be updated quarterly, at a minimum, to manage the availability of TennCare IS resources (staff, environments and tools). These resources need to be scheduled for use in testing all solution projects to avoid delays or collisions. |

| Master Test Plan Content | Guidance |
|---|---|
| Common Methods and Resources | The Common Methods and Resources section will describe testing methods, tools, and environments that are provided by TennCare for reuse by multiple vendors.<br><br>Additionally, preferred methods and documents will be listed in this section, such as a preferred way of managing defects, or definitions of preferred reports and metrics.<br><br>Vendors will have access to the resources listed in this section, and should seek approval before deviating from the TennCare preferred methodology. |
| Common Metrics | This section will specify testing metrics that must or should be provided by all solution projects covered by the Master Test Plan.<br><br>For each metric, define the frequency or timing, calculation, inclusions, baseline and target levels. See section 10.65 for suggested metrics. |
| Privacy and Security | This section will articulate privacy and security requirements and considerations that apply to testing of multiple solutions. In particular, it will specify procedures for handling Personal Health Information and other sensitive data. This may include the de-identification procedures for test data, further explained in section 7.2. |

## 6.2. Solution Test Plan

A Solution Test Plan is needed for each solution. Each Solution Vendor is responsible for defining a Solution Test Plan which must be aligned with the Master Test Plan, and clearly denote Solution Vendor approach to internal Unit-Testing and System Integration Testing (SIT)-approachs, tools, execution progresss reporting, defect management, and entry/exit/release governance criteria.

This Solution Test Plan must also provide a test-data strategy approach to ensure that de-identified real-world data conditions have been exercised through the solution during System Integration Testing.

If a solution has multiple modules, this deliverable should cover each of the modules in their solution, and the integrations between the modules.

A preliminary version of the Solution Test Plan is submitted by bidding vendors as part of their response to an RFP. This test plan is updated and baselined after the selected vendor is on-board. The plan must be approved by TennCare to ensure the vendor's testing plans align with TennCare's expectations and strategies.

The Solution Test Plan should contain detailed descriptions of testing, including specific references to the module test cases and scripts to be used, as well as the approach, tools, resources, and activities for managing the schedule, scope, test environments, and test data throughout the project. The table below describes the content:

Table 12: Solution Test Plan Content

| Solution Test Plan Content | Guidance |
|---|---|
| Testing schedule | The expected dates when each stage of testing will be prepared and executed, in a table or Gantt chart.<br><br>Needed for planning availability of environments, UAT testers, and TennCare IS staff. |
| Testing personnel | Describe the testing team: number of testers and testing managers, their relevant qualifications, and their security compliance. The vendor must produce adequate coverage plans to mitigate delays associated with resource turnover or termination. |
| Training | Describe the topics and duration of any training planned for testing personnel. The vendor should provide adequate training including documentation, access to user stories, product demonstrations, requirements and architecture documents/diagrams to all testing resources prior to the start of all testing activites. |
| Testing meetings | Describe the purpose, participants, length, and frequency of meetings required, such as regular reviews of test results and defects. |
| TennCare involvement | For UAT testers and TennCare IS staff: Describe the person-days required and any special knowledge or access they require, to complete UAT, ORT and Beta testing. |
| Testing Methods | Describe the testing methodology and how it will be implemented. |

| Solution Test Plan Content | Guidance |
|---|---|
| Collaboration with other vendors | The Solution Test Plan will show how the solution vendor works with the IS Vendor to coordinate testing in the integrated environments. The vendor should collaborate with all parties to establish working procedures and agree on testing timelines to mitigate upstream and downstream effects that can cause delays in testing.This includes environment downtime, maintence, deployment schedules that impact all interfacing applications required for testing.<br><br>Indicate the activities for which the module vendor will rely on the ISL or other vendors, and how much time is expected from them.<br><br>Indicate the activities that the module vendor will do to assist the ISL and other vendors and how much time is planned for those activities. |
| Test cases, scripts and scenarios | A summary of, and references to detail of, the test cases  scripts / scenarios to be used at each test stage. |
| Number of Scripts Needed | For each functional change (CR and/or Incident), the number of scripts needed for testing must be included for resource planning purposes. A sample reporting matrix that includes Number of Scripts Needed can be found in the **eTMO Reporting Metrics** document in the section 'Expedited Release/Testing Effort' |
| Test data | Summary and detailed requirements for test datasets, as specified in Table 13. This may include the de-identification procedures for test data, further explained in section 7.2.<br><br>Mocked up data may only be used for Unit-Testing.  System Integration testing must include de-identified test-data for TennCare review and acceptance into UAT. |

| Solution Test Plan Content | Guidance |
|---|---|
| Test environments | A list of all the TennCare environments that the vendor wishes to use should be populated. Include environments required for testing the new solution, and for testing decommissioning and migration from a legacy system.<br><br>A separate list of all non-TennCare environments that the vendor wishes to use or create should be written, with detailed descriptions of each environment for approval by TennCare. |
| Tools | A list of all the TennCare tools that the vendor wishes to use should be populated. A separate list of all non-TennCare tools that the vendor wishes to use should be created, with detailed descriptions of each tools for approval by TennCare. |
| Daily & Weekly Test Execution tracking and results | A summary of the process and methods for tracking completed tests and their results, with test execution and defect management reports produced on a daily and weekly basis aligned to TennCare KPIs, metrics and definitions (defect priority, severity)<br><br>*Note: Execution results need to be maintained and available for review by TennCare, TAS and other parties to support downstream inspection and certification activities.* |
| Defect management | Define how defects will be managed over their life cycle, from identification, categorization, logging, review, reporting, resolution, and approved closure. |
| Re-testing and regression testing | Indicate how the vendor will support re-testing and regression testing after fixes and other technical changes.The full regression suite and smoke test cases must be approved by TennCare. The vendor must provide detailed descriptions both technical and business related for all defect deployements and provide sufficient communication surrounding potential upstream and downstream effects. The regression test suite should be updated after each defect deployment to include any additional regession tests.Changes may occur throughout the solution's lifecycle, including in the Operations and Maintenance phase. Include regression testing for changes made by TennCare or other vendors to solutions and systems that are integrated with this module. |

| Solution Test Plan Content | Guidance |
|---|---|
| Test reporting and metrics | Specification of the reports and metrics to be produced at specified times throughout the testing process. For each metric, define the calculation, inclusions, baseline and target levels. See section 10.6 for suggested metrics. |

The review and approval of each Solution Test Plan shall be tracked by TennCare, using the following data elements:

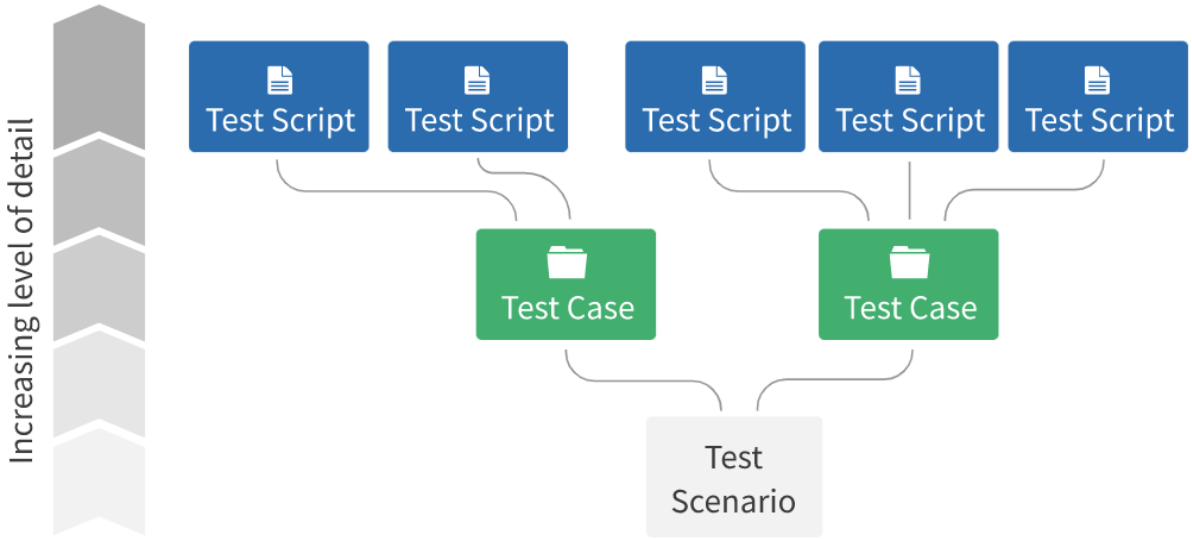
- Solution Test Plan unique identifier
- Solution Test Plan version (and description if appropriate)
- Status, with values to include: In review, In revision, Approved
- Revisions required (description)
- Date approved
- Approved by (Person name and Organization role)

The approved version of the Solution Test Plan shall be captured and linked to the approval record.

# 7. Test Preparation

The test preparation activities are described in the RACI chart (section 4.2) and the Solution Test Plan (section 6.2). This section describes the requirements for a solution project to prepare its test cases, scripts and scenarios; its test data; and its tools to manage testing.

