Integrated Eligibility & Enrollment System (IEES)
Test Plan and Artifacts Report

Version: 1.0
Last Modified: February, 2023

APPROVALS

Submitting Organization's Approving Authority:

Signature	Printed Name	Date	Phone Number

CMS' Approving Authority:

Signature	Printed Name	Date	Phone Number

REVISION HISTORY

Version	Date	Organization/Point of Contact	Description of Changes
1.0			Baseline Version

Table of Contents

INTRODUCTION

Kentucky's Integrated Eligibility and Enrollment System (IEES), along with kynect Benefits, kynect HealthCoverage and Medicaid Management Waiver Application (MWMA) portals, is designed to provide Medicaid, Supplemental Nutrition Assistance Program (SNAP), Temporary Assistance for Needy Families (TANF), Childcare, State supplementation payments, MWMA, Kentucky Level of Care System (KLOCS), Kentucky Integrated Health Insurance Premium Payment (KI-HIPP) program, and State-based Marketplace services to Kentuckians.

Kentucky's systems and testing team is led by The Office of Application & Technology Services (OATS), Division of Eligibility Systems (DES). This division hires testers and business analysts with specified qualifications. DES also has testers, business analysts and developers within the Quality Management & Improvement Branch. This testing team structure includes the combination of these specific staff members and team members from the contracted vendor, Deloitte. Statements of Work (SOW) between Deloitte and Kentucky include a Responsibility Assignment Matrix, (RACI Matrix) that outlines tasks and who is responsible, accountable, consulted, or informed of specific activities.

Various tasks are managed by OATS' Deloitte IEES MO&E SOW, which is in place currently. In alignment with expectations from the CMS MES Testing guidance, Kentucky notes that CMS reviewed and approved the Deloitte IEES MO&E Support Extension Statement of Work (SOW) Master Agreement 758 1300000392-1, under submission KY-2022-11-10-EE-SOW-IEES MOE Support Extension. The PHE Unwinding changes are handled through M&O.

Deloitte has designed, developed and implemented an integrated multi-layer IEES solution on .net and Salesforce platforms to deliver these services. As part of the Public Health Emergency (PHE) planning, Kentucky accounted for the eventual re-configuration to end, or unwind specific functionality in these systems that was implemented for the PHE.

Deloitte and OATS are responsible for executing the testing to ensure monthly/major as well as minor releases to IEES, kynect and MWMA are delivered with the required level of quality. This Test Plan documents the proven testing strategy that has been jointly developed and modified over the years to include the various types of testing required to support a system like IEES and includes unit, integration, functional, user-acceptance, interface, data migration, automation, performance, security and regression testing.

Testing will be conducted as part of standard System Development Life Cycle (SDLC) process. Testing Phases include the following:

- Partner Integration Testing (PIT) - Testing includes partner integration testing in order to simulate the data that partner system are to ingest. Partner system includes but not limited to: MMIS, CMS, MCOs.
- System Integration Testing (SIT) - SIT includes verifying the functional system and corresponding interfaces. During SIT planning, test scenarios will be mapped to PHE requirements and scripted to validate the functionality is covered in testing.
- User Acceptance Testing (UAT) - UAT will validate that the functional release is performing satisfactorily in accordance with PHE design specifications. Daily reporting of test case execution and defects found will be provided by change request. During UAT

testers will ensure that post PHE changes that the system is stable, converted data can be processed, and the system can process the administrative work.

- Regression Testing - Regression Testing will include the re-execution of PHE scenarios to ensure that the constant builds into the wider system that were already executed were not altered by the frequent weekly UAT builds. Regression testing occurs in sync with the development process to test parts of the solution to confirm the stability of the development and reduce risk related to introducing unexpected defects to users while modifying system code. UAT testers use a combination of both manual and automated regression test scripts both simulated (data created) and converted data.
- Operational Readiness Testing (ORT) – ORT is done as part of all IEES releases where key Medicaid functionality is tested.
- End-to-end test scenarios are included in all phases of testing.

A report of testing will be provided at the conclusion of testing for each release and Kentucky is happy to share testing knowledge regularly.

- Release 23.03, implementation date 3/31/2023
- Release 23.04, implementation date 4/28/2023
- Release 23.05, implementation date 6/2/2023

Objectives

The objective of this system artifacts report is to describe the overall test plan strategy. The purpose of the plan is to:

- Describe the testing approach and philosophy
- Identify types of testing, test data, and test environments
- Define test monitoring, defect management

Testing Overview

Testing is an integral part of the Systems Development Life Cycle (SDLC) because it validates the ability of components and systems to meet expectations. Deloitte has developed a set of methods that have been used to shape the standard approach to testing. Deloitte's methodology provides an industry-leading testing approach that integrates testing processes, methods, and tools with testing artifacts produced during the solution development. Deloitte's standards and guidelines for both reporting detection of defects as well as resolution of defects, consist of qualitative values. Deloitte's vulnerability assessment approach is designed to align with leading industry standards such as National Institute of Standards and Technology (NIST) Special Publication (SP) 800-64, Open Source Security Testing Methodology Manual (OSSTMM) version 3.0 and Open Web Application Security Project (OWASP).

Multiple testing tasks are important to identify the systems' abilities to meet functional and technical requirements. Deloitte's experience will bring a structured, low-risk verification approach to testing the IEES solution. The methodology initiates testing activities in the Elaboration phase where we define the specific approaches and business scenarios to support the rest of the IEES project. The following graphic illustrates the close linkage between design and testing activities supporting traceability to business requirements.

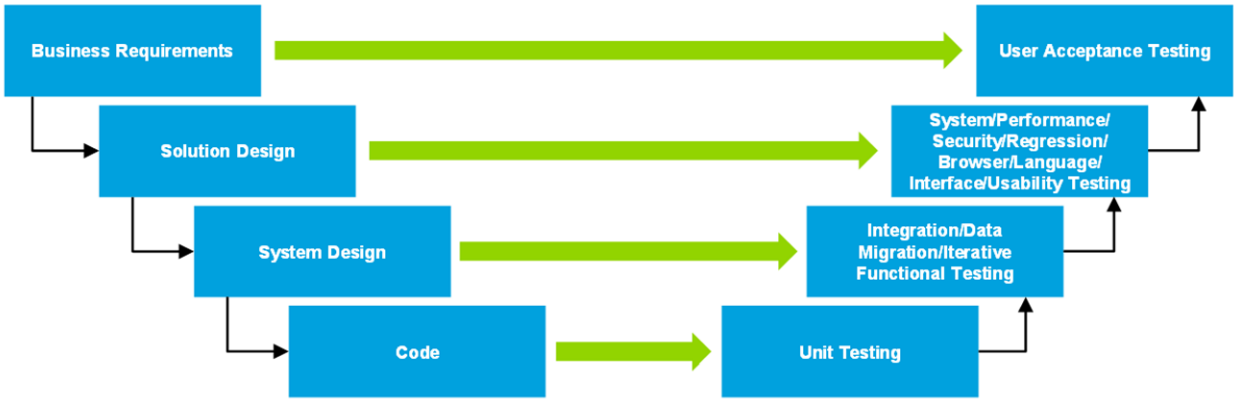


Figure 3-60. Overall Testing and Verification Approach

Deloitte's testing and verification approach has a one to one map to the design activities which helps in identification of meeting the system's functional and technical requirements.

General Approach

Deloitte will execute an incremental testing approach with documented test plans to address the functional and technical portions of the system and environment. The test plan is developed consistent with the overall test strategy and serves as a guide for creating test cases and test data. Creation of the test plan includes, but is not limited to, the following:

- Testing approach
- Roles and responsibilities
- Staffing requirements
- In/Out of scope tasks
- Risks
- Test schedule
- Test tools
- Defect classification
- Description of test environments
- Test data requirements
- Entrance and exit criteria
- Test case management
- Test reporting

Microsoft Team Foundation Server tool will be used to document test scripts for each type of test. It has specific sections for recording the test action, the expected results of running the data, the actual test results to be compared with the expected results. This provides a common approach to documenting tests regardless of the type of test. This also provides a single repository for the data related to testing including defect tracking. TFS provides the capability to extract and generate a Requirement Traceability matrix that can provide the test coverage details with each release

Deloitte's testing approach includes the following test plans including:

- Unit Testing- provides verification of the hardware or software prior to integration of those items. Unit testing is highly iterative and is an essential aspect of defect management. By performing unit testing as components are developed, defects and issues are identified earlier and as a result are less costly and time-consuming to resolve than defects found later in the testing process.
- Integration Testing - validates that the components are functioning as expected when operated individually or as a group. This involves testing the assembled individual components and tests them with other components.
- Iterative Functional Testing - leverages test case execution using a subset of System test scripts that verifies the components developed for each logical iteration of the system meet all functional and technical requirements as defined and approved in the Test Plan and the Requirements Definition Phase.
- System Testing - testing the software and required hardware/network infrastructure as a whole. System testing will validate the system by simulating the numerous variations of user process flows (both positive and negative),user or application security and system initiated use cases (both positive and negative), and other application requirements.
- Data conversion – Testing includes data integrity checks at both the record level and the data element level.
- Interface - verifies that functional requirements for full integration with other groups are in place for testing. The interface testing will begin early in the testing life cycle and can continue throughout the life of the project should there be a need to add additional components.