## 7.1. Test Cases, Scripts and Scenarios




| Terms | Description |
|---|---|
| Test Script | A test script is a set of instructions that is performed on a system under test to verify that the system performs as expected. They can be written in either a human language (used for manual testing) or a scripting / programming language (used for automated testing). A test script is a part of a test case and a test case can have either a single test script or multiple test scripts |
| Test Case | A test case is a documented set of preconditions (prerequisites), procedures (inputs / actions) and postconditions (expected results) which a tester uses to determine whether a system under test satisfies requirements or works correctly. A test case can have one or multiple test scripts and a collection of test cases is called a test suite. |

| Terms | Description |
|---|---|
| Test Scenario | The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases. A test scenario has one relation to many relationships with the test cases. |

Test cases, scripts, and scenarios are collectively known as **test content**.

Any kind of test content specifies what requirement shall be tested and how. These instructions may be at various levels of detail or precision. The test content may be written for a user to manually test, or for technical testing by a developer or tester working manually, or may be encoded for an automated testing tool. Test content may be specified for the module, point-to-point or end-to-end scope of testing (section 3.4).

For any TennCare solution, the test content must:

- Be comprehensive, covering all types of test to be done, and all parts of the solution and module
- Have direct traceability to both functional and non-functional requirements, using a Requirements Traceability Matrix.
- Contain enough specific and unique information for testers, as specified below
- Contain updated user guides, training materials, and communications.
- Be clear, concise, and written so that they can be easily followed, meaning the tester does not need to have extensive subject matter experience to understand the test script and expected resultsIn cases where test cases require additional knowledge of the system under test, the vendor should provide adequate documentation, training and demonstrations for both third party, and TennCare resources
- Be approved by the accountable TennCare representative
- Latest versions of documentation, such as user guides, must be provided by the vendors so that TennCare has accurate information for review and approval.
- Be stored and tracked in a test repository, with versions kept when the test content is revised
- Be linked to the results of testing the case, script or scenario

Once test cases, scripts, and scenarios have been drafted, they should be shared with the State for approval, and then sent to storage in their test repository for tracking and reporting.

Section 10.4 specifies the data elements that must be collected about test cases, scripts and scenarios. This required data enables tracing what was tested, to aid in defect management, and also is needed for planning and managing the testing process. Solution vendors may add more data elements to categorize and track their test content.

The review and approval of test cases, scripts, and scenarios shall be tracked by TennCare, using the following data elements:

- Test content's unique identifier
- Test content's name (and description if appropriate)
- Status, with values to include In review, In revision, Approved
- Revisions required (description)
- Date approved
- Approved by (Person name and Organization role)

The approved version of the test content shall be captured and linked to the approval record.

## 7.2. Test Data

The finalized Solution Test Plan (section 6.2) shall specify how test data will be managed for its solution project. TennCare will assist the solution vendor in planning for and obtaining needed test data. TennCare Privacy and Security approval is required prior to sending any sensitive data or production data to a vendor. Table 13 lists the specifications required and additional considerations that provide guidance to solution vendors on TennCare's expectations for managing test data and the testing process:

Table 13: Test Data Specifications

| Test Data Specifications | Guidance about Test Data |
|---|---|
| Test data approval | TennCare approval of a solution vendor's Test Plan, with these data specifications, is required prior to extracting any production data or sensitive data for testing. |
| Test data elements required | Names of the databases, tables, and fields to be extracted. Specify selection criteria and any referential integrity required between tables. |
| Test data extraction | The vendor shall work with TennCare or associated database administrators (DBAs) to extract, transform, and load data into the test environments as appropriate. |
| Test data transformation | Transforming test data (including cleansing and de-identifying it) may be the responsibility of the solution vendor, an IS Vendor or TennCare, depending on how data will be shared. |
| Test data cleansing | Consider whether the tests need cleansed data. Data with realistic quality issues may be needed to test how the solution handles errors and inconsistencies. |

| Test Data Specifications | Guidance about Test Data |
|---|---|
| Test data de-identification | Only PROD, PREPROD (and STAGE environments that are in the production domain with Tenncare Executive Leadership approval) are allowed to use real data without de-identification in tests. Vendors requesting any deviation from this ***must obtain written approval from TennCare Executive Leadership.***<br><br>De-identify PII, PHI, FTI, and other sensitive data where necessary.<br><br>Specify which fields need to be de-identified.<br><br>Provide a detailed methodology and documentation of the rules for de-identifying in the Master or Solution Test Plan. Consider algorithms that keep the data realistic, to maximize how much testing can be done with de-identified data. Ensure that the de-identification method is suitable for integration testing across modules and vendors, where applicable. |
| Test data access | Designate which personnel will be authorized to see sensitive data when managing data, conducting tests, or viewing test results. |
| Test data security | For tests that must be conducted with sensitive data, ensure full security compliance measures are applied to the test databases, environments, and personnel. |
| Test data loading | See the RACI chart for responsibilities for managing shared test environments, including loading test data to those environments. |
| Test dataset tracking | Use a tool to track the test datasets loaded to any environments. Track dataset versions, refreshes, updates, and test datasets created by other tests. |
| Test data identification in results | Test results shall track which test dataset version was used to run the test to help with defect management. Testing oversight may use this information to monitor and plan data “fixes” that enable a test to succeed. |

| Test Data Specifications | Guidance about Test Data |
|---|---|
| Test data refreshes | Specify the frequency of refresh required for each test dataset. As testing modifies the test data, with both successful and failed tests, the data gradually becomes less realistic. Periodically, the test data can be refreshed with a "gold copy" snapshot of production data (de-identified as appropriate).<br><br>Data refreshes must be coordinated, scheduled, and communicated to all users of the test environment to avoid interrupting an in-flight test or yielding false test results.<br><br>To support O&M testing and Decommissioning activities it is recommended to continue refreshing the test data after Go-Live. |
| Test data backups | Specify the frequency of backups required for each test dataset. |
| Test data retention and deletion | Specify whether, how and for how long each dataset will be retained, archived, or securely deleted.<br><br>A dataset might be retained for use in a subsequent testing stage or for later testing of fixes and regression testing.<br><br>A dataset might be archived, such as by removing it from a test environment but storing it for reference in interpreting test results.<br><br>A dataset might be deleted at the end of a stage or after the solution project is completed. |

The review and approval of each test dataset shall be tracked by TennCare, using the following data elements:

- Test data unique identifier
- Test data version (and description if appropriate)
- Status, with values to include In review, In revision, Approved
- Revisions required (description)
- Date dataset and uses approved
- Approver information (Person name and Organization role)
    - The approved version of the test dataset shall be captured and linked to the approval record.

## 7.3. Use and Choice of Tools

Solution vendors are encouraged to use hardware and software tools, repositories, and automation to conduct and manage testing and defects. These tools are expected to yield mature and consistent management of the many people, processes, and things involved in testing a complex solution.

Solution vendors must specify all the testing-related tools they plan to use in their Solution Test Plan (section 6.2), which is to be approved as specified in the RACI chart.

The TennCare Technology Standard may specify Mandatory tools that vendors are required to use, and preferred tools that vendors are recommended to use, for various capabilities related to testing. The version of the TennCare Technology Standard at the time of procurement (appended to the RFP) shall apply.

From time to time, TennCare may have one or more shared testing-related tools, for use across all solutions in a program or the entire enterprise.

To enable integration and coordination across modules, vendors may be required (through an RFP or contract) to do any of the following:

- Use a shared tool or repository for specified test-management capabilities
- Interoperate with a specified tool
- Regularly provide test-management data in a specified format

For a given capability, if no relevant tool is specified by the TennCare Technology Standard, RFP or contract, the solution vendor shall acquire their choice of tool that provides the required functionality and is widely used in industry. For complex functions, it is not recommended to use spreadsheets, custom-built tools, or manual processes.

Vendors are encouraged to minimize the variety of tools they use and maximize the interoperability of their tools, by choosing tools that offer multiple capabilities, plug-in extensions, or other integrations. Where multiple tools offering overlapping capabilities are selected, the applicable Test Plan must specify which tool will be used for what.

Please see appendix section 10.6 for additional information on capabilities.

# 8. Testing Oversight

## 8.1. Test Results

As test cases are completed, vendors must track test results. The records of test results support management and oversight of the testing process, and satisfying any necessary audits.

The records and artifacts of test results should include all applicable screen-shots, reports, mouse clicks, keyboard entries, or query results, that show the completion of each test step. All error messages should be captured and included in test collateral. Error messages may aid in locating defects and assist in defect triage activities.Test collateral should be organized and stored in an approved repository/tool that is accessible by TennCare resources for visibility and certification of test results for all phases of testing. Test collateral should be documented and organized in a standard format agreed upon by TennCare.

The vendor stakeholder responsible for certification of test results for each phase of testing activities should be consistent throughout the entirety of the certification process. The vendor stakeholder should certify test results only if all test collateral has been captured and uploaded to the test record/artifact repository and approved by TennCare. In the event the certification vendor stakeholder is changed the vendor should gain TennCare approval before any further test results are certified.

The vendor should communicate and provide an accurate timeline for the certification of test results based on the estimated effort to gather all necessary evidence and collateral needed for certification. Results certification that require a large amount of effort should be prioritized early in the certification life cycle so an efficient certification process can be established and clear expectations, transparency are shared among all teams.

Vendor test-result validation approaches and tools should be covered in the Solution Vendor Test Plan, refer to section 6.2.  The Solution Vendor should provide their recommended test-validation strategy, approach and tools to TennCare during test-planning and to obtain acceptance/approval on the recommended validation approaches, including certification-test validation to mitigate or alleviate execution validation delays during SIT, UAT, ORT and phases subsequent to Solution Vendor Unit-Testing.

To enable this reporting, test results must be linked to the test case, script or scenario that was tested, which in turn must be linked to the requirement being tested.

Section 10.4 specifies the data elements that must be collected about Test Results of all types at all stages.

## 8.2. Monitoring and Metrics

In addition to the Test Report deliverables (section 5.1.1), many reports and metrics may be produced to oversee the testing process.

Metrics for TennCare's oversight should focus on the quality of the solution, such as by monitoring the completeness of defect resolution.

Vendor will need to provide visibility and reporting to monitor the productivity of testing to achieve timely deliverables. Test completion reports are required as specified in the RACI chart (section 4.2).

Suggested metrics are listed in section 10.6. Requirements for metrics may be set in the enterprise or program Master Test Plan or in the Solution Test Plan; see section 6.

In order to support monitoring, reporting, and metric calculations, vendors must provide a minimum set of testing and defect data elements outlined in section 10.4. This test management data must be available in disaggregated form to validate the aggregate reporting and metrics, for transparent test reporting.

Test executon and defect metrics should be disseminated for the SIT and UAT environments on a daily basis. A summary of the metrics for each environment should be disseminated weekly.

## 8.3. Testing Evaluation Criteria

The project team tracks the following information to drive the assessment of the Design Management status of the project. The document below outlines the current eTMO reports and the underlying metrics used to generate them:

eTMO Reporting Metrics.docx

The quality status information above is continuously evaluated in alignment with the project's status cadence. The applicable quality status evaluation is reviewed with the overall project team and approved by TennCare project leadership based on this assessment .The Quality Status Thresholds are listed in the following table:

*Table 14: Quality Status Thresholds*

| Indicator | Critera |
|---|---|
| **Green** | • Quality status will be assessed as Green if all of the following criteria are met:<br>  • Number of outstanding critical severity defects = 0; all high severity defects have business acceptance; medium and low severity defects are allowed, subject to acceptance by the defect management process and agreement of business stakeholders<br>  • Number of test cases executed vs the number planned is >= 95%<br>  • Number of passing test cases is >= 92% |
| **Yellow** | • Quality status will be assessed as Yellow if any of the following criteria are met:<br>  • Number of outstanding critical severity defects = 0; all high severity defects have business acceptance, and workarounds are approved only for short duration; medium and low severity defects are allowed subject to acceptance by the defect management process and agreement of business stakeholders<br>  • Number of test cases executed vs the number planned is >= 80% and < 95%<br>  • Number of passing test cases is >= 80% and < 92% |
| **Red** | • Quality status will be assessed as Red if any of the following criteria are met:<br>  • Number of outstanding critical severity defects is > 0, and/or number of outstanding high severity defects do not have an identified workaround or business acceptance<br>  • Number of test cases executed vs the number planned is < 80%<br>  • Number of passing test cases is < 80% |

# Defect and Production Incident Management

The testing process includes identifying and reporting defects, also known as bugs. Defects need to be located, resolved, and re-tested.

Production Incidents are potential defects that typically require prompt attention due to undesirable business impacts. Once confirmed to be an issue through the triage process, production incidents follow the same Defect Management process outlined in this section.

The RACI chart, section 4.2, specifies defect management activities and who is responsible for them. Solution vendors testing their own solution may find defects in other interfaces and modules, at which point, they should initiate the defect management and be responsible until that responsibility is transferred to someone else. Defect resolution and mitigation plans are included in the Test Report deliverables, section 5.1.1.

The below document outlines the Defect and Production Incident Management reporting standards currently used to generate eTMO reports.

eTMO Reporting Metrics.docx

Table 15 gives more specifics and guidance about the defect management process.

## 9.1. Defect Management Process

*Table 15: Defect Management Process*

| Step | Requirements and Guidance |
|---|---|
| Log defect | A defect record must be created for each actual test result that does not match the expected result.<br><br>Defects will be managed in a tool that has the capabilities listed in section 7.1.<br><br>The information required about defects is listed in section 10.4. |

| Step | Requirements and Guidance |
|---|---|
| Triage defect | The severity of a defect must be rated. This influences the priority of fixing it. The possible ratings of the severity of a defect are in section 10.5. In order to effectively plan, vendors would need to provide an accurate resolution ETA<br><br>All currently open defects are also reviewed during triage, to ensure that current ETAs are accurate, and that the trajectory for resolution are being met based on severity and priority. |
| Locate Defect Cause | Defects may be discovered by module, point-to-point, or end-to-end tests (section 3.4), and so their source and root cause needs to be located across multiple modules and interfaces.<br><br>The vendor responsible for a test shall lead the effort to locate a defect discovered through that test. The other related vendors (the IS Vendor or others developing or managing the modules and interfaces involved in the test case / script / scenario) shall assist in locating the defect's source.<br><br>The repository of test results (section 7.1) will aid in identifying the source of a defect. |
| Assign responsibility for defect resolution | The vendor responsible for the source of the defect is responsible for fixing the defect. If the responsibility is not clear, or no vendor accepts responsibility for a defect, TennCare IS shall investigate and assign responsibility to the vendor most capable of resolving the defect. |

| Step | Requirements and Guidance |
|---|---|
| Defect Communication | Strategy and Approach for Environment Outages due to fixes:<br>• Automated smoke tests and health checks should be run periodically throughout the day.<br>• Define the core flows that must be executed as part of a regression / health check and gain approval from TennCare.<br>• Create a distribution list to communicate environment outages.<br>• The vendor should send out regular communication surrounding the stability and status of all aspects of the testing environment.<br>• Creation of a dashboard that shows the health and status of critical services is a best practice that should be introduced<br><br>If the vendor disagrees to any reported defects the vendor must provide supporting evidence/proof supporting their position. If TennCare disagrees to the defect communication the defect management/triage processshould be followed to evaluate the legitimacy of the defedt. |
| Deploy Fix | The modified code, customization, or configuration is packaged and deployed to environments appropriate to the testing stage. The Vendor should create a deployment schedule for all defect fixes that is communicated and approved by TennCare. Defect deployments outside the approved deployment schedule should be approved by TennCare prior to the deployment. |
| Communicate Fix | The vendor making the fix communicates the resolution to any other vendors who will need to make related tests. The vendor should clearly communicate all defect tracking numbers and defect details for every defect deployment through demonstrations, daily stand ups and documentation. The vendor should allocate sufficient time before any defect deployment for official approval of the proposed fix. |

| Step | Requirements and Guidance |
|---|---|
| Test Fix | The applicable test case / script / scenario needs to be re-tested, by the vendor originally responsible for that test. The vendor should communicate any potential upstream and downstream effects for each fix to ensure sufficient test coverage for functional and regression tests.<br><br>Defects need to be tested in prior test stages to confirm that they are fixed. For example, UAT defects need to be validated in SIT to confirm that the issue has been fully resolved.<br><br>For production incidents, test cases should be created and reviewed with TennCare Business and TennCare IS. The approved test cases should be added to the project's test repository for re-testing and regression purposes, as described in section 3.6.2. |
| Regression Test | Regression testing must be performed (perhaps by multiple related vendors) to confirm that the defect resolution has not introduced any unintentional consequences. Unit tests, SIT, UAT or ORT tests may need to be repeated in regression testing.<br><br>For the State of Tennessee project, SIT and UAT Regression Test Suites have been approved by TennCare Business. |
| Mitigate Defect | If a defect cannot be immediately resolved, a mitigation may be proposed, such as a work-around, deferral, documentation, training, or remediation. All mitigations must be approved by TennCare. Mitigations need to consider all the modules and parties that may be affected. A mitigation does not resolve (close) the defect. |
| Update and Close Defect | The defect record needs to be updated with the resolution or mitigation. Those overseeing the testing are informed of the defect being resolved. |
| Approve Defect Resolution or Mitigation | Each defect resolution is reviewed to show that the fix is successful. Each mitigation proposal is reviewed for approval as part of the stage's Test Report.<br><br>Sign-off will indicate that all defects have been successfully fixed or acceptably mitigated. The RACI chart indicates the level of defect resolution required to pass each phase gate. |

## 9.2. Tracking Defects

A defect will be logged for each testing incident (i.e., actual test result does not meet expected result). Section 10.4 lists the data elements that must be collected about for defects found during tests of all types at all stages.