- Performance - confirm the performance of the entire technical architecture, which includes the performance of software applications, integration, database, network, and the hardware components included within the scope of the implementation.
- Regression - confirm that previously tested application functionality or critical end-to-end business workflows or system performance has not been adversely impacted by the implementation of new or updated functionality, defect fixes, production fixes or infrastructure upgrades.
- Security - confirm that only authenticated users with the required role(s) are able to access the appropriate functioning of the new IEES solution per applicable security configuration. It also validates operational processes such as granting a user access to an application or logging user activity on an application are working as designed.
- User Acceptance and system (includes usability, language, browser, and Iterative functional) - evaluate the effectiveness as a whole while taking into consideration unique or special usability needs of users.

Testing Philosophy

Deloitte's framework incorporates testing in each phase of the System Development Life Cycle (SDLC), from start to finish. The testing methodology is built upon the following guiding principles and philosophy:

- **Plan Testing Early**. Up-front planning likely facilitates starting to test on time and staying on schedule.
- **Test Early**. It is less costly to fix errors early on in the systems development life cycle rather than later.
- **Clearly Define and Measure Testing Entry and Exit Criteria**. Minimize the gaps and overlaps in testing by clearly defining the objectives of each test level/cycle and measure against entry and exit criteria to address objectives.
- **Define Test Cases During Design Activities**. Create test cases while executing design activities in order to validate there is a direct correlation between business requirements and test cases.
- **Prioritize What Will be Tested and in What Order**. Plan so that the critical, significant, or riskiest requirements are addressed as early as possible to provide the time needed to resolve possible issues.
- **Develop Solid Test Models**. Develop well-documented, repeatable test models to facilitate analysis and regression testing of identified defects in the current level of test, as well as other test levels.
- **Test with Appropriate User Involvement**. Users will not only take ownership of the system but also have the business expertise and are in the best position to determine and validate if the application conforms to the business requirements.
- **Automate Testing Where Possible**. Use automated testing tools to increase testing execution speed and accuracy within the testing levels.

- **Establish Defect Thresholds**. Clearly define and communicate Service Level Agreements (SLA's) for test level transitions and defect resolution.
- **Exercise End-to-End Business Process Lifecycles Early and Often**. Structure testing to support end to end business processes testing and execute early and often to increase test exposure across the system.

Test Standards

Deloitte follows recognized and agreed upon standards for software testing to follow supreme quality approach towards Quality Assurance. These standards provide the guidelines for standardization of test documents and establish a methodology for implementing, analyzing, and validating the software testing process as well as providing the quality metrics.

- Test documents such as Test Plan, Test Case and Requirement Traceability Matrix are created using pre-defined approved templates.
- Test coverage and Test Case Review checklists are used to make sure coverage and reviewing test cases.
- Standard Test reports and test metrics documents are followed which can be curated to accommodate client specific requirements.

Approach to Non-testable Requirements

Deloitte will work in collaboration with CHFS to determine possible approaches to test difficult to test or untestable requirements. There may be test scenarios that include non-testable requirements necessitating either specific data unavailable to test and/or time dependent escalation rules that take weeks to execute but cannot happen in the testing time frame of the IEES project. Under such scenarios, we will simulate such scenarios by staging test data manually to facilitate the condition necessary. For example, if an email is supposed to go out if a certain failure occurs, which we cannot trigger to occur in the IEES application, then such scenarios are tested by simulating the failure. Deloitte understands conducting integration testing with external operational systems requires availability of an external test system. However, if such an external test environment for systems operational in production is not available, Deloitte works with CHFS and external entities in developing a test stub that simulates the external system interface necessary to conduct testing. Deloitte also employs techniques such as automation and performance tools to test performance requirements, by simulating real world conditions which may not be possible or practical in the given testing phase.

Testing Tools

Microsoft TFS

Deloitte and CHFS will be using Microsoft Visual Studio Team System Team Foundation Server (TFS) to track requirements, usage scenarios, and test scripts. Deloitte and CHFS has successfully used TFS in the past in large project implementations for source code management as well as for project repository for all usage scenarios, use cases, requirements, designs, test scenarios, test cases, test results and other project artifacts.

Testing Tools Integration with Test Activities

The teams deep experience in testing and in the use of Microsoft Team Foundation Server testing tools allow us to quickly develop, organize, execute, and report on testing activities in both manual and automated testing. The following table illustrates how test activities benefit from test tools integrated throughout the testing activities. A customized tool might be developed when it is necessary to assistant testing execution such as a test driver harness or test stub harness to simulate external interfaces to feed in response for components under testing.

Figure 3-72. Testing Tools

Types of Testing	Tools Used	Usage
Unit Testing	MbUnit SonarQube	MbUnit is a generative unit test framework. It gives the end-users the "high order" test fixtures and to the developers the tools to build new custom fixtures without modifying the Core. It implements the simple test pattern and provides new fixture types. Sonar Cube sample reports are generated which represent the bugs, vulnerabilities and code smells to be addressed before deploying code to higher environments.

Types of Testing	Tools Used	Usage
• Integration Testing • Iterative Functional Testing • System Testing • Regression Testing • Interface Testing • Iterative Functional Testing • User Acceptance Testing • Data Migration Testing	TFS REST Assured for API Automation Selenium and Appium	TFS Provides ability to manage Custom test automation. Tester can publish test results to the TFS data warehouse to get better management of test results. It allows these results to be viewed by all other members of the development team and easily determines the relationships between bugs, builds, and test results. Our Interface testing methodology addresses the complexities associated with implementing multiple APIs and Services which includes SDH and HBE Interfaces. We have leveraged REST Assured to optimize test coverage across all key Interfaces, enable test suite reusability, and implement a data-driven test strategy without being contingent upon front-end readiness. Increased automation test coverage (smoke, SIT,UAT, regression) with Selenium and Appium Automation test frameworks.
Security Testing	• HCL AppScan • Microfocus Fortify Static Code Analyzer • Burpsuite	HCL Appscan is a Dynamic Application Security Testing (DAST) tool which helps in identifying security vulnerabilities in the application by performing automated scan Scan the source code with all external and internal libraries Burpsuite is used to perform end to end manual security testing of the applications.
Performance Testing	Visual Studio Team System Profiler (VSTS) DynaTrace SQL Diagnostic Manager- Idera	Allows running customizable tests and collecting critical data. The test reporting Web site feature enables tester easily share test results, analyze data, and manage stored test results. Automated monitoring and reporting of web applications using Dynatrace. Database monitoring and profiling is performed with Idera tool.

Types of Testing	Tools Used	Usage
Usability Testing	• Internet Explorer Toolbar (Web Accessibility Toolbar) by Vison Australia • Compliance Sheriff	The Web Accessibility Toolbar aids manual examination of Web pages for a variety of aspects of usability/accessibility. It consists of a range of functions that: • Identify components of a Web page; • Facilitate the use of 3rd party online applications; • Simulate user experiences via various window sizes and other options; and • Provide links to references and additional resources. Compliance Sheriff runs on a regular basis to check for Section 508 compliance, and the reports are forwarded to the content owners within the agency.
Browser Testing	Visual Studio	Provides comprehensive testing solution that enables tests on IE, FF, and Safari browsers.

Testing Phases

Each testing phase is performed in a dedicated test environment. The advantage of testing in isolated environments is that it allows testing to occur without interrupting the development life cycle, facilitates test data management, and allows the easy reproduction of defects for further analysis. As part of the IEES Solution, Deloitte proposed a multi-faceted testing phases to coordinate and manage planning and development of testing activities. Deloitte's testing approach brings together the business users, developers, and testers. They collaborate to reach a common understanding about what functionality to construct and test within the software life cycle. The team then determines when the tests should happen, what test criteria is needed to validate the features, and what quality aspects should be considered (e.g., functionality, reliability, usability, efficiency, performance). This makes the whole team responsible for quality, not just the testers. Key testing phases are described under the section Testing Processes.

Unit Testing

Unit testing is primarily performed by the development teams as part of the development effort. Deloitte's approach to unit testing provides verification of the hardware or software prior to integration of those items. Unit testing is highly iterative and is an essential aspect of defect management. By performing unit testing as components are developed, defects and issues are identified earlier and as a result are less costly and time-consuming to resolve than defects found later in the testing process.

As a part of the unit test plan, Deloitte will submit the approach to Unit Testing; including targets for unit test coverage and pass rates, for approval to CHFS prior to the commencement of the development phase. The unit testing will begin early in the development life cycle and can continue throughout the life of the project should there be a need to add additional components. Unit testing will take place in the development environment. Deloitte will use MbUnit tool to test the different system components during unit testing. Additionally, a customized tool might be developed to aid in unit testing. Once the application modules have been adequately unit tested, they will be ready for the integration test phase. A member of the Deloitte team will complete the development of software components then move into unit testing. The Deloitte team member who is responsible for component construction is also responsible for identifying and creating test data and completing the unit test process. Deloitte understands and will work with CHFS to submit its approach to Unit Testing including targets for unit test coverage and pass rates, for approval to CHFS prior to the commencement of the development phase.

Integration Testing

Integration Testing validates that the components are functioning as expected when operated individually or as a group. This involves testing the assembled individual components and tests them with other components. The developer evaluates the resulting artifacts while testing the unit as a component of the system, emphasizing regression testing for common objects or other objects that have dependencies to other artifacts. The test plan includes test data, expected inputs

and outputs, and any automated testing to be utilized. By involving selected tests, early validation of functionality is achieved. The Integration testing will begin early in the development life cycle and can continue throughout the life of the project should there be a need to add additional components. Integration Testing will take place in the development environment. Once the application modules have been adequately integrated tested, they will be ready for the iterative functional and system test phase.

Deloitte understands and will work with CHFS to include its approach to Integration Testing, including the recommended environment for Integration Testing, in its Test Plan. Integration testing guidelines shall be included in development standards documentation.

Iterative Functional Testing

Deloitte's team will use an Iterative approach to Functional Testing that leverages test case execution using a subset of System test scripts that verifies the components developed for each logical iteration of the system meet all functional and technical requirements as defined and approved in the Test Plan and the Requirements Definition Phase. The IEES solution will be released to production incrementally in a release approach. The iterative testing approach not only supports the verification of the new functionalities being released, but also includes the verification of the previous existing functionalities. This is achieved by conducting regression testing components developed for each logical iteration of the system as defined in the test plan. Regression testing is an integral part of the iterative test approach and may be performed within each test phase or as a separate test phase by itself. Please refer to the regression testing section for additional details.

System Testing

System Testing is the process of testing the software and required hardware/network infrastructure as a whole. System testing will validate the system by simulating the numerous variations of user process flows (both positive and negative),user or application security and system initiated use cases (both positive and negative), and other application requirements. It confirms that the tested requirements are met in a manner consistent with the system requirements. The structured approach to testing the system assesses the functionality and interoperability of the System and the multiple other systems and subsystems it interacts with, such as databases, hardware, software, rules engine, document management system, identity management system, workflow, interfaces and web services, and their integration with infrastructure into an overall integrated system based on our experience with large-scale HIX and IE systems. System testing is performed in a separate environment to work diligently toward a stable test environment.

Deloitte's approach also includes testing both manual and automated processes to confirm a valid test. By involving selected end users in these tests, early validation of functionality is achieved. The system testing will begin early in the testing life cycle and can continue throughout the life of the project should there be a need to add additional components. System Testing will take place in the system test environment. Once the application modules have been adequately integrated tested, they will be ready for the subsequent functional regression phase.

Interface Testing

Interface Testing is a critical part of systems and integration testing. Interfaces are responsible for sharing information such as verifications and data exchanges. Deloitte's structured approach to testing interfaces is based on experience with large-scale HIX and IE systems. It is important to identify test scripts that will cover the data sensitive needs of various business scenarios. Interface testing is a formal procedure carefully planned and coordinated, focusing on areas of the IEES solution to plan for the completeness of interface development and the readiness of developed interfaces for integration in the wider system. These scenarios are meant to evaluate not only that the featured functionality is performing as expected but also that other related functionality is not impacted by this code. Interface testing verifies that functional requirements for full integration with other groups are in place for testing. The interface testing will begin early in the testing life cycle and can continue throughout the life of the project should there be a need to add additional components. Interface Testing will take place in the QA environment. Once the application modules have been adequately tested, they will be ready for the subsequent functional regression phase.

Regression Testing

Regression testing is an integral part of the iterative test approach and is performed within each test phase or as a separate test phase by itself. The objective of regression testing is to confirm that previously tested application functionality or critical end-to-end business workflows or system performance has not been adversely impacted by the implementation of the following:

- New or updated functionality
- Defect fixes
- Production fixes
- Upgrades to infrastructure

Regression testing also establishes a baseline measure of critical functionalities and technical metrics. Once the baseline is accepted by CHFS, the baseline is used as a gauge when performing subsequent regression tests to confirm that the system achieves the expected level of system performance. The baseline is updated to align with newly released functionality. Our regression test approach takes on the following two forms:

- **Regression — Upgrade.** This type of regression is only applicable when testing a software upgrade. A representative subset of test cases from the previous release would be executed to confirm that the new functionality does not negatively affect existing working production code.
- **Regression — Post Exit Criteria.** To confirm that issues that are detected and corrected and associated with later phases of testing do not affect any previously passed working functionality.

For both types of regression testing, the scope of the test scenarios is typically a subset based on system test scenarios and the identification of key business processes within a release. Deloitte

will work with CHFS to obtain a mutually agreed upon set of scenarios, chosen to promote maximum functional and technical coverage of regression testing. Based testing methodology, regression test scripts will be executed during each iteration before the new IEES solution code is deployed into production. This verifies the quality of the application and builds the CHFS's confidence that newer code changes have not adversely impacted existing functionality. . Our comprehensive automation approach brings in test efficiency during the regression phase

Security Testing

Our approach is tailored to provide a holistic solution for application security testing that will help you make sure risks are identified and addressed early in the software development life cycle. We understand the State requirements and provides custom solutions by reviewing and analyzing the output from automated SAST/DAST and perform additional manual tests to reduce developer effort and provide truly actionable results by eliminating false positives.

The table below describes our security testing techniques that may be used in assessing the in-scope applications. The usage of the tools may be limited to what the CHFS may provide for the purposes of testing. Security testing will be performed in a representative non-production environment. The applicable vulnerabilities are remediated and the controls are then propagated and/or applied across other environments.

Figure 3-74 Vulnerability Assessment and Penetration Testing Techniques.

Security Testing Activity	Description
Application Vulnerability Assessment Testing	**Static Application Security Testing (SAST)** • We will perform a secure code review of the programming code. An automated code review will be performed on the application and webservices source code using a static code analysis tool. Manual validation of the identified SAST tool vulnerabilities will be performed to eliminate false positives. **Dynamic Application Security Testing (DAST)** • We will perform DAST of the in-scope applications and web services in a runtime environment using automated tools to identify potential security vulnerabilities. The specialist performs an (false-positive) analysis of the findings using industry-standard methodologies, and techniques (e.g., OWASP) to determine vulnerabilities, threats, and risks to the application and its data. **Manual Testing** • Manual Testing will be performed to analyze the application and web services for standard and advanced application security findings using industry-standard methodologies, and techniques (e.g., OWASP). This manual testing approach target to identify real-life cyber-attacks, associated threats and uncover any risks to the application by simulated cyberattack against the application. **Manual Code Review** • Deloitte will perform a manual source code review of the programming code to identify violations of security-specific coding rules and guidelines and discover the vulnerabilities not reported by the automated tool.

Performance Testing

The focus of performance testing is to confirm the performance of the entire technical architecture, which includes the performance of software applications, integration, database, network, and the hardware components included within the scope of the implementation. Therefore, it is important to confirm that the performance testing correctly models the anticipated production workload in the new solution. Stress and volume testing are extensions of performance testing. Where performance testing strives to simulate a typical day in the life of the system, stress and volume testing push the system beyond its intended usage levels—-more users are added, more data is added, extended longevity testing is performed. The main objectives of performance testing, stress testing, and scalability testing are to:

- Measure the IEES user and customer experience of application response time before the system goes live.

- Discover performance flaws in the application and supporting infrastructure in a controlled manner so flaws are addressed early on rather than potentially causing severe performance degradation, or even an outage once the application is deployed in production.
- Simulate “high usage” of the system (i.e., stress and volume tests are executed at 120 percent, 150 percent, etc. of normal performance testing), so as to provide indication of how the system would respond if volumes unexpectedly increased significantly.
- Seek any further performance and/or system tuning opportunities.

Deloitte will actively test for performance flaws in the application and supporting infrastructure in a controlled manner so we address flaws early on through additional sizing, rather than potentially causing severe performance degradation, or even an outage once the application is deployed in production.

Load Test

The purpose of load test is to evaluate the IEES system performance under a peak load conditions of current concurrent users and transactional volume metrics. Load test will measure the expected response time, transactional volume, and system resources against simulated real-world user loads.

- Load tests will begin with a minimal user load to validate and verify that scripts have been properly recorded and that the environment is ready for test.
- The scripts will then be ramped up to an estimate of over 110 percent of the current peak concurrent user and transactional volume metrics.

Scalability Test

Measure and identify how far the application and architecture can scale to sustain the anticipated growth of concurrent users and transactional volume.

- Scalability Tests will be run post Load Test completion

Stress Test

This test is essential to determine upper limit capacities for the IEES operations. Scenario volumes will be ramped up to scaled up estimates to identify situations of saturated resources. This is a form of testing that is used to determine the stability of the IEES.