Recording defects allows testing teams to prioritize more severe defects while tracing their source and cause, and planning fixes, re-testing, regression testing, and approvals.

Throughout the stages of test execution, oversight is needed. The oversight of testing includes reporting, meetings and other observations to monitor defect management and the quality of the solution in progress.

Tracking of defects allow oversight of the testing process, such as monitoring for:

- Delays in defect resolution
- Defects that stay unresolved through multiple stages of testing
- Defects that need many cycles of fixing and re-testing
- Too many defects being cancelled because of tester error, etc.

The review and approval of each defect resolution or mitigation plan shall be tracked by TennCare, using the following data elements:

- Defect unique identifier
- Defect version (and description if appropriate)
- Status, with values to include In review (specifying which stage), In revision, and Approved
- Revisions required (description)
- Date approved
- Approved by (Person name and Organization role)

The approved version of the defect resolution or mitigation plan shall be captured and linked to the approval record.

# 10. Appendices

## 10.1. Testing Principles: Implications and Rationale

This section outlines the Guiding Principles for Test Management of TennCare Solutions. Guiding Principles are statements of intent or purpose, and described preferred practices. They define the general rules that an organization will use to deploy all business resources and assets, and help provide a framework where more detailed strategies and solutions can be used.

### 10.1.2. Complete Testing

*Table 16: Complete Testing*

| | |
|---|---|
| **Description** | Testing is a process to evaluate whether a developed solution meets specified requirements. Testing also assures that there are no unintended consequences when changes are made to a solution.<br><br>TennCare requires that all solutions be rigorously and completely tested, to assure functionality and adherence to requirements and design. |
| **Rationale** | The users of TennCare's IT need automated solutions that work effectively and fulfill their needs, so that they can perform tasks reliably and efficiently.<br><br>TennCare is entrusted with personal and sensitive data that must be protected, according to the non-functional requirements for security. |
| **Implications** | All requirements must be tested, using the Requirements Traceability Matrix to link each requirement to test cases/scripts/scenarios.<br><br>TennCare business representatives must perform User Acceptance and Beta Testing, and accept the results, to show that solutions meet their expectations.<br><br>TennCare IS representatives must perform technical testing to ensure that all functions of a solution work correctly, and that all non-functional requirements are met (including performance and security). |

### 10.1.3. Integrated End to End Process Testing

Table 17: Integrated End to End Process Testing

| | |
|---|---|
| **Description** | TennCare will manage testing of end-to-end business processes across solutions required to enable each process and the Integration Services Layer (ISL) that manages the integration of solution modules. |
| **Rationale** | TennCare strategy of a modularized Medicaid Management Information System connected by an ISL means that end-to-end business processes will be enabled by multiple solutions provided by multiple solution providers |
| **Implications** | ISL solution vendor must be able to test connectivity of solution modules that use the ISL services.<br><br>Module solution vendors and ISL must be able to test each ISL service they use against functional and non-functional ISL requirements<br>Solution vendors must be able to manage “time travel” requirements across the ISL for end-to-end business processes that have longer cycle times (e.g., month-end reporting, meeting TennCare response times (e.g., Medicaid enrollment time)<br><br>Solution vendors should introduce stubs (like CA LISA), that captures and simulates the behavior data and performance characteristics of the complete composite application environments. Stubs enable the simulated services available for development and test teams throughout the software lifecycle for faster time-to-market with quality software functionality at lower infrastructure cost. This will allow testing for interfaces that have not been released to testing yet or that are still in development.<br><br>ISL solution vendor must be able to report on results of integrated testing of end-to-end business processes to TennCare and vendors of solution modules in scope of the end-to-end tests<br>ISL and module solution vendors must use test tools that have the interoperability to share test results from each solution model in scope of end-to-end test |

### 10.1.4. Coordinated Release Management Testing

Table 18: Coordinated Release Management Testing

| | |
|---|---|
| **Description** | TennCare will coordinate the testing of new releases of the Integrated Services Layer modules and function modules that subscribe to the ISL. |

| | |
|---|---|
| **Rationale** | With the introduction of the ISL and subscribing function modules, TennCare will have to coordinate multiple releases of multiple solutions across the ISL, including new releases of modules within the ISL |
| **Implications** | TennCare requires a robust and standardized release management process and enabling tools that coordinate and orchestrate multiple releases of solutions, including both function modules and the ISL components |

### 10.1.5. Traceable Defect Management

Table 19: Traceable Defect Management

| | |
|---|---|
| **Description** | To facilitate defect management in an integrated modular environment, all solution vendors must be able trace a defect to the requirements that it fails, test cases that detected it and the modules containing the defect |
| **Rationale** | TennCare strategy of a modularized Medicaid Management Information System connected by an ISL means that, when running end-to-end business processes tests, there is considerable complexity to identify a defect, document which test cases generated it, trace it to solution modules or services that caused it.<br>Traceable defect management is critical to being able to pinpoint which solution and which vendor is responsible for resolving the defect |
| **Implications** | Solution vendors must be able to trace defects back to requirements and modules in their own modules and to interfaces with the ISL.<br>The ISL solution vendor must be able to support the location of defects occurring in end-to-end tests involving the ISL and interfaces with subscribing modules<br>Testers must provide clear and comprehensive documentation of test cases, test scripts, test scenarios and test data to support reproducibility of a defect by the responsible module vendor |

### 10.1.6. Rigorous Regression Testing

Table 20: Rigorous Regression Testing

| | |
|---|---|
| **Description** | TennCare operates a rigorous process for testing fixes and changes to solution modules as these changes are promoted through test phases and environments from unit to integration to acceptance to operational readiness testing through to production. |

| | |
|---|---|
| **Rationale** | With the introduction of the ISL and subscribing function modules, TennCare will have to test fixes and changes to multiple different solution modules across the ISL, including fixes and changes to releases of modules within the ISL |
| **Implications** | Solution vendors must support technical change management testing with increasingly automated regression testing<br>Solution vendors must evolve their automated regression testing on a priority basis by focusing on automated testing of priority business processes<br>Solution vendors must follow TennCare's enterprise process for technical change management which is automated in ServiceNow |

### 10.1.7. Transparent Test Reporting

Table 21: Transparent Test Reporting

| | |
|---|---|
| **Description** | Solution vendors will report on test progress and results to TennCare in a form that facilitates aggregate test information for transparent governance and management across the scope of test operations. |
| **Rationale** | With the introduction of the ISL and subscribing function modules, TennCare will have to oversee test operations by solution vendors testing releases of function modules and ISL components.<br>The scope and volume of testing across function modules and ISL releases requires significant transparency to test reporting to govern and orchestrate test operations at the module release, solution vendor, and TennCare levels. |
| **Implications** | Solution vendors must follow TennCare standards for reporting on test progress and results including data standards for test reports and metrics.<br>The ISL solution vendor must be able to aggregate test report data from solution vendors when testing subscribing function modules to support an integrated and transparent view of test results across all vendors. |

## 10.2. Key Terms Glossary

*Table 22: Key Terms Glossary*

| Term | Definition | Referenced Section |
|---|---|---|
| **Defect Mitigation** | A method of handling a defect without resolving (fixing) it. Includes work-arounds, deferrals, writing documentation, providing training, or other remediation. | 9 |

| Term | Definition | Referenced Section |
|---|---|---|
| **Defect Resolution** | An action that resolves a defect, and closes out the defect record. May be a fix (modifying software code or configuration). May be a determination that there was no defect, or that the defect no longer needs to be fixed (for example, it is in a component that has been removed from the solution). | 9 |
| **Function module** | A function module performs a business function, such as Eligibility and Enrollment or Financial Management. It is distinct from the Integration Services Layer.<br><br>Statements about function modules also apply to the components of, or the whole of, a non-modularized solution. | 2.3 |
| **Functional requirements** | A functional requirement defines a system or its component. It describes the functions a software must perform. | 3.5 |
| **Integration Services Layer (ISL)** | The Integration Services Layer (ISL) is a planned solution, composed of the “ISL component”: technology tools that move data between modules and/or control access to data by users and third-party modules. See section 2.3 for components of the ISL. | 2.3 |
| **Module** | A module may be a commercial software product or custom-built code. These modules are integrated, configured, and customized to form a solution. | 2.3 |
| **Module Release** | A module release is a version of module code, configuration or customization, delivered at one time. There may be multiple releases of a module over time, at a frequency determined by business requirements and the solution vendor’s methodology. | 2.3 |