Usability Testing

The usability of an application can be affected by various combinations of technology, design, users and process. Usability testing addresses considerations across these core areas while taking into consideration both quantitative and qualitative measures of usability. Diversity in the user base, as defined by varying levels of skill or knowledge, education level, access to technology,

and familiarity with process among others, is considered in testing. The main objectives of usability testing are to:

- Identify major usability problems - including problems related to the specific skills and expectations of the users
- Obtain measurements for the ability of users to complete tasks within a reasonable amount of time or steps
- Assess users' effectiveness, efficiency and satisfaction
- Provide input back into the design and development process to improve the overall system
- Identify areas of need for potential user training

Deloitte will work with the Commonwealth to identify a role-based approach to usability testing in which we are identifying target user segments and tailor testing plans to the role. Testing will be delivered through a combination of formal usability labs, user research, and automated testing tools. Features of the application that are typically tested in Usability Testing include:

- Ability to find the site
- Home Page for context and guidance
- Navigation
- Site Architecture
- Layout and Design
- Content and Readability
- Search and Results
- Calls to Action

The test strategy will address six key steps to delivering effective usability testing:

Figure 3-77 Key Focus Areas of Usability Testing

Key Component	Key Focus Areas
Define Target User Segments	• Identify the most effective methods for usability testing • Identify up to 3 target user segments for testing • Determine appropriate tasks and scenarios to be tested based on scenarios that mirror tasks that users would perform with the solution • Tailor test scripts to characteristics of target user segments
Identify Success Metrics	• Success metrics may be quantitative or qualitative, or a combination of both • Quantitative success metrics may include task completion within a specified time frame or a percentage of successful tasks. • Qualitative metrics may consist of verbal or anecdotal approval of functionality or look and feel
Execute Usability Tests	• Usability Lab: Scenario and task based testing in a 1:1 Usability Lab for the three groups identified • Accessibility testing tailored to disabled users • Use of automated testing tools for 504 and 508 compliance
Report Results	• Communicate findings on an ongoing basis • Summarize and deliver final results in document form and include findings relating to each of the guiding hypotheses.
Incorporate Feedback	• Usability testing is only valuable if the results can be evaluated and implemented into the solution. • The testing strategy will establish the process by which usability testing results are prioritized, design changes made, and incorporated into the solution

An important component in developing the Usability Testing Strategy is to identify the appropriate target user groups and that the testing of a particular group includes an appropriate representation of users. While developing the strategy, Deloitte will work with the Commonwealth to identify and define target user segments for usability testing. These groups are identified as:

- Internal and external users
- Power users
- Users with limited computer skills
- Users who will require training in the system to complete their daily work

- Users with disabilities
- Prospective new users

User Acceptance Testing

UAT assesses end-to-end business and technical functionality using actual production rules and data resembling production operations. Deloitte's UAT plan outlines an approach for planning, testing, and verifying the functionality of the solution in accordance with the business and technical requirements. It defines the approach to confirming that the requirements are implemented successfully in the solution.

OATS develops UAT test conditions, scenarios, and scripts. The test scripts represent the new IEES functionality that the user would typically use, including functional, technical, and application security testing. These scripts also allow a repeatable testing process providing regression testing as needed for changes made.

Deloitte will work together with CHFS to identify the production data to use for each test case within each business scenario.

User Acceptance Execution Facilitation

A key element of success throughout User Acceptance Testing is the planning, coordination, facilitation and communication activities that take place among stakeholders on a regular basis, as described in the following sections.

Pre User Acceptance Test Execution

Prior to the start of User Acceptance Testing, as needed, there will be a kick-off meeting to communicate details of the UAT plan, execution approach, and other key information to the entire UAT team

User Acceptance Test Execution

The following activities are performed to coordinate, facilitate, and communicate the progress of testing:

- Execute UAT scripts and log defects (performed by the CHFS/SMEs/Testers)
- Produce a weekly schedule of planned case executions and refine it on a daily basis as needed, taking into consideration the dependencies and sequencing activities that need to occur as part of the case executions, including activities at the different testing locations (performed by the CHFS UAT Manager with assistance from Deloitte)
- Record the status of daily case executions real-time in Test Management Tool as the scripts are executed, to enable real-time reporting and analysis of the test execution status (performed by CHFS/SMEs/testers with assistance from Deloitte)

- Conduct daily UAT touch point calls with the CHFS UAT Manager, his designee(s), and the Deloitte testing lead and his designee(s), making the key testing statistics from that day available for distribution to the CHFS/Project Managers. This meeting serves as an integrated forum to provide a status of testing for the day as well as discuss key issues and defects that require priority attention
- Work through defects and questions that arise through the testing process via daily communications and interactions between the Deloitte UAT Support Team and Development, Data, and Infrastructure Team members
- Coordinate communications with the CHFS/Site Leads and SMEs for any functional clarifications and defect resolutions that require additional subject matter input (performed by Deloitte)
- Participate in defect triage meetings and facilitate communication between the Deloitte Team and SMEs/testers in those sites as required (performed by CHFS Site Leads)
- Resolve test case defects that may require some clarification from the SMEs to validate the intended test steps, recording and managing any case defects that require a fix by Deloitte through Test Management Tool
- Distribute UAT execution reports to the CHFS/Project Managers and Deloitte indicating the status of case executions and defects (distributed by CHFS UAT Manager with secondary support from Deloitte)
- Discuss UAT status updates at weekly project status meetings and at regularly scheduled Executive Leadership and Steering Committee meetings

Data Migration Testing

Data conversion and migration is one of the riskiest parts of any large technology initiative such as IEES. Data migration is done only in cases of new projects (for example the Childcare implementation where data from old system was being loaded into IEES or the implementation of the State Based marketplace where data from the federal marketplace was loaded into IEES). The Deloitte team brings production proven automated data conversion experience from similar statewide system implementations and a tested conversion approach to migrate data meeting the business needs of CHFS. In addition, Deloitte will actively engage CHFS in the planning, designing, construction, testing and execution of migration. Involvement from all parties is critical to a smooth migration. Deloitte will work with CHFS to finalize the data migration test plan where in exception tolerance levels shall be agreed with and approved by CHFS prior to the commencement of migration testing. Test results will be reviewed and approved by CHFS prior to commencement of production Data Migration.

Conversion testing provides the benefit of improved data conversion rate, reduced data cleanup efforts, and improved data quality. This activity involves setting up the environments, testing the migration programs, validating results, and preparing for the final execution of the migration. This is one of the more extensive pieces of the migration process. Deloitte is experienced in working with other states for data conversion efforts and has an in depth understanding of the requirements and considerations of preparing and testing the conversion load process. The

testing of the conversion programs starts during the cleansing phase, however the focus during the load process shifts to integration testing and testing conversion outputs against the requirements. As part of this activity, capacity planning and estimates of data volume are finalized. Control counts are used to monitor and control the conversion process as well as test the conversion timings as we run mock conversions during the testing phase of the migration.

The following activities are used to test the migration process and programs:

- **Development, System Integration, User Acceptance, and Conversion Environments.** These environments are configured to support the repeated testing of the conversion process and associated code artifacts.
- **Preparing Test Data**. Data for testing the conversion programs is developed, pre-conversion reports are run, and test scenarios are created to compare the converted data and test the transformation logic
- **Mock Conversions (Dry Runs).** Periodic mock conversions for an entire set of source data are conducted to perform load testing and testing of conversion programs/code in a test environment. During this phase, we also determine how the converted data runs against test scenarios and run the data into the IEES database tables to validate the integrity and completeness of the data to support the new application, both from the database side and the application side. Since the mock conversions require the data to be of the same quality as the actual conversion data and require the new application to support converted data, this step needs to take place later in the overall application development.

Testing Methodologies

Usability Labs

Usability Labs are a core approach employed by Deloitte to evaluate the effectiveness of an application. Usability Lab testing consists of one-on-one interviews, one facilitator with one user, in which each of the participants will be supplied with a computer and monitor. Participants will be asked to perform specific tasks in predetermined areas of the site.

Automated Testing

At the Commonwealth of Kentucky, we have established an Automation Foundry that continually defines new and improved frameworks and uses emerging technologies to augment test automation for you. We can personalize automation suites to match trends we observe in production or to heavily test areas of the system that are high impact and have been promoting the most code changes. Our comprehensive automation approach brings in test efficiency during the regression phase

We have leveraged a combination of Selenium and Appium for increased automation test coverage of UI based component testing. All the 7 UI applications (Worker Portal, Salesforce Self Service Portal, Waiver, KLOCS, Agent Portal, Issuer Portal and Provider Portal) have been automated using the current framework. We have implemented reusable components and have created a library to expand the automation of UI functionality and improve re-usability.

Deloitte's custom-built test automation solutions provide low cost, low complexity, and highly effective solutions in testing. They offer a robust capability to overcome challenges posed by traditional automation tools across the various stages of the test lifecycle.

Software testing tools such as Compliance Sherriff will be used to perform automated testing to validate 504 and 508 compliance.

Language Testing

Deloitte understands and acknowledges the Test Plan will include a strategy for testing the IEES system in both English and Spanish. Deloitte's approach also includes testing manual processes to confirm a valid test. The language testing will begin early in the testing life cycle and can continue throughout the life of the project should there be a need to add additional components. By involving bi-lingual testers in these tests, early validation of language testing is achieved. Language testing will take place in the system testing environment. Once the application modules have been adequately tested, they will be ready for the subsequent UAT regression phase. Deloitte's approach to language testing follows system testing by simulating the numerous variations of user process flows (both positive and negative),user or application security and system initiated use cases (both positive and negative), and other application requirements.

Browser Testing

Deloitte understands and acknowledges the Browser testing shall be performed using a subset of System test scripts that promotes maximum system coverage. Deloitte will test the web components of the IEES solution in the current release and at least one previous version of Microsoft's Internet Explorer, Mozilla's FireFox, Google's Chrome, and Safari. Deloitte understands and acknowledges the machine configurations to perform all necessary browser testing will be provided by Deloitte as per the Test Plan.

Test Techniques and Methods

Deloitte's technique and method includes manual testing, automation of functional tests, regression, and generation of virtual users to simulate load in an environment that matches production. Using Team Foundation Server (TFS), test cases are grouped into test suites composed of manual or automated tests to manage and execute the testing for a particular release, or for any controlled ad hoc testing. A particular test suite targets the testing for a single component in isolated testing. Test cases are selected and assembled into a test suite to guide ad hoc test needs in any test cycle. In addition to support manual ad hoc testing, tool features will be

leveraged that record automation scripts along test execution so it can playback the test scenarios the tester might envision.

The iterative testing approach not only supports the verification of the new functionalities being released, but also includes the verification of the previous existing functionalities. This is achieved by conducting regression testing in each release cycle. Regression testing is an integral part of the iterative test approach and may be performed within each test phase or as a separate test phase by itself.

For regression testing, the scope of the test cases is typically a subset based on technical analysis and the identification of key business processes within a release. Based on our testing methodology, we will execute regression test scripts every iteration before the IEES code is deployed into production.

The use of automation technique is the key to reducing the time required for testing, improving quality, and increasing test coverage. Our testing automation approach provides methodology, guidance, and an automation testing framework. The development of automation scripts follows the code development process going through design, coding, testing, and code configuration control activities. Our testing automation approach includes repeatable processes that provide the following benefits to the Commonwealth such as:

- **Increased Efficiency of Test Execution**. Test automation speeds up test case execution. It saves test execution time and enables efficient testing schedules. It provides the ability to execute scripts across browsers and operating systems.
- **Increased Quality of Test Coverage**. Data driven testing automation extends test coverage of the release.
- **Increased Testing Repeatability**. Test script automation allows test cases to be executed repeatedly in each iteration release to maintain the high quality standards along incremental releases.
- **Replication of Defects**. Automation of test scripts provides for quick and easy replication of software defects and verification of defect fixes.

Automation will begin with a smoke test, and then move to regression testing to include frequently performed test cases, time-consuming test cases, and high precisions test cases. In each iteration release, we keep adding automated test cases to cover new functionality whenever a new component is available for testing. Our automation testing framework includes components to allow effective creation and maintenance of automation test scripts. It also supports automated test case execution in a repeatable and unattended way.

Testing Processes

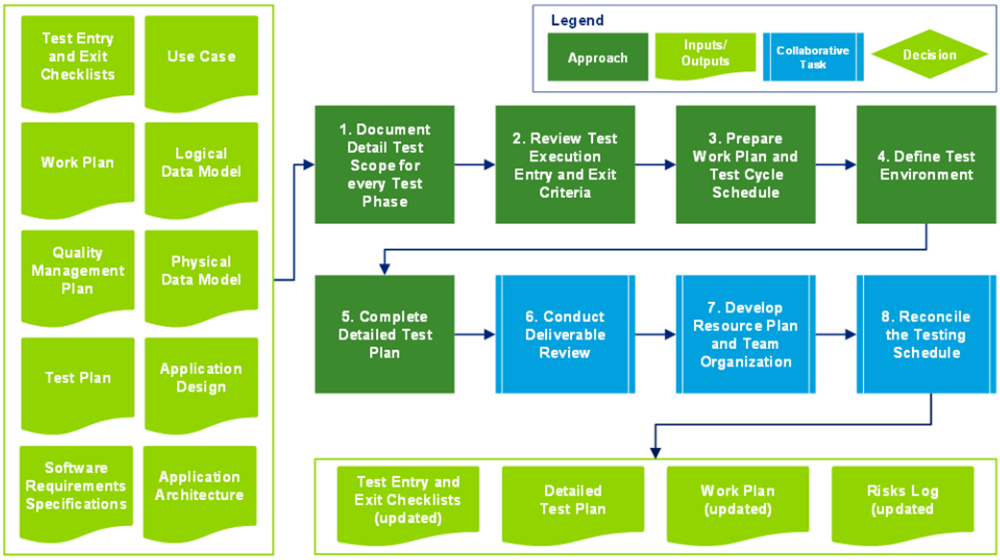
The IEES project requires a strong commitment to each phase of testing. Deloitte's experience allows us to bring a structured approach to testing the new IEES system. For CHFS's

implementation to be effective, quality must be built in from the beginning and not just tested at the end. We bring an established testing approach that includes a transparent and broad process supporting development and use a fully integrated toolset throughout the entire testing life cycle to confirm requirements traceability. We provide an organized, well documented, and structured process for managing and executing functional, technical, and deployment testing to drive effectiveness at CHFS.

Preparation, Orientation and Kickoff

A key element of success throughout testing is the planning, coordination, facilitation and communication activities that take place among stakeholders on a regular basis. Successful planning and development of the testing strategy depends upon early identification of external stakeholders and owners. The following graphic outlines the test planning and preparation process from our EVD for SI method that will be used on the IEES project.

Figure 3-62 IEES Process Flow for Creating a Detailed Test Plan



Deloitte's detailed test plan methodology extends previously created deliverables and produces a test plan through a collaborative process with CHFS.

The table below outlines the various activities in the test preparation, orientation and kickoff for the IEES effort.

Figure 3-63. Test Preparation, Orientation and Kickoff.

Key Activities	Description
Identify testing stakeholders.	Identify the appropriate stakeholders for the various testing activities, including requirements stakeholders and end-user acceptance stakeholders. These testing stakeholders include representatives from CHFS business and IT (including business analysts and technical SMEs) to serve as an integrated group providing sufficient representation of the organization.
Determine external coordination requirements.	Identify any external or third-party systems that affect the target system. This includes, but is not limited to, external partners or customers that interface with the system; and downstream systems, partners, or customers that receive output from the target system or provide input into the target system. In addition, coordination of schedules, resources, and content validation may also be required to successfully complete full-cycle testing activities.
Confirm test phases.	In collaboration with the CHFS, evaluation of the system being developed, and input from project leadership, confirm which test types (e.g., System, Integration, User Acceptance) will be executed. Document the specific test types that will be a part of the testing process
Document the structure and number of roles needed to plan and execute test activities.	Provide sufficient detail with respect to the specific roles and responsibilities for each individual participating in the testing and defect management process. The list includes, but is not limited to, the following: • Specialized tester roles (e.g., Security, Accessibility, Load, Regression, Interface, or Batch) • Defect managers • Developers • Test data management resources • Configuration management resources • Environmental/administration resources
Enable testing tools.	Confirm all testing tools are installed and configured for all identified test computers. Confirm that any access permissions have been requested and acquired for any users needing such privilege.
Provide training on using testing workstations	Provide training on using testing workstations, Test Management Tool, and other testing tools. Conduct training and on boarding for new testers so they are familiar with how to execute test scripts and record the results, including how to assess severity, and how to log, retest, and close defects in a consistent manner to maintain the overall standards and quality of the testing process (performed by each CHFS/SME/tester)
Review test readiness criteria.	Confirm the readiness of the testing environment for test execution by conducting shakedown tests before each cycle when data is refreshed (Deloitte Team with secondary support from CHFS/SMEs/Testers at the discretion of the CHFS Project Manager)
Conduct Kick Off	Conduct one or more kick-off meetings to communicate details of the Test plan, execution approach, and other key information to the entire testing team (conducted by CHFS Project Manager, CHFS Site Lead(s), and supported by the Deloitte team)

Deloitte's preparation, orientation and kickoff activities include early engagement of key CHFS stakeholders in planning to promote a highly successful project testing effort.

Test Data

Deloitte has planned detailed test data prepared to address the functional and technical requirements of the given release. This includes the manual creation of new test data via the screens already functional at the beginning of testing – an advantage attained by leveraging experiences implementing what is already successfully operating in production environments. It also includes the creation of data from automated test scripts and again, with Deloitte, the Commonwealth benefits from past experiences and ability to leverage the functioning scripts that have been vetted in similar systems implementations. We also define the importance of, and use of data sourced from conversion testing. For example, if the release includes the functionality for redetermination of benefits, we create test data from the two sources that will reflect the data reality of production once the system goes live: converted cases and newly created cases. We use converted data where the converted case has already reached various states of redetermination:

- Cases where the redetermination is due
- Cases where the redetermination has been received back for processing, including testing examples where the information received is complete and incomplete
- Cases where the redetermination is past due, and negative action is warranted on the case

We use the "aging" environment to simulate a newly created case which has aged to the various stages of redetermination, and repeat the same tests as outlined for converted case data. In a technical example, if the release includes an interface that relies upon data from other batch jobs, our tests include technical aspects such as running all of the related jobs in their correct sequence to confirm that batch job dependency rules are correctly invoked.

User Acceptance Testing

The principal source for data for UAT is the converted data. This is essential because this is the same data that will be most predominantly used as soon as the system goes live. Equally important is the creation of new data through simulated applications, reported changes to existing case data, and events such as redetermination. Data that is brought into the system from an external interface is also included in the UAT test data set to validate the correct functioning of interfaces. All of these data sources are documented in the UAT Plan including details on how the data is made available to the testers.

Data Migration Testing

An important piece of the testing strategy is the use of production similar data. By using the data that is used for a production conversion run, we are able to better analyze conversion timings and benchmarks, data discrepancies, and the correctness of the defined conversion logic.

Anonymous Data for Testing Purposes

Obfuscating production data for testing and training purposes is done to protect sensitive information from a multitude of threats posed both inside and outside the organization. Data masking uses an irreversible process to replace sensitive data with realistic-looking, scrubbed data. It does this based on masking rules that confirm the original data cannot be recovered. Test data will be created by cleaning real CHFS customer information, including name, address, and social security number (SSN) and inserting test data by using a randomly generated combination from a random information generation database program.