| Term | Definition | Referenced Section |
|---|---|---|
| **Non-functional requirements** | A non-functional requirement defines the quality or performance attribute of a software system. These requirements represent a set of standards used to judge the specific operation of a system. | 3.5 |
| **Release Note** | Release note is a necessary document that is prepared before the Vendor tested product is released for the use of the Business Stakeholder and UAT Team. This document is prepared by the Vendor team and it contains information about the new enhancements and known issues that are part of a specific project release. The details provided encountered by the development and SIT team, while developing and testing the product.<br><br>The importance of these release notes is immense, as they are the communication documents shared with Business Stakeholder and UAT Team. Moreover, these documents usually consist of brief as well as detailed information about the project's development process. A release note is usually a terse summary of recent changes, enhancements, and bug fixes in a particular software release and it provides information that is clear, correct and complete. However, one should not confuse release note with the user guides, as it is not a substitute for the latter. | 3.8 |
| **Solution** | A solution is a combination or configuration of computer application software and infrastructure components that automate some processes within a business function. A solution may consist of one or more modules. | 2.3 |
| **Solution Project** | A solution project delivers a new or upgraded solution. | 2.3 |

| Term | Definition | Referenced Section |
|---|---|---|
| **Solution Vendor** | A solution vendor is a service provider contractor that designs, delivers, and implements a solution. They may also provide operations, maintenance, support, or other services related to the solution. The solution vendor should establish set maintenance windows across the suite of applications. Automated or manual regression smoke tests should be conducted after each deployment to ensure the integrity of the application. This process is ment to reduce the amount of invalid defects caused by unknown system downtimes. | 2.3 |
| **Test Content** | Test content includes test cases, scripts, and scenarios.<br><br>The test content may be written for a user to manually test, for technical testing by a developer or tester working manually, or may be encoded for an automated testing tool. Test content may be specified for the module, point-to-point, or end-to-end scope of testing. | 7.1 |

## 10.3. Test Types Glossary

Table 23: Test Types Glossary

| Test Type Name | Definition + Objective |
|---|---|
| **Accessibility Test** | The Accessibility Test is to ensure that the product is compliant with applicable Section 508 of the Rehabilitation Act of 1973. Section 508 Testing is performed by the system developer/maintainer to ensure that the EIT product is compliant with applicable Section 508 Accessibility Standards identified in the completed Section 508 Product Assessment. A testing contractor will perform Section 508 Testing to ensure that the EIT product is compliant with applicable Section 508 Accessibility Standards identified in the completed Section 508 Product Assessment.<br><br>Software products (whether COTS, Government Off-the-Shelf (GOTS), or custom-developed software applications) must adhere to Section 508 accessibility and other regulatory requirements governing the use of EIT in accordance with the CMS Policy for Section 508 Compliance. Accessibility Testing is required if the business application has a user interface or produces electronic output for direct access or use by federal employees or the public. |
| **Alert/Monitoring Test** | The Alert/Monitoring Test is the type of testing that is done where you purposely end test scenarios/cases in actions that would result in a system alert/message, ensuring that the correct actions are taken at that time. Alerts could be from a user perspective (telling the user that they provided the wrong SSN because it begins with 999), or on the system side (the system has received error code 01239, and the system knows how to handle that error code). Monitoring testing is to validate that, for known errors/messages, the system should be ready to monitor for those and intercept them and react to them appropriately. |
| **Batch Testing** | Batch Testing is a group of tests that is sequentially executed. Every Batch Test consists of multiple dependent test cases. In those batches every end state is the base state to next case. Batch Testing is also known as a Test suite or Test belt. |

| Test Type Name | Definition + Objective |
|---|---|
| **Beta Testing Stage** | The Beta Testing Stage is a stage of testing where the users (TennCare Representatives) are enabled to test the full and completed solution, prior to formal Go-Live. Business and/or IS users will ensure that a module works as expected, in concert with other integrated modules.<br>Beta testing is also known as pilot testing or parallel testing. The scope of Beta testing is end-to-end testing of the full solution (including all its modules and interfaces). The scenarios normally encountered in daily operations are tested, rather than a set of formal test cases. More information on the Beta Testing Stage is provided in section 3.5.5. |
| **Boundary Test** | The Boundary Test consists of testing the extremes of the input domain, e.g. maximum, minimum, just inside/outside boundaries, typical values, and error values. |
| **Capacity Test** | The Capacity Test determines how many users and/or transactions a given system will support and still meet performance goals. |
| **Compatibility Test** | The Compatibility Test validates how well a software performs in a particular hardware/software/operating system/network environment. Backward Compatibility Testing tests the application or software in old or previous versions. Forward Compatibility Testing tests the application or software in new or upcoming versions. |
| **Compliance Test** | The Compliance Test determines whether the solution complies with specified security standards or procedures. |
| **End-to-End Test** | End-to-end testing is to ensure that the entire product is functional from beginning to end, meaning the application flow behaves as expected. It defines the product's system dependencies and ensures all integrated pieces work together as expected. |
| **End-to-End Testing** | End-to-end testing is the testing to verify that the actual solution modules and interfaces, as developed and configured, function together as intended. This ensures that all steps in a process can be successfully executed, through the many modules and connections involved. More information on end-to-end testing is provided in section 3.4.3. |

| Test Type Name | Definition + Objective |
|---|---|
| **Error Handling Test** | The Error Handling Test assesses the ability of the system to properly process erroneous transactions. The main objectives are to ensure that all reasonably anticipated error conditions are recognizable by the application system, accountability for processing errors has been assigned, that the procedures provide a high probability that the error will be properly corrected, and that reasonable control is maintained over errors during the correction process. |
| **Exploratory Test** | The Exploratory Test is an unscripted and improvisational testing method that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the quality of his/her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project. |
| **Functional Test** | The Functional Test assesses the input/output functions of a business application against pre-defined functional and data requirements. Each and every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results. Types of functional testing include: Unit Testing, Smoke Testing, Sanity Testing, Integration Testing, White box testing, Black Box testing, User Acceptance testing, Regression Testing, etc. |
| **GUI Navigation Test** | The GUI Navigation Test validates the system logic behind when a user navigates from one screen to another. In a GUI system, at each time frame there is an active screen interacting with the user. The active screen when triggered by specific event, will disappear or be deactivated and another one will be loaded in or activated. The two screens are logically connected by the event and such a scenario where the screen focus is shifted is called screen navigation. |
| **GUI Software Test** | The GUI Software Test ensures that the graphical user interface meets agreed upon specifications as defined prior to software development. GUI testing evaluates design elements such as layout, colors, fonts, font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links and content. |

| Test Type Name | Definition + Objective |
|---|---|
| **Load Test** | The Load Test tests a system's performance under a specific expected load to determine how the application behaves when multiple users access it simultaneously. Its purpose is to ensure smooth functioning of the software under real-life load conditions. Load Testing is a Non-Functional Test and a subset of the Performance Test. |
| **Localization Test** | The Localization Test is to ensure that the product behaves according to the local culture or settings. For example, it tests that the UI, default language, currency, date, time format, and documentation are designed as per the targeted country or region. The major area affected by localization testing includes content and UI. It ensures that the application is capable enough for use in that particular country. |
| **Module Testing** | Module testing is the testing of one module within one vendor's solution project.<br><br>Module-specific tests are needed within each solution project. Module testing is needed for function modules, and for ISL components within the ISL, and for solutions not being integrated with the ISL. More information on module testing is provided in section 3.4.1. |
| **Negative Test** | The Negative Test assesses the response of the system outside of normal parameters and is designed to assess the system's ability to successfully perform error handling with the unexpected input. |
| **Non-Functional Test** | The Non-Functional Test is designed to test the non-functional aspects (performance, usability, reliability, security, etc.) of a software application. It is explicitly designed to test the readiness of a system as per nonfunctional parameters which were not addressed by functional testing. Types of non-functional testing include, but is not limited to: Performance Testing, Capacity Testing, Availability Testing, Volume Testing, Stress Testing, Load Testing, Reliability Testing, Scalability Testing, Recovery Testing, Vulnerability Testing, Compliance Testing, Accessibility Testing, Security Assessment Test, Role-Based Access Testing, Penetration Testing, and Disaster Recovery Testing. |

| Test Type Name | Definition + Objective |
|---|---|
| **Operational Readiness Testing (ORT) Stage** | The Operational Readiness Testing (ORT) Stage is a stage of testing where solution vendors ensure that their module, integrated with other modules, complies with non-functional requirements. It does not involve business users.<br><br>The ORT stage has two sub-stages, conducted in this sequence:<br><br>1. Point-to-point testing of all interfaces and connections to or from the module<br>6. End-to-end testing of all defined scenarios involving the module<br><br>More information on the ORT Stage is provided in section 3.5.4. |
| **Penetration Test** | The Penetration Test tests a computer system, network, or web application to find security vulnerabilities that an attacker could exploit. This can be automated with software applications or performed manually. Either way, the process involves gathering information about the target before the test, identifying possible entry points, attempting to break in and reporting back the findings. This is also called pen testing or ethical hacking. |
| **Performance Test** | The Performance Test verifies that the application and technical environment will properly support the anticipated increased transaction volumes, according to an acceptable set of response or elapsed time metrics. It assesses the capacity and throughput of a business application and/or infrastructure in processing time, CPU utilization, network utilization, and memory and storage capacities relative to expected normal (average and peak) user, and processing load as defined in the system's requirements document and/or Operation Manual (OM) document. Performance testing can be used to establish a baseline against which future performance tests can be compared against. |
| **Point-to-Point Test** | Point-to-point testing is to ensure that all parts of a product are functional from any two specific points. It is similar to the end-to-end test but only covers a portion of the product as opposed to the entirety of it. |
| **Point-to-Point Testing** | Point-to-point testing is the testing of the connection or interface between each pair of modules needed. More information on point-to-point testing is provided in section 3.4.2. |