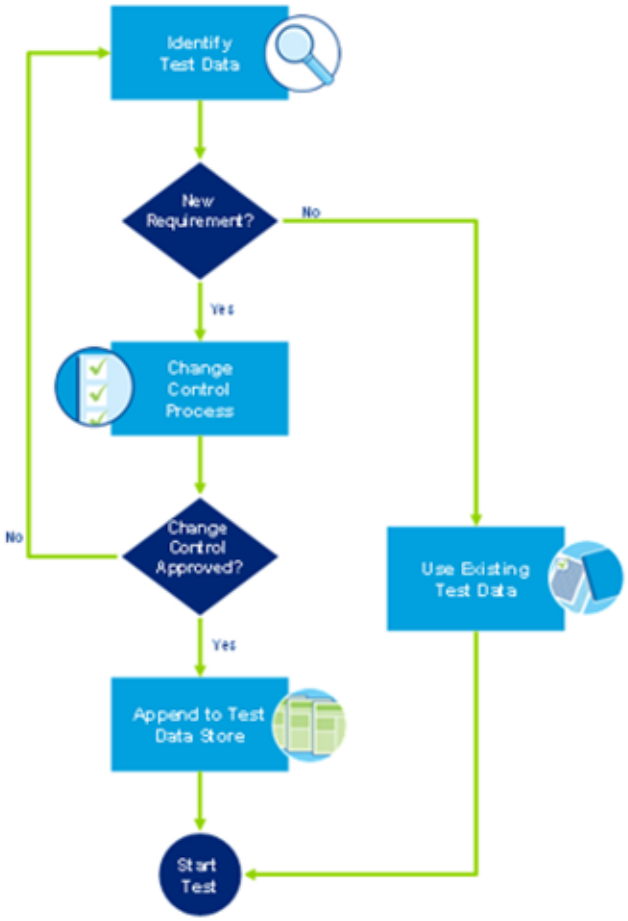
The random generator is loaded with a pre-defined list of first names, last name, addresses, and SSNs and the program assigns each of these to the test data randomly to create realistic data that is completely fictional.

Test Data Refresh

Test data is refreshed, disguised, and readily available throughout the implementation by having an efficient data extract, data disguise, and data execution model that is flexible yet repeatable as shown in the figure that follows.

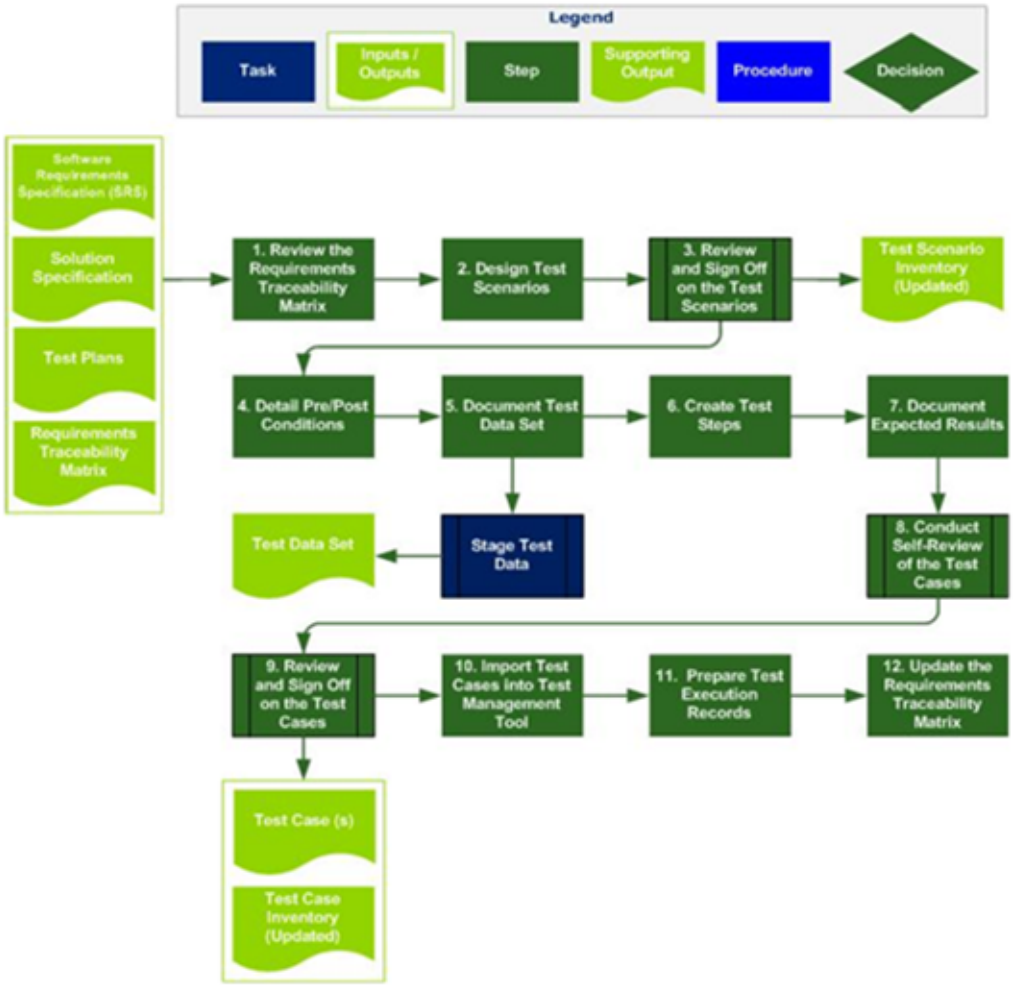
At the end of each system test cycle within a phase, test scenarios will be reviewed and include any appropriate changes in the test plan for the subsequent cycles. Data will be refreshed for each release in the test environments.

Figure 3-61 Test Data Refresh Model
Test Data Refresh Model promotes a stable system test data during each test cycle.



Test Development

The objective of this process is to identify and prepare all test data required for the test effort. The identification, creation of test data may occur in parallel with the test case development. All subsequent iterations of the test or environment refresh will go through this task to prepare incremental test data.



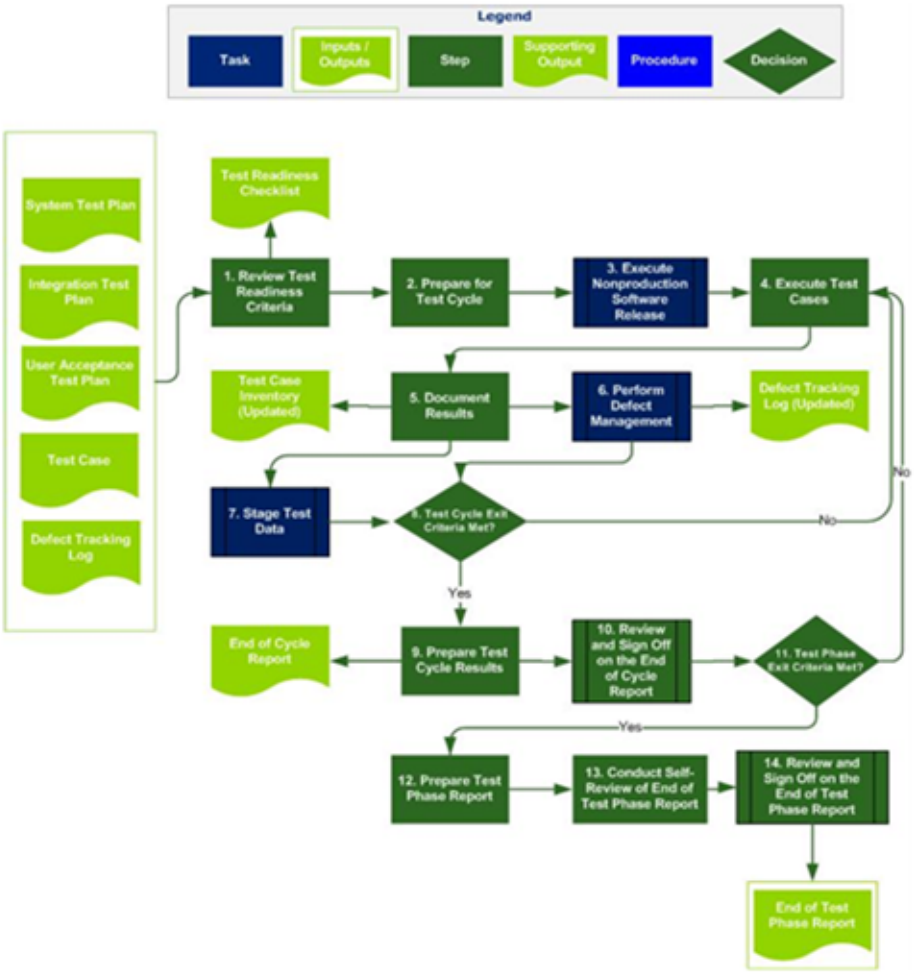
Test Execution

During test execution, the team executes a test cycle as documented in the test plan for a specific test type. As a part of the test plan, a set number of test cycles will be scheduled for execution. Each test cycle has a set number of test cases that will be executed according to a specific time frame. This task will execute the test cycles planned during the previous task. We then evaluate and compare the analyzed results against predetermined thresholds, and communicate how well a software build specifies the stated requirements. As the test cycle is executed, any defects or

issues are reported during this process and resolved by the team. At the conclusion of the test execution cycle, the metrics are reported.