| Test Type Name | Definition + Objective |
|---|---|
| **Positive Test** | The Positive Test is performed on the system by providing the valid data as input. It checks whether an application behaves as expected with positive inputs. This test is done to check if the application is performing as expected. |
| **Recovery Test** | The Recovery Test verifies the system's ability to recover from points of failure like software/hardware crashes, network failures etc. This is to determine whether operations can be continued after a disaster or after the integrity of the system has been lost. It involves reverting to a point where the integrity of the system was known and then reprocessing transactions up to the point of failure. |
| **Regression Test** | Regression testing is a form of software testing that confirms or denies a software's functionality after the software undergoes changes. Whenever there has been a change to software, there is the possibility of unintended consequences. Therefore, the software must be tested to confirm that the quality of the system has not regressed. Regression testing is ideally performed every time a software component or feature is modified, to help identify (and resolve) any newly discovered or regressed issues. The Regression Test monitors the operational availability of business applications and/or infrastructure, problems/incidents, performance/service level, and capacity utilization of production systems, and will validate the gathered data against expected results (documented in the system's requirement document and/or Operation Manual (OM) document) to ensure that the implemented application or infrastructure performs as expected in production. This testing function is sometimes referred to as Reliability Validation, Burn in Period, Reliability Test, or Extended Reliability Test. |
| **Reliability Test** | The Reliability Test monitors the operational availability of business applications and/or infrastructure, problems/incidents, performance/service level, and capacity utilization of production systems, and will validate the gathered data against expected results (documented in the system's requirement document and/or Operation Manual (OM) document) to ensure that the implemented application or infrastructure performs as expected in production. This testing function is sometimes referred to as Reliability Validation, Burn in Period, Reliability Test, or Extended Reliability Test. |

| Test Type Name | Definition + Objective |
|---|---|
| **Role Based Access Test** | The Role Based Access Test validates that the intended security permissions can prevent unauthorized end users from accessing information and performing functionalities with the given role in the application. |
| **Sanity Test** | Sanity Tests are a subset of regression testing and is performed to ensure that the new code changes (proposed functionality) works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing. It is performed only after the build has cleared the smoke test and been accepted by the Quality Assurance team for further testing. |
| **Scalability Test** | The Scalability Test determines a software's ability to maintain satisfactory performance when any of its non-functional capability requirements is scaled up, such as the user load supported, transaction rate, data volumes, etc. Scalability testing is a subset of the Performance Test. |
| **Security Assessment Test** | Security Assessment Tests validate all applicable security controls, including those defined in the CMS Policy for the Information Security Program. They validate that the security controls of business applications or of infrastructure are implemented correctly, operate as intended, and produce the desired outcome. This should include vulnerability testing, static code scans, dynamic code scans, penetration testing, and/or compliance with testing security standards and policy. |
| **Smoke Test** | The Smoke Test is a type of software testing that determines whether the deployed build is stable or not. The purpose of Smoke Tests it to confirm whether the QA team can proceed with further testing. Smoke tests are a minimal set of tests run on each build. It is also called as Build Verification Testing or Confidence Testing. |
| **Static Regression** | Static Regression is a type of regression testing that is fixed across all Major and Minor releases. The purpose of Static Regression tests is to confirm the baseline functionality has not been impacted by the release. These tests are expected to pass, regardless of whether the component or related functionality has changed. |

| Test Type Name | Definition + Objective |
|---|---|
| **Stress Test** | The Stress Test is designed to determine how well the environment and application can maintain a desired level of effectiveness under unfavorable conditions. This is also useful to determine the breaking point of your system, to determine if design specifications have been met and to document how the system fails as load increases. |
| **System Integration Testing (SIT) Stage** | The System Integration Testing (SIT) Stage is a stage of testing where solution vendors ensure that their module meets requirements for integration with other modules including the ISL. It does not involve business users. In SIT, functional and non-functional requirements must be tested, both point-to-point and end-to-end. More information on the SIT Stage is provided in section 3.5.2. |
| **Time Travel Test** | The Time Travel Test is the act of testing the date and time-sensitive functionalities in order to validate business rules and logic that exists in a software stack. |
| **Unit Testing Stage** | The Unit Testing Stage is a stage of testing where the solution vendor performs Module Testing within their module or component. More information on the Unit Testing Stage is provided in section 3.5.1 |
| **Usability Test** | The Usability Test is performed by end users to verify the appropriate level of ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component. |
| **User Acceptance Testing (UAT) Stage** | The User Acceptance Testing (UAT) Stage is a stage of testing where solution vendors ensure that a module meets business user needs, in concert with other integrated modules. Users are asked to manually test all defined scenarios that involve the module. More information on the UAT Stage is provided in section 3.5.3. |
| **Volume Test** | The Volume Test tests the system's performance by increasing the volume of data in the database to evaluate the impact on response time and system behavior when exposed to a high volume of data. It is also referred to as flood testing. |

| Test Type Name | Definition + Objective |
|---|---|
| **Vulnerability Test** | The Vulnerability Test determines if there are any security vulnerabilities in an organization's personnel, procedures or processes. Vulnerability Tests will automatically scan for known vulnerabilities. |

## 10.4. Test Management Data Requirements

Certain data elements must be collected and maintained to manage testing:

- The data elements with a checkmark in the Test Content column of Table *23* are required for reporting and approval of test cases, scripts and scenarios, as described in section 7.1.
- The data elements with a checkmark in the Test Results column of Table *23* are required for reporting and approval of test results as described in section 8.1.
- The data elements with a checkmark in the Defects column of Table *23* are required for reporting and approval of defects as described in section 9

These requirements are at a conceptual level. These data element may be named and structured in any appropriate way by vendors and tools. Unique identifiers for all entities are required but not listed below. Collect multiple records as appropriate (such as multiple test results of one test case). Additional data elements may be collected as appropriate.

Table 24: Data requirements for Test Content, Results and Defects

| Data element | Description | Test Content (Cases, Scripts & Scenarios) | Test Results | Defects |
|---|---|---|---|---|
| Solution Name | Identifies the solution to be tested. For integration tests, identify one primary solution. | ✓ | ✓ | ✓ |
| Module Name | Identifies the solution module to be tested. For integration tests, identify one primary module. | ✓ | ✓ | ✓ |
| Module Version | Identifies a version or release of the module's code, configuration or customization. | | ✓ | ✓ |

| Data element | Description | Test Content (Cases, Scripts & Scenarios) | Test Results | Defects |
|---|---|---|---|---|
| Related Modules | Reference to other solution modules that are involved in the test. | ✓ | ✓ | ✓ |
| Test Stage | The stage of testing: Unit, SIT, UAT, ORT, or Beta. | ✓ | ✓ | ✓ |
| Test Subject | A name of, or reference to, the thing being tested, such as a part of a module, an interface, or an end-to-end scenario. | ✓ | ✓ | ✓ |
| Requirement | A reference to the requirement that is being tested | ✓ | ✓ | ✓ |
| Test Dataset Name | A reference to the test dataset to be used for the test | ✓ | ✓ | ✓ |
| Test Dataset Version | Identifies a version or release of the test dataset | | ✓ | ✓ |
| Environment Name | Name the server (such as DEV, SIT, STAGE, and PREPROD) where the test was conducted. | | ✓ | ✓ |
| Test Content Name | Short name or title of the test case, script or scenario | ✓ | ✓ | ✓ |
| Test Content Description | High level description of test case, script or scenario | ✓ | ✓ | ✓ |

| Data element | Description | Test Content (Cases, Scripts & Scenarios) | Test Results | Defects |
|---|---|---|---|---|
| Test Content Revision Date | The date when this test case, script or scenario was last revised. (Keep versions of test content.) | ✓ | ✓ | ✓ |
| Test Preconditions | What needs to be set up to create the correct situation to test | ✓ | | |
| Test Step Actions | The exact and specific steps necessary to follow to complete the test. These steps should be clear, concise, and written so that they are easy to follow by testers who are subject matter experts. | ✓ | | |
| Test Acceptance Criteria | Specific outcome that would signify the requirement is met | ✓ | | |
| Test Content status | Values such as:<br>• Proposed<br>• Approved<br>• Executed<br>• Outstanding (testable)<br>• Outstanding (untestable because of defect)<br>• Failed | ✓ | ✓ | |
| Test Execution Date | The date when a test was run. | | ✓ | ✓ |
| Test Result | Values such as:<br>• Passed<br>• Failed | | ✓ | |