These metrics are compared against set thresholds to determine if the build meets a specified set of requirements and can be promoted. The following figure outlines the process for test execution Deloitte will use for the IEES.

Figure 3-65. IEES Process Flow for Executing Test Cases



Deloitte's detailed test execution process extends previously created deliverables to conduct successful and thorough test execution on the IEES system. By executing the testing tasks outlined in Deloitte's approach CHFS will have the confidence that the following objectives are being achieved:

- Find and document defects in software quality
- Manage defects in software quality to resolution
- Advise stakeholders/clients/project team members on the perceived quality of software

- Confirm the assumptions made in requirement and design specifications through concrete demonstration
- Verify the software product works as designed
- Verify the requirements are implemented appropriately

Test Monitoring

Microsoft TFS will be used as the defect tracking system to meet CHFS specific needs. Defects are managed and monitored from three perspectives: 1) logging and tracking the problem through discovery, resolution, and closure; 2) providing for and managing the communication among interested parties, and; 3) providing notification/escalation procedures to keep the Deloitte management and CHFS management aware of the status of problems.

The progress of issue resolution is communicated through notification and escalation procedures tailored to specific client requirements. Designated points of contact are identified for notification of problems at multiple levels of escalation. After prescribed periods of time, notification is escalated to the next level of management specified within the notification rules for each party.

The information outlined in the table that follows should be collected at the minimum for each defect. Detailed standards and guidelines provide testers from both the Deloitte team and CHFS with guidance on how to document and report defects.

Figure 3-66. Defect Elements.

Defect Elements	Description
Title	A brief description of the defect
Description	A more detailed description of the defect
Severity	An initial determination of the severity of the defect based upon standards and guidelines
User Impact	A description of the impact of the defect to the user or tester
Priority	An initial determination on the priority of the defect based upon standards and guidelines
Functionality Impact	What is the effect of the defect on the functional performance of the system (e.g., is it a “cosmetic” defect or a functional defect?)
Reported Date	The date on which the defect was first reported
Resolution	A description of the resolution that took place
Verification Date	The date the defect was assessed to be resolved by the testing team

The standards and guidelines for both reporting detection of defects as well as resolution of defects, consist of qualitative values, an example of which is outlined in the table that follows, providing key inputs in process improvement and, ultimately, prevention.

Figure 3-67. Defect Quantitative Values.

Defect Values	Definition	Quantitative Example
1 – Critical	Defect prevents developers from developing or testers from testing. Software crashes, hangs, or causes loss of data. There is no workaround.	40% or more test cases blocked
2- Major	Important functionality with dependencies is defective. There is a cumbersome workaround, if any. Defect causes delays in release from test.	20%-40% test cases blocked
3 – Medium	There is a loss of functionality and a workaround in place but undesirable and time/effort consuming.	5%-20% test cases blocked
4 - Minor	There is a small loss of functionality, and a workaround is acceptable and unnoticeable.	< 5% cases blocked
5 - Trivial	This is a cosmetic problem, such as misspelled words with no effect on the development cycle.	No block

Defect Management Responsibilities

The table that follows depicts the team responsibilities throughout the life cycle of defects. Each stakeholder is responsible for carrying out the specific responsibilities necessary to collect, document, track, and resolve defects.

Figure 3-68. Stakeholder Testing Responsibilities.

Stakeholders	Responsibilities
Deloitte Project Manager	• Work to coordinate across team in assisting prioritizing defects • Manage resource constraints and coordinate across Design, Development, and Testing teams • Work with state stakeholders in release planning • Facilitate Defect Review Board for large effort or impact defects • Coordinate efforts for process improvement based upon defects detected
Deloitte Development Team Leads	• Coordinate resources for defect management

Stakeholders	Responsibilities
	• Assist in prioritization and development team effort estimation
Deloitte Test Team Leads	• Determine test schedule • Coordinate test teams efforts across testing • Assist in determining priority and impact for defects
Deloitte Development Team	• Work to resolve defects • Perform analysis of defects
Deloitte Test Team	• Identify defects • Assist Development Team in recreating defects • Assess defects have been resolved
CHFS Project Management Stakeholders	• Identify defects • Assist in prioritizing defects and determining user impacts • Participate in defect review for large effort or impact defects

Test Status Meetings and Reporting

Deloitte's testing methodology provides a tested set of processes and templates that will be used throughout the projects to monitor and control the testing activities and the status of defects detected during the phases of testing as defined in the Test Plan. Testing status will be covered as part of weekly status meetings on a regular basis and during key phases of testing, Deloitte may institute a separate test status meeting for involved parties.

Deloitte will provide CHFS with a Test and Defect Status Report. This report will be used to provide regular updates on the status of testing activities. The same report format will be used across all test phases. Using the same report format for all test phases increases the efficiency of the report generation process and provides the stakeholders with a consistent structure that quickly becomes familiar.

Closure Evaluation Criteria

Deloitte recognizes the importance of the Final Testing Report in communicating the results, findings and closure evaluation criteria from the various testing phases. The End of Phase Test Report is used by the test manager to outline the overall status of the testing, test cycle summary, overall test results summary, exit criteria summary and tracking any deviations from the original plan, open issues and risks, and the test phase defect analysis.

This report summarizes the testing results from all testing activities. Results of the testing can be sorted or grouped by component, module or package and include the following metrics:

- Number of Test Suites
- Number of Test Cases/Scripts
- Number/ratio of failed and passed tests
- Code Coverage percentage (amount of code executed by tests)
- Number of defects identified
- Latest status of defects (e.g. number of defects resolved, rejected, withdrawn, deferred)
- Number of outstanding defects and issues

In addition to testing metrics, the Final Testing Report includes additional findings in the following areas:

- Confirmation that test cases have been properly executed in each test type
- Description, impact and resolution plan for outstanding defects and issues
- Defect trends over time (e.g. trends in defect priority, type, repair time, severity)
- Completion status of testing exit criteria

When the Final Testing Report has been completed, it will be published to stakeholders for review. Additionally, Deloitte conducts a final walk-through of the testing results with Project Management and Project Stakeholders.

Approach to Creating Test Environments

Deloitte understands the importance of a clear distinction between development and testing environments. The proposed environment plan provides a controlled migration process from development and test environments to the production environment. Stable testing data and an environment that mimics the production environment allows for consistently and properly recreate defects and allow efficient repairs of any production problems. The Deloitte team will implement an instance management and transport strategy to manage the testing process. There is no configuration access provided to any environments other than the development environment. Deloitte will make any configuration or code changes identified during testing in the development environment and then transport to the testing environment for re-testing.

Our Test Environment Management Plan

Deloitte will utilize VMWare virtual machines for each development and testing environment. This virtual environment approach allows the creation of isolated environments based on CHFS' needs. Each virtual environment includes the database, network, and application software needed according to the usage of the environment and the types of testing that are performed.

Code Migration and Testing through Environments

Deloitte's environment approach segregates code between environments so different testing activities can occur independently. Each environment has specific entry and exit criteria that must be met before code can be promoted to the next environment.

Development is performed on the local machines of each developer, with a shared database development environment. When development of a component and its associated unit tests are complete, the developer must compile and build the code on their local machine. They must then verify the unit tests execute as expected. The developer can then check the code into the TFS source control.

The Integration environment receives code that compiles and builds successfully. It must also contain unit tests for all in-scope functionality and meet a minimum unit test pass rate and code coverage percent. The Integration Environment is used to conduct automated Developer Testing (Unit and Integration Testing) and manual or automated integration testing to verify the code base is stable. Any automated tests can be re-run in subsequent environments for regression testing purposes.

The System Test environment receives code that meets a minimum unit test pass rate and code coverage percent (the minimum criteria are higher than the Integration environment). The System Testing environment is used to conduct functionality and interoperability tests of the System and the multiple other systems and subsystems it interacts with, such as databases, hardware, software, rules engine, document management system, identity management system, workflow, interfaces and web services. Regression testing is automated using a subset of test cases used for system testing to verify the system is stable. System Testing can be manual or automated and is used to verify that the system is working end-to-end.

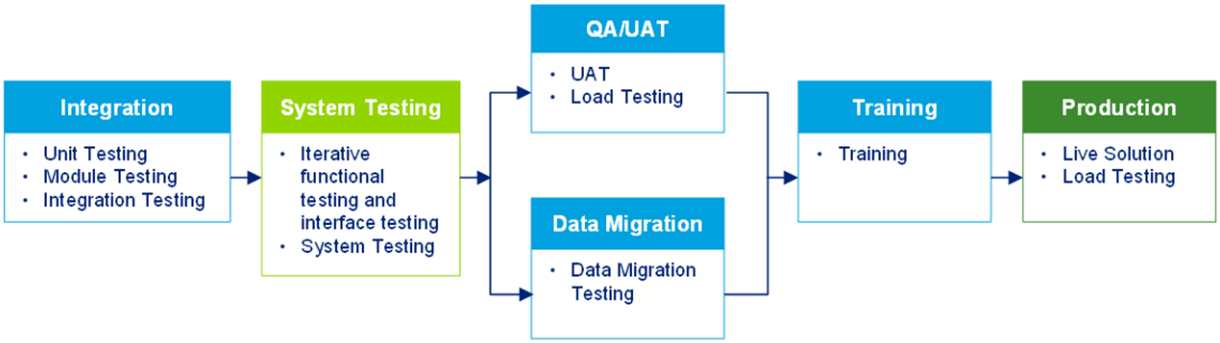
The Quality Assurance (QA) and User Acceptance Testing (UAT) environment receives a stable code base that has passed Integration and System Testing. The entry criteria for moving code to QA/UAT are more stringent than for Integration or Systems. This environment is used for User Acceptance Testing (UAT), performance Testing and Load Testing. UAT verifies that the system is ready to perform all required functions and meets all contract requirements from an end user perspective. Load Testing verifies that the system can handle typical processing loads. It may also include performance, stress and scalability testing, depending on the nature of changes from previous releases.

The Data Migration environment also receives a stable code base that has passed Integration and System Testing. Like QA/UAT, the entry criteria for moving code to this environment are more stringent than for Integration or Systems. This environment is used for Conversion Testing which contains migrated or converted data and verifies that the solution functions properly with this data.

The Training environment receives production ready code after the testing has been completed and accepted in the previous environments. This environment serves as a production like staging environment to validate that the solution can be migrated to Production. It is used for training end users on the production ready system.

The following figure depicts the progression of code and testing across the IEES environments from integration through production.

Figure 3-71. Code Progression and Testing through Environments.



Multiple environments provide the ability to separate distinct testing efforts. The figure above depicts the flow of code through the IEES virtual environments and the types of testing that occur in each environment.

Additional Testing Responsibilities

Deloitte understands and acknowledges they will be developing all test conditions, scenarios and scripts. Deloitte will work within mandated project timelines to obtain approval by CHFS through the deliverable process prior to execution of any test phase. Deloitte understands and is also responsible for preparation of all test data, including identifying data required for each test phase that may require collaboration from Commonwealth resources to acquire that data.

Deloitte understands and acknowledges that all review milestones for System design and the System Design documentation is kept up to date with updates to the design which occurs due to changes or fixes that arise in the Testing Phase. Deloitte team realizes IEES's need to meet CMS guidelines and it is criticality for the solution's production deployment and future sustainability. Deloitte's team carefully plans and aligns the IEES solution delivery milestones with the CMS Certification processes, IEES Enterprise Roadmap, and CMS Exchange Life Cycle. Deloitte will offer guidance and support so that IEES system meets the criteria required for CMS certification.

Appendix: PHE Testing Plan Presentation

Deloitte.

TEAM KENTUCKY
CABINET FOR HEALTH AND FAMILY SERVICES

Kentucky Integrated Eligibility and Enrollment System (IEES)

Commonwealth of Kentucky
Cabinet for Health and Family Services

UAT Test Panning
January 17, 2023

Table of Contents

PHE Scope

CR	Release	Release Date	Part of PHE	SIT TCs	UAT TCs
CR 1448: Medicaid Renewal Realignment - redistributes the accumulated case load. The cases will be distributed based on agency provided threshold limits and other distribution factors such as prioritization based on age and potential QHP qualification	23.03	3/23/2023	Yes	74	44
CR 1613: Facilitated Enrollment – allows IEES to leverage SNAP eligibility data during the ex-parte renewal process in order to redetermine Medicaid eligibility	23.03	3/31/2023	Yes	18	11
CR 1623: Unwinding Passive Renewals (AVS) – IEES will not generate a Request for Information (RFI) for resources when the Asset Verification Service (AVS) returns zero results	23.03	3/31/2023	Yes	8	5
CR 1610: Unwinding – IEES will rollback changes made at the beginning of the PHE as well as allowing continuous eligibility until the Medicaid renewal is complete	23.04	3/31/2023	Yes	TBD	TBD
CR 1621: Returned Mail Bot – IEES will systematically read returned mail and perform the required updates, including outreach (robocalls, nudges) and the addition of a message to Worker Portal and the Self Service Portal for those who have returned mail	23.04	4/28/2023	No	TBD	TBD
CR 1619: Unwinding - Fair Hearing Extension – extends the timeframe to take final administrative action	23.04	4/28/2023	No	TBD	TBD
CR 1616: Renewal Nudges – send nudges to inform Medicaid recipients of their upcoming renewal.	Already In Prod (22.07. Phase 1 22.11. Phase 2 22.12. Phase 3)	N/A	Yes	N/A	N/A
CR 1626: MRT Decision Automation – systematically review the Medical Review Team (MRT) determination	Already In Prod (22.09)	N/A	No	N/A	N/A

Notes: All items above are related to PHE but not all items will be a part of the initial PHE window. Also, items in green have already been deployed to PROD with configurations to turn functionality on once PHE ends.

Testing Timeline

PHE Releases	Testing Timelines (SIT/UAT)	Alignment with M&O Releases	Jan '23	Feb '23	March '23	April '23
Release 23.03	SIT Testing	M&O Go-Live: **3/31**				
	UAT Testing					
Release 23.04	SIT Testing	M&O Go-Live: **4/28**				
	UAT Testing					

Notes:

- All PHE UAT timelines will be merged with M&O GO Lives Dates for better alignment with other changes being delivered as a part of the regular monthly releases. These dates are tentative and solely a projection. Timelines will not be final until confirmation is received from the KY DMS team.

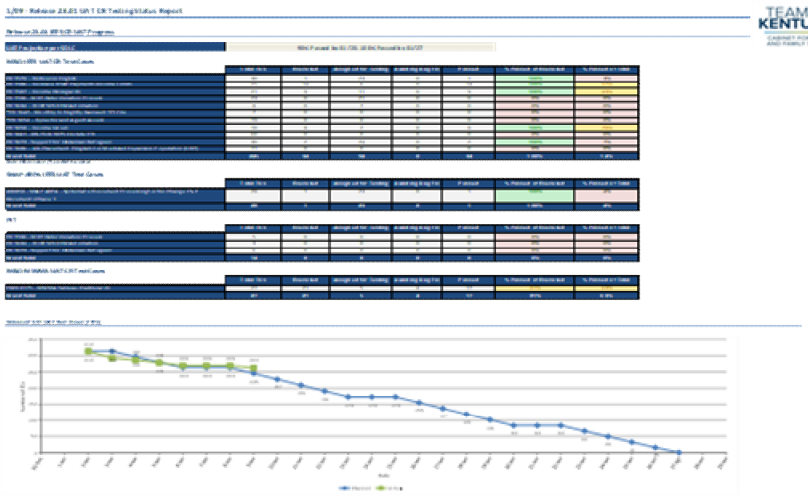
OATS UAT Test Case Scripting/Selection

Test Cases count	SIT TC Count (Planned)	UAT TC Count (Forecast)
Release 23.03	XX	XX
Release 23.04	XX	XX

- The above UAT Test case counts are approx. 60% of Total SIT planned test cases. Historically, this is a method used for UAT Test case planning
- OATS Team can review the current SIT Test Bed and finalize the Total Test Scenario and decide if additional coverage beyond 60% is needed.
- Release 23.03 SIT Test cases are under way
- Release 23.04 Test planning is in progress and is subjected to vary based on final counts. Expected to be finalized by 05/01 (tentative) before execution start date
- High Level Functionalchanges to be reviewed in the Official Kick Off meeting with the Tester before the UAT execution begins.

Weekly Cadence: SIT/UAT Reporting

Daily reporting of test case execution and defects found will be provided by PHE change request. Example of 23.01 burndown provided below. Same burndown will be built to reflect PHE burndown during 23.03 & 23.04.



Automated Regression

Regression Testing will include the re -execution of PHE scenarios to ensure that the constant builds into the wider system that were already executed were not altered by the frequent weekly UAT builds. A robust master Regression suite will also be executed post code freeze to ensure end to end health check of the system as a whole. Below is a sample of the regression suite executed and reviewed by OATS at the end of December 2022 (22.12) release.

Task Description	Start Date	End Date	Status	Total TCs	TCs Completed	TCs Blocked	TCs Pending	%Complete
22.12 UAT ORT WP	12/9/2022	12/16/2022	Completed	128	128	0	0	100%
22.12 UAT ORT MWMA	12/9/2022	12/16/2022	Completed	8	8	0	0	100%
22.12 UAT ORT SSP	12/9/2022	12/16/2022	Completed	10	10	0	0	100%
22.12 UAT ORT SBE R4	12/9/2022	12/16/2022	Completed	17	17	0	0	100%
22.12 UAT ORT SBE R1	12/9/2022	12/16/2022	Completed	3	3	0	0	100%
TOTAL				166	166	0	0	100%

Operational Readiness Testing (ORT)

UAT testers will also execute a set of manual scripts to ensure that the constant CR specific builds into the wider system executed early in the release month were not altered by the frequent weekly UAT builds. Below is a snapshot of what the manual and automation report will look like.

Release 23.03 ORT						
	Total TCs	Executed	Pending	Passed	% Passed of Executed	% Passed of Total
Manual ORT- PHE Focused	20	0	0	0	0%	0%
Manual M&O ORT	10	0	0	0	0%	0%
Automation ORT	136	0	0	0	0%	0%
Total	166	0	0	0	0%	0%