| Data element | Description | Test Content (Cases, Scripts & Scenarios) | Test Results | Defects |
|---|---|---|---|---|
| Tester | Identifies the person who executed the test, or the automated testing software. This is also the defect initiator, if applicable. | | ✓ | ✓ |
| Test Evidence | Records that show that the test was done and what the results were. May include screen-shots, reports, mouse clicks, keyboard entry, or query results, that show the completion of each step.<br><br>Re-tests after defect fixes will also collect this evidence. | | ✓ | ✓ |
| Defect Description | Summary description of the unexpected result. | | | ✓ |
| Defect Status | Values such as:<br><br>• Identified<br>• Source Located<br>• Mitigation Proposed<br>• Mitigation Approved<br>• Cancelled<br>• Fix In Progress<br>• Fix Ready to Test<br>• Fix Tested Successfully<br>• Resolution Approved | | | ✓ |
| Defect Date Identified | The date when a defect was first identified (a Test Execution Date). | | | ✓ |
| Defect Status Dates | Records of each date that the defect status changed. | | | ✓ |

| Data element | Description | Test Content (Cases, Scripts & Scenarios) | Test Results | Defects |
|---|---|---|---|---|
| Defect Severity | • The level of impact of the defect. Values defined in Table 24. | | | ✓ |
| Defect Cancellation Reason | Values such as:<br>• Tester error<br>• Unable to reproduce<br>• Operating as designed | | | ✓ |
| Defect Cause | Describes the root cause of the defect, including the source module, and the nature of the problem. | | | ✓ |
| Defect Resolution Action | Summary of the fix or other action (e.g. training) planned or attempted. | | | ✓ |
| Defect Planned Resolution Date | The date when a defect is expected to be fixed and re-tested. | | | ✓ |
| Defect Mitigation | Description of a work-around, remediation, documentation, training, or other mitigation for a defect that will not be fixed. | | | ✓ |
| Defect Owner | The person who is responsible for the defect resolution and all required follow-up actions. | | | ✓ |
| Defect Date Resolution Approved | The date when a defect's fix was approved as meeting requirements. | | | ✓ |

## 10.5. Defect Severity

Table *23*: Data requirements for Test Content, Results and Defects calls for tracking defects, including the level of severity, as defined in Table 24: Defect Severity.

The Acceptance Criteria (section 5.2) include defect severity criteria. They also mention "blocking defects," which are defects that prevent further testing of the module, component, connection, etc.

*Table 25: Defect Severity*

| Severity | Description | Example |
|---|---|---|
| Critical<br>Sev-1 | A complete failure of the Solution application or supported process in the Production Instance has occurred. There is no work-around for the problem. A majority of end users of the Production Instance are affected or an entire business division is affected or the outage has occurred during a critical business process or period, such as end of month or end of year processing. Critical defects take precedence over all other requests. | • System is down state-wide<br>• Test environment is down and not available to the testing team |
| High<br>Sev-2 | Major issues exist within the Solution or supported process in the Production Instance. The issue affects large portions of the user community. This includes high visibility issues involving upper management or time sensitive issues. An example of this priority level is an impact to multiple users of the system, or an inability to process applicants. | • Incorrect enrolment information for popular Category of Assistance (MAGI COEs) with a cumbersome workaround to do a manual calculation and approve eligibility through manual override |
| Medium<br>Sev-3 | Issues exist with an application or supported process in the Production Instance that affects a few users on a regular basis, thereby preventing some work from being accomplished.<br>Examples of this type of priority would be inability to access implemented functionality or implemented functionality not operating as it should. | • Notice is not being generated for a small subset of cases, but a workaround exists<br>• Page level comments are not functioning, but case comments are functioning<br>• Task is being routed to incorrect queue e.g. Customer Service vs. Change |

| Severity | Description | Example |
|---|---|---|
| Low<br>Sev-4 | An informational inquiry or nonrecurring issue exists with the Production Instance that affects a few non-critical users or processes. Workarounds are readily available. | • Spelling correction on a page<br>• Extra page that appears in the driver flow but allows the user to continue without having to enter data<br>• Extraneous non-required field on a page<br>• Enhancement request to page layout |

## 10.6. Suggested Testing Metrics

The following section provides a list of suggested testing metrics that TennCare IS and vendors may use to evaluate their testing and defect management activities. The metrics are grouped in the following categories, first for testing, then for defects:

- Timeliness
- Progress to Completion
- Efficiency
- Quality of Testing
- Quality of Development

Table 26: Suggested Testing Metrics – Test Cases

| Category | Metric + Formula |
|---|---|
| Timeliness | Test Cases Production Timeliness (%) = $\left(\frac{\text{\# of Test Cases Produced on Schedule}}{\text{Total \# of Test Cases}}\right) * 100$ |
| Timeliness | Test Results Production Timeliness (%) = $\left(\frac{\text{\# of Test Results Produced on Schedule}}{\text{Total \# of Test Results}}\right) *100$ |
| Progress to Completion | Number of Test Cases Outstanding Caused by Defects (untestable) |
| Progress to Completion | Test Coverage (%) = $\left(\frac{\text{\# of Test Cases Executed}}{\text{Total \# of Test Cases}} *\right) 100$ |
| Progress to Completion | Number of Test Cases Ready |
| Progress to Completion | Number of Test Cases Outstanding Testable |

| Category | Metric + Formula |
|---|---|
| Progress to Completion | Number of Test Cases Outstanding Untestable |
| Progress to Completion | Number of Test Cases Executed |
| Efficiency | Average Test Case Execution Time = $\frac{\sum \text{End Time - Start Time}}{\text{Total \# of Test Cases Executed}}$ |
| Quality of Testing | Test Case Effectiveness = $\frac{\text{\# of Defects Detected by Test Cases}}{\text{Total \# of Defects}}$ |
| Quality of Development | First Run Success Rate (%) = $\left(\frac{\text{\# of Test Cases Passed in First Run}}{\text{Total \# of Test Cases Executed}}\right)$*100 |
| Quality of Development | Test Cases by Status (Passed, Failed, Blocked) (%) = $\left(\frac{\text{\# of Test Cases}_{\text{Passed, Failed, Blocked}}}{\text{Total \# of Test Cases Executed}}\right)$*100 |
| Quality of Development | Rate of Rework (%) = $\left(\frac{\text{\# of Test Cases Reworked}}{\text{Total \# of Test Cases Executed}}\right) * 100$ |

Table 27: Suggested Testing Metrics - Defects

| Category | Metric + Formula |
|---|---|
| Timeliness | Defect Resolution Timeliness (%) = $\left(\frac{\text{\# of Defects Resolved on Schedule}}{\text{Total \# of Defects}}\right) * 100$ |
| Progress to Completion | Defect Aging = $\frac{\sum \text{Date of Detection - Date of Fix (if fixed) or Current Date (if open)}}{\text{Total \# of Defects}}$ |
| Efficiency | Defect Cycle Times = $\frac{\sum \text{Date of Detection-Date of Fix or Current Date}}{\text{Total \# of Defects}}$<br>To be calculated at particular stages of a defect resolution cycle |
| Efficiency | Defect Resolution Efficiency = $\frac{\text{\# of Defects Resolved}}{\text{\# of Defects}}$ |
| Efficiency | UAT Efficiency = $\frac{\text{\# of Defects Detected in UAT}}{\text{\# of Defects Detected in Production+ \# of Defects Detected in UAT}}$ |
| Quality of Testing | Defect Leakage (%) = $\left(\frac{\text{\# of Defects Undetected}}{\text{Total \# of Defects}}\right) * 100$ |

| Category | Metric + Formula |
|---|---|
| Quality of Testing | Defect Discovery Rate = $\frac{\text{\# of Defects Detected During Testing}}{\text{Total \# of Defects}}$ |
| Quality of Development | Delivered Defect Density = $\frac{\text{\# of Defects Resolved}}{\text{Size of Code Delivered}}$ |
| Quality of Development | Average Number of Attempts to Fix a Defect = $\frac{\text{\# of Defect Resolution Attempts}}{\text{\# of Defects}}$ |
| Quality of Development | Root Cause Analysis (RCA) = Count of defects, by source |

## 10.7. Capabilities


Figure 8: Capabilities for Lifecycle Management

Figure 8 shows that a Program or Project Management Office should use a tool for solution Lifecycle Management, such as tracking which solution has passed which gate approval. This tool needs to be connected to a repository that stores deliverable documents such as Test Plans. On complex projects, vendors will need access to this tool and repository or will need to maintain similar information in parallel.

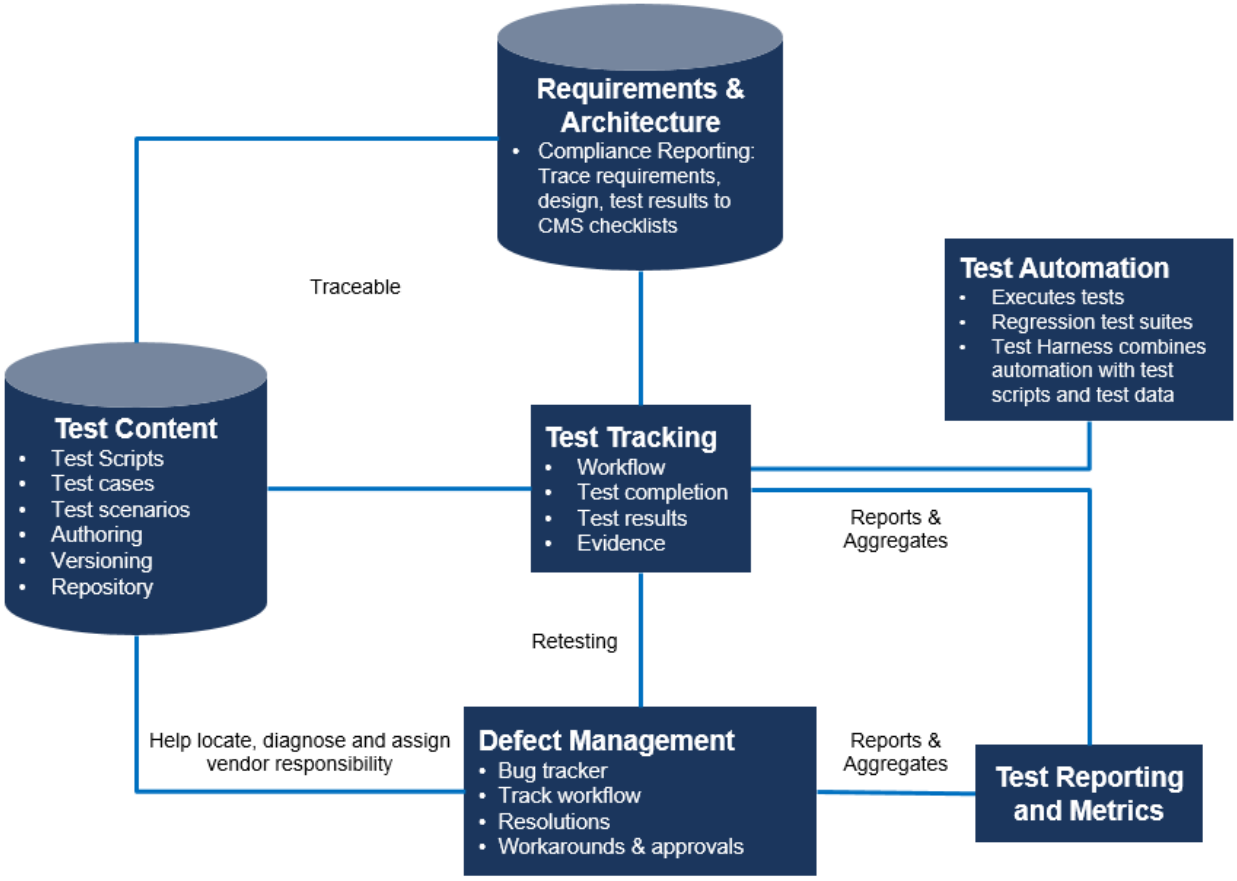



Figure 9: Capabilities for Test and Defect Management

In Figure 9, each test case, script, or scenario shall be captured in a Test Content repository, and shall be traceable to a Requirement. The Requirements and Architecture repository is used for tracking compliance to Centers for Medicare and Medicaid Services (CMS) checklists.

A Test Tracking capability will track each test that is executed, and link it to the Test Content and Requirement being tested. Tracking the test workflow includes recording who executed a test, when, using which test dataset, and which version of the module. Test results record the evidence (such as screenshots) for which tests succeeded and failed. The results of both automated and manual tests will be recorded and the data should be available for reporting.

Standardize on an automation platform(s) and establish automation repository that can be used by all testing teams. A Test Automation tool should be used to execute large suites of tests repeatedly for regression testing.A test automation tool is the best way to increase the effectiveness, efficiency and coverage of your software testing.

Benefits of Test Automation tool:

- Set of tests requiring same actions with different inputs can be repeatedly executed.

- Automated tests can be run at the same time on different machines with different OS platforms.
- Automating the testing process significantly faster than manual testing as it consumes more time.
- Eliminates the human error when running the same operations every time.

A Defect Management tool, or “bug tracker,” can be used to track the workflow described in section 9.1. This tool will record defect resolutions (fixes), as well as mitigations, and approvals thereof. Locating defects will require reference to the test cases, scripts, scenarios, and results. Defect resolutions trigger re-tests and regression tests, which are recorded in the Test Tracking tool. Use of more effective and popular Tracking Tools such as Jira, Pivotal Tracker over ALM is preferred to improve visibility of testing status, action items, defects, and project deliverables.

The Test Reporting and Metrics capability is likely to be included in one or more other tools used for test and defect management. Data required about tests and defects, as specified in section 10.4, must be available to the reporting tool. The tool used for reporting must be able to output a variety of reports, both of individual records (such as a list of tests executed) and aggregations (such as the number of tests executed, by stage). Customizing the format, data content, and frequency of reports must be possible. The reporting tool must be able to calculate summary metrics, including custom-defined measures. Reports and metrics must be output in spreadsheet files (.csv or .xlsx) and PDFs, and may also be available in dashboards or other formats.

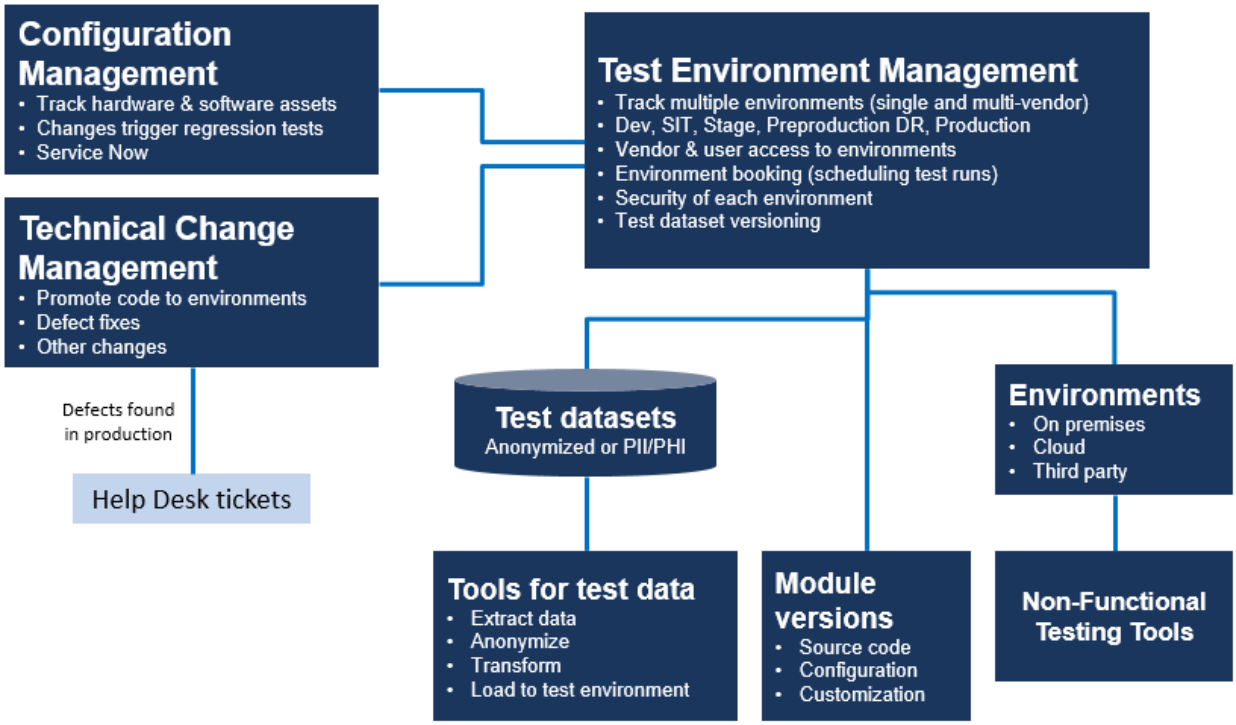



Figure 10: Capabilities Related to Test Environments and Change Management

In Figure 10, we see capabilities for managing the technical configuration to be used for testing. These capabilities are needed by TennCare IS and/or the IS Vendor. A solution vendor may need access to some of these tools.

The Test Environment Management capability will coordinate use of the many shared Environments needed for the multiple testing stages of multiple solutions being integrated. This capability includes scheduling (booking) usage of each environment, managing demand for environments, and informing stakeholders when an environment is unavailable.

There should be tracking of the Module Versions and Test Datasets added to each Environment, as well as removed from environments or archived.

Test datasets require tracking information as described in section 7.2. It is important to track which datasets include sensitive information (PII, PHI, etc.) and therefore have restricted access.

A tool to extract, transform, de-identify, and load test data will be needed.

Tools will also be needed to perform tests of performance, security and other non-functional requirements.

TennCare's tool for Technical Change Management and its Configuration Management database, are involved when promoting a module from Testing phase to Implementation phase.

They are also involved in fixing defects and regression testing. After go-live, Help Desk support tickets are a source of defects to be resolved through Technical Change Management